TUM School of Management

Technische Universität München

Proceedings of the 14th International Conference on Project Management and Scheduling

March 30th – April 2nd 2014 · TUM School of Management

Editors
Thomas Fliedner
Rainer Kolisch
Anulark Naber

Proceedings of the
14th International Conference
on Project Management and Scheduling
March 30th – April 2nd 2014, TUM School of Management

Editors: Thomas Fliedner
Rainer Kolisch
Anulark Naber

TUM School of Management
Arcisstraße 21
80333, Munich, Germany

Cover design: ediundsepp Gestaltungsgesellschaft mbH
Domagkstraße 1
80807, Munich,Germany

ISBN 978-3-00-045630-5

Table of Contents

Editorial

Thomas Fliedner, Rainer Kolisch and Anulark Naber

TUM School of Management
thomas.fliedner@tum.de, rainer.kolisch@tum.de, anulark.naber@tum.de

For the 14th International Conference on Project Management and Scheduling, which was held from March 30 until April 2, 2014 at Technische Universität München, Germany, we received 76 submissions, each of which was reviewed by two reviewers. Finally, 66 extended abstracts were accepted. The accepted extended abstracts are provided in this volume, in alphabetical order of the family name of the first author. At the end of the volume there is a list of all authors and a reference to the pages with their (co-)authored abstracts. Gratefully, we wish to acknowledge the following reviewers who have generously given their time and expertise in order to review all submitted abstracts thereby assisting in maintaining the high quality of the conference:

Allessandro Agnetis	Università di Siena (Italy)
Ali Allahverdi	Kuwait University (Kuwait)
Lucio Bianco	Università di Roma (Italy)
Jacek Blazewicz	Poznan University of Technology (Poland)
Jacques Carlier	Université de Technologie de Compiègne (France)
Erik Demeulemeester	Katholieke Universiteit Leuven (Belgium)
Thomas Fliedner	TUM School of Management (Germany)
Markus Frey	TUM School of Management (Germany)
Willy Herroelen	Katholieke Universiteit Leuven (Belgium)
Johann Hurink	University of Twente (Netherlands)
Joanna Jozefowska	Poznan University of Technology (Poland)
Sigrid Knust	Universität Osnabrück (Germany)
Rainer Kolisch	TUM School of Management (Germany)
Mikhail Kovalyov	National Academy of Sciences of Belarus (Belarus)
Wieslaw Kubiak	Memorial University (Canada)
Roel Leus	Katholieke Universiteit Leuven (Belgium)
Anulark Naber	TUM School of Management (Germany)
Linet Özdamar	Yeditepe University (Turkey)
Erwin Pesch	Universität Siegen (Germany)
Marie-Claude Portmann	Ecole des Mines de Nancy (France)
Christoph Schwindt	Technical University of Clausthal (Germany)
Avraham Shtub	Technion - Israel Institute of Technology (Israel)
Joseph Szmerekovsky	North Dakota State University (USA)
Vincent T'kindt	Université François Rabelais Tours (France)
Gündüz Ulusoy	Sabanci University (Turkey)
Vicente Valls	Universitat de Valencia (Spain)
Mario Vanhoucke	Universiteit Gent (Belgium)
Jan Weglarz	Poznan University of Technology (Poland)
Jürgen Zimmermann	Technical University of Clausthal (Germany)

Furthermore, we wish to thank Paul Sutterer and Tobias Stöhr for their assistance in compiling the proceedings.

Analysis of Activity Networks with Phase Type Distributions by Kronecker Algebra

Alessio Angius[1], András Horváth[1] and Marcello Urgo[2]

[1] University of Turin, Italy, angius@di.unito.it,horvath@di.unito.it
[2] Politecnico di Milano, Italy, marcello.urgo@mecc.polimi.it

Keywords: Activity Networks, Phase Type distributions.

1 Introduction

In the production of complex Manufacturing-to-Order products, uncertainty may stem from a number of possible sources, both internal and external, affecting the execution of the scheduled activities. A disrupted schedule incurs high costs due to missed due dates, resource idleness, or higher work-in-process inventory. Robust scheduling approaches aim at being able to provide a balanced compromise between expected performance and the protection against rare but extremely unfavourable events. Tackling this problem entails the need of estimating the probability distribution associated with a scheduling objective function. One approach, described in (Urgo 2014), is to use phase type distributions to approximate generally distributed activity durations.

The aim of this work is to show that an efficient manner to deal with activity networks in which durations are described by phase type distributions is provided by Kronecker algebra. This algebra alleviates the state space explosion problem, which is typical when phase type distributions are used in models where several activities are executed in parallel, and opens a way to an efficient and modular analysis methodology. The proposed approach is applied to a simple test case demonstrating how a proper design of the PH approximation provides a tool to asses the completion time distribution with a good accuracy.

2 Markov Activity Networks

In Markov Activity Networks (MAN) (Kulkarni & Adlakha 1986), given that (i) the durations of the activities are mutually independent and (ii) exponentially distributed, the execution of the activity network can be represented through a *continuous time Markov chain* (CTMC). Modelling the execution of a network of activities with a Markov chain provides the capability of exploiting the wide set of tools and approaches available for Markov models to address stochastic scheduling problems. As an example we consider the activity network in Figure 1, representing the execution of a set of activities $1, 2, ..., 5$ and their precedence relations. The durations of the activities are exponentially distributed with parameters $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, and λ_5,, respectively.

The CTMC describing the execution of the set of activities is depicted in Figure 2 where the on-going activities of a state are indicated inside the circle. The initial state is the state in which no activities have yet been executed. Assuming that this state is the first state of the CTMC, the initial probability vector is given as $\pi(0) = |1, 0, ..., 0|$. Denoting by Q the infinitesimal generator of the CTMC, the transient probabilities of the states at time t is given by the vector $\pi(t) = \pi(0) \exp(tQ)$ where $\exp(\bullet)$ is the matrix exponential function. Assuming that the last state of the CTMC corresponds to the situation when all activities are executed, the last entry of $\pi(t)$ gives the probability that the execution of all activities took less than t time units. I.e., the last entry of $\pi(t)$ provides the cumulative distribution function of the makespan of the underlying stochastic scheduling problem. Note that the

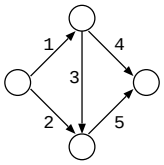


Fig. 1. An activity network.

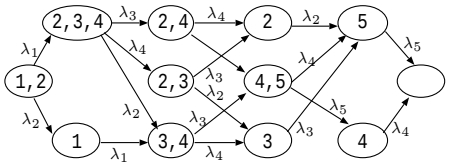


Fig. 2. CTMC of the activity network in Figure 1.

last state of the Markov chain is absorbing and the chain is acyclic. However, the restriction to exponentially distributed activity durations represents a limiting hypothesis, since the exponential distribution is quite seldom applicable to real industrial processes.

3 Phase Type Distributions

In the field of Markov models, phase type (PH) distributions are widely used to provide an approximation of a general distribution. Basically, a set of inter-related exponential delays are put together to form a distribution to approximate a general one. Formally, a continuous time PH distribution is the distribution of the time to absorption of a CTMC and the order of the PH distribution is given by the number of transient states of the chain. Consequently, the PH distribution is determined by a vector, β, which gives the initial probabilities of the transient states and a matrix, T, which contains the intensities of the transitions among the transient state. The cumulative distribution function, the probability density function and the moments of a PH distribution are given by

$$F(x) = P\{X \leq x\} = 1 - \beta e^{Tx}\mathbb{1}, \quad f(x) = \beta e^{Tx}(-T)\mathbb{1}, \quad m_i = i!\beta(-T)^{-i}\mathbb{1}$$

where $\mathbb{1}$ is a column vector of ones.

The use of PH distributions is popular because these distributions can be easily used as building blocks of more complex models. Indeed, if we are given a system in which all sojourn times are according to PH distributions and the next state distribution is Markovian then the overall system behavior can be described by a Markov chain.

The class of PH distributions is dense in the field of positive valued distributions, i.e., any positive valued distribution can be approximated by PH distributions with any accuracy. This fact does not provide however directly a practical method to fit distributions by PH distributions. Several authors proposed fitting methods and most of these fall into two categories: maximum likelihood (ML) based estimation of the parameters and moment matching techniques. One of the first works on ML estimation considered acyclic PH distributions (Bobbio & Cumani 1992) while an approach for the whole family, based on the expectation-maximization method, is proposed in (Asmussen, Nerman & Olsson 1996). Since these early papers, many new methods and improvements have been suggested for the whole PH family and for its sub-classes. For what concerns moment matching methods the following results are available. For low order (≤ 3) PH distributions, moment bounds and moment matching formulas are either known in an explicit manner or there exist iterative numerical methods to check if given moments can be captured (Telek & Heindl 2002, Horváth & Telek 2007). For higher order there exist matching algorithms, but these often result in improper density functions and the validity check is a non-trivial problem. In (Bobbio, Horváth & Telek 2005) a simple method is provided that constructs a minimal order acyclic PH distribution given three moments. Tool support is available for the construction of PH and ME distributions. Specifically, ML based fitting is implemented in PhFit (Horváth & Telek 2002) and a set of moment matching functions is provided in BuTools.

4 Analysis of Activity Networks with Phase Type Distributed Activity Durations by Kronecker Algebra

We assume now that the time to carry out an activity is distributed according to a PH distribution. We will denote the vector-matrix pair that describes the PH distributions associated with activity i by (β_i, T_i). Then infinitesimal generator of the overall system can be constructed by blocks. The diagonal blocks describe the parallel execution of a number of activities and the diagonal block associated with state j is given by

$$Q_{j,j} = \bigoplus_{i \in \mathcal{A}(j)} T_i$$

where $\bigoplus$ denotes the Kronecker sum operator and $\mathcal{A}(j)$ is the set of on-going activities in state j. An off diagonal block describes the finishing of an activity, the initialization of one or more activities and must maintain the phase of those activities that remain active. Accordingly, the block that describes the transition from state j to state k (with $j \neq k$) is given as

$$Q_{j,k} = \bigotimes_{i \in \mathcal{A}} R_i \quad \text{with} \quad R_i = \begin{cases} (-T_i)\mathbb{1} & \text{if } i \in \mathcal{A}(j) \text{ and } i \notin \mathcal{A}(k) \\ \beta_i & \text{if } i \notin \mathcal{A}(j) \text{ and } i \in \mathcal{A}(k) \\ I_i & \text{if } i \in \mathcal{A}(j) \text{ and } i \in \mathcal{A}(k) \\ 1 & \text{if } i \notin \mathcal{A}(j) \text{ and } i \notin \mathcal{A}(k) \end{cases}$$

where $\bigotimes$ denotes the Kronecker product operator and I_i is an identity matrix whose size is equal to the order of the PH distribution associated with activity i. The four cases in the above equation correspond to the cases: activity i finishes (described by the vector $(-T_i)\mathbb{1}$ which contains the "finishing" intensities of the associated PH distribution), activity i starts (described by β_i), activity i remain active (described by I_i), activity i is neither active in state j nor in state k (scalar 1 in the Kronecker product). The initial probability vector of the overall system is composed of the initial probabilities of the starting activities: $\pi(0) = |\bigotimes_{i \in \mathcal{A}(1)} \beta_i, 0, ..., 0|$.

The infinitesimal generator for the model depicted in Figure 1 is as follows

$$Q = \begin{vmatrix}
T_1 \oplus T_2 & t_1 \otimes I_2 \otimes \beta_3 \otimes \beta_4 & 0 & 0 & I_1 \otimes t_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & T_2 \oplus T_3 \oplus T_4 & I_2 \otimes I_3 \otimes t_4 & 0 & 0 & I_2 \otimes t_3 \otimes I_4 & t_2 \otimes I_3 \otimes I_4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & T_2 \oplus T_3 & I_2 \otimes t_3 & 0 & 0 & 0 & t_2 \otimes I_3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & T_2 & 0 & 0 & 0 & 0 & t_2 \otimes \beta_5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & T_1 & 0 & t_1 \otimes \beta_3 \otimes \beta_4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & I_2 \otimes t_4 & 0 & T_2 \oplus T_4 & 0 & 0 & 0 & t_2 \otimes I_4 \otimes \beta_5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & T_3 \oplus T_4 & I_3 \otimes t_4 & 0 & t_3 \otimes I_4 \otimes \beta_5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & T_3 & t_3 \otimes \beta_5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & T_5 & 0 & 0 & t_5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_4 \otimes I_5 & T_4 \oplus T_5 & I_4 \otimes t_5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & T_4 & t_4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{vmatrix}$$

where t_i is the finishing vector of the PH distribution (β_i, T_i) (i.e.,$t_x = -T_x\mathbb{1}$). Each block of rows corresponds to a set of currently ongoing activities and these sets are ordered as $\{1,2\}, \{2,3,4\}, \{2\ ,3\}, \{2\}, \{1\}, \{2,4\}, \{3,4\}, \{3\}, \{5\}, \{4,5\}, \{4\}$ and the empty set, $\emptyset$.

As a numerical example, we assume that activity 1 follows a log-normal distribution with mean equals to one and with pdf $f(x) = 1/(b\sqrt{2\pi}x)\ e^{-\frac{(\log(x)-a)^2}{2b^2}}$ and parameters $a = -1.62, b = 1.8$ while the duration of the other activities are given by order 4 Erlang distribution with mean equal to one (note that Erlang distributions are in the PH family). We constructed four different fitting distributions for the log-normal distribution of activity 1. The first one is an order 8 ML estimation by PhFit (Horváth & Telek 2002). The second and the third one are order 8 ML estimations extended with 2 and 4 phases, respectively, to fit the tail (obtained with PhFit). The last one matches three moments of the log-normal distribution (this can be done with an order 2 PH distribution by (Bobbio et al. 2005)). The pdf of the log-normal and the fitting PH distributions are depicted in Figure 3. The pure ML estimation fails to catch the tail behaviour and the moment matching PH distribution fails to capture the shape of the log-normal distribution.

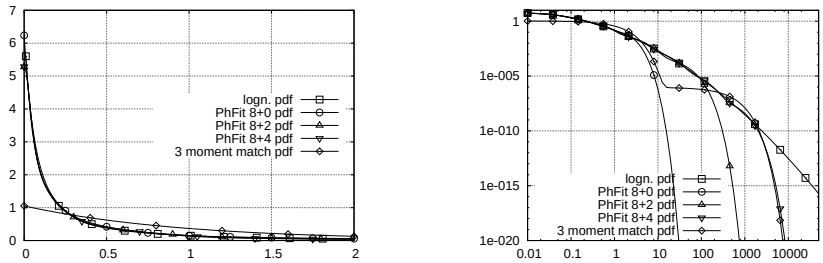


Fig. 3. Main part (left) and tail (right) of the pdf of the PH distributions fitting the log-normal distribution.

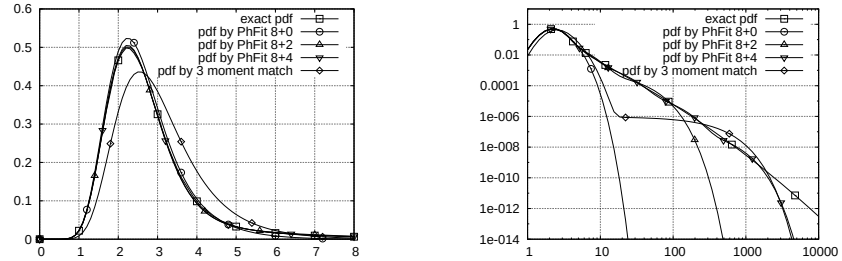


Fig. 4. Main part (left) and tail (right) of the pdf of the makespan of the activity network.

In Figure 4 we depicted the pdf of the makespan of the activity network both with the original log-normal distribution and with the approximating PH distributions. It can be seen that the best approximation is achieved by applying the third PH distributions that captures both the main part and the tail of the pdf.

Acknowledgments. This research has been partially funded by the EU FP7 Project VISIONAIR - Vision and Advanced Infrastructure for Research, Grant no. 262044.

References

Asmussen, S., Nerman, O. & Olsson, M. (1996), 'Fitting phase-type distributions via the EM algorithm', *Scandinavian Journal of Statistics* **23**(4), 419–441.

Bobbio, A. & Cumani, A. (1992), 'Ml estimation of the parameters of a ph distri- butions in triangular canonical form', *Computer Performance Evaluation* pp. 33–46.

Bobbio, A., Horváth, A. & Telek, M. (2005), 'Matching three moments with minimal acyclic phase type distributions', *Stochastic Models* **21**, 303–326.

Horváth, A. & Telek, M. (2002), Phfit: A general phase-type fitting tool, *in* 'Proc. of 12th Performance TOOLS', Vol. 2324 of *Lecture Notes in Computer Science*, Imperial College, London.

Horváth, G. & Telek, M. (2007), '"a canonical representation of order 3 phase type distributions', *Lecture Notes in Theoretical Computer Science* **4748**, 48–62.

Kulkarni, V. & Adlakha, V. (1986), 'Markov and Markov-regenerative pert networks', *Operations Research* **34**(5), 769–781.

Telek, M. & Heindl, A. (2002), 'Matching moments for acyclic discrete and continuous phase-type distributions of second order', *Int. J. of Simulation Systems, Science & Technology* **3**(3-4).

Urgo, M. (2014), *Sequencing and Scheduling with Inaccurate Data*, Nova Publisher, chapter Stochastic Scheduling with General Distributed Activity Durations Using Markov Activity Networks and Phase-Type Distributions.

Choosing and scheduling patients for surgery: multi-objective optimisation

Ballestín F[1], Lino P[1], Pérez A[1], Quintanilla S[1] and Valls V[2]

[1] Dept. de Matemáticas para la Economía y la Empresa, Universidad de Valencia
e-mail: francisco.ballestin@uv.es, angeles.perez@uv.es, maria.quintanilla@uv.es, pilar.lino@uv.es

[2]Universidad de Valencia
e-mail: Vicente.valls@uv.es

Keywords: scheduling, health care, operating room, multi-objective

1. Problem statement

Health care is a major problem, both in our society and in operations research. Hundreds of papers have been published using optimisation to solve health services problems. Our area of interest lies in the scheduling of operating rooms, see e.g. Cardoen et al. (2010) or Guerriero and Guido (2011) for a review of the planning and scheduling of operating rooms. We are working with a hospital in Spain which serves an average of 4,000 patients daily, has 1,000 single beds, 45 operating rooms and 6,300 employees. The management of the hospital, which has been built recently, is trying to mechanize some of the processes of the hospital, using in some cases new technology available in the hospital. Our work is focused on the Urology unit, which is among the top performing units regarding the measures imposed by the local government. The process that concerns us is the planning of the operations to be carried out in a planning period (established as two weeks), specifically the selection of patients on the waiting list who have to be operated on, and the operating room and start time assigned to those operations. Currently, this task is done by hand by the head doctor of the unit.

The main elements to be considered in the optimisation problem are the waiting list, the operating room sessions assigned to the unit, the unit resources and the performance criteria. The medical unit has a certain number of operating room sessions assigned. A session is determined specifying the date, the room number of the operating room and the start and finish time of that session. The waiting list is a dynamic list of patients who have to be operated on. Although this list is continuously updated, we work with the list of those patients included ten days before the first day of the two-week plan, because it is at that moment when the plan is designed. Later on we will discuss the information we have about these patients. The resources refer to doctors and specialties. In addition, an operation may belong to a specialty of the unit, or it may be classified as "general". Each session can have one or more specialties assigned (not always the case in papers from the literature). A specialised operation must be done in a session assigned to the corresponding specialty, whereas a general operation can be done in any session. Regarding the patients, we only consider elective patients in this problem. An elective patient is someone for whom the surgery is planned well in advance. The opposite concept would be non-elective or urgent patients, those who need an operation urgently. We work both with inpatients and outpatients. An outpatient enters and leaves the hospital on the same day, whereas inpatients must stay at least one night in the hospital. For each patient we know the following information: the date of admission, whether they are inpatients or outpatients and their priority. The priority is one, two or three, depending on how urgent the operation is. Patients with priority one, two or three should be operated on in less than one, three and twelve months respectively. These values, together with the admission date, result in a (soft) due date for the operation. The waiting list also indicates if the patient needs a specialized surgery or not and the estimated duration of the operation. We are going to deal with durations as constants, not as random variables. The hospital also plans inpatients and outpatients separately and therefore we deal with those patients in the same way. Furthermore, between operations there should be a waiting time for cleaning fixed at twenty minutes. Regarding the objective functions, the hospital is interested in two performance criteria, to schedule first patients with smaller due dates (waiting time, WT) and the maximisation of the utilisation of the surgery rooms (UR). The waiting time can be calculated in several ways. In the model we will explain the specific measure we have adopted, called waiting time indicator (WTI).

The utilisation rate is the rate between the utilisation of the operating rooms and their capacity. The capacity is obviously the sum of the durations of all the sessions. The utilisation is the sum of the duration of the operations assigned to all the sessions.

We have modelled the problem and solved it with GAMS. The mathematical model is similar to others in the literature, see e.g. Agnetis et. al (2012).

Notation

I is the set of surgeries (indexed by i) on the current waiting list; P_i is the duration of the i-th surgery; dd_i is the due date of the i-th surgery; J is the set of sessions, indexed by j; O_j is the length time of session j; c_{ij} = 1 if the surgery i can be assigned to session j because the specialty of i is assigned to session j and 0 otherwise; K is a constant greater than the maximum of the deadlines for the surgeries; t(j) is the day in the planning of session j (we plan two weeks, a session in the first day has t(j)=1, a session in the second t(j)=2, etc.); T is the last day of the planning period, in general 14 (two weeks); ct is the cleaning time after an operation.

Variables

x_{ij} = 1 if the i-th surgery is assigned to session j and 0 otherwise.

Model

$$Max\ WTI = \sum_i \sum_j (K - dd_i)\big(T + 1 - t(j)\big) \cdot x_{ij} \quad (1)$$

$$Max\ UR = \frac{\sum_j \sum_i P_i x_{ij}}{\sum_j O_j} \quad (2)$$

s.t.

$$\sum_j x_{ij} \leq 1 \quad \forall i \quad (3)$$

$$\sum_i (P_i + ct) x_{ij} \leq O_j + ct \quad \forall j \quad (4)$$

$$x_{ij} \leq c_{ij} \quad \forall i, j \quad (5)$$

$$x_{ij} \in \{0, 1\} \ \forall i, j$$

Equation (1) tries to include in the schedule those surgeries closer to their due date. The multiplication $(K - dd_{is})(T+1-t(j))$, together with the maximisation gives priority to those patients with smaller due dates, since $(K - dd_{is})$ is bigger. We want to schedule those patients as soon as possible, and (T+1-t(j)) is bigger the sooner the session is. Equation (2) calculates the average utilization. Constraints (3) ensure that each surgery is performed no more than once. Constraints (4) take into account the length of a session. The duration of all operations assigned to a session cannot exceed its length. Constraints (5) guarantee that a surgery is assigned to a session with the appropriate discipline.

2. Multi-objective optimisation

For decades, researchers studied how to optimise a single objective function in optimisation problems, as this task was more than difficult enough. However, in practice, several goals compete with each other; practitioners want to optimise several measures at the same time. We are talking then about multi-objective or multi-criteria optimisation, see e.g. Ehrgott and Gandibleux (2004)

or Zitzler (1999). There are several ways to cope with these kinds of problems. The approach used depends on the relation between the decision maker (DM) and the optimisation process. According to Hwang and Masud (1979) and Horn (1997), there are three categories.

(1) Decision making before search: Information given by the DM leads to a single objective optimisation problem where the objective is often a linear function of the original objectives of the multi-objective problem (MOP).

(2) Search before decision making: A set of candidate solutions is calculated and then the DM selects a solution among them.

(3) Decision making during search: The optimisation is divided into steps, after each of which a number of alternative trade-offs are presented to the DM, whose information guides the next step.

In our case, the head doctor prefers to be able to choose among several "top" solutions regarding the two objective functions. Therefore, we are going to follow approach two and calculate several efficient solutions. A feasible solution x is called efficient if there is no other feasible solution y as good as x in all objective functions but one, and strictly better than x in at least one objective function. If x is efficient, then f(x) is called non-dominated. The set of all non-dominated solutions is also called the Pareto set or Pareto front. In general, a multi-objective problem is considered to be solved if an efficient solution for each element of the Pareto front is given. In our case, we do not seek to find the whole Pareto front, but we look for several efficient solutions to present to the head doctor. We will see in the next section how we build those solutions.

3. Computational Results

We have worked with a real instance, a two week period in which we know all the real data. The two weeks (weekends do not count) go from 11/03/2013 till 22/03/2013, even though there are two holidays, 18/03 and 19/03. There are 12 sessions and 10 doctors. There are 172 patients on the waiting list, of which approximately 40 are scheduled in a solution. In the future, we plan to create an instance generator capable of producing instances with the same structure as the real instances. We plan to test our procedures in those instances, too.

Figure 1 shows a typical output of our algorithm, which is given to the head doctor. We can see the day of the week (11/03/2013), the name of the day and the time, morning/afternoon ("lunes mañana") and the name of the operating room ("quirófano Q15"). There might be several operating rooms per session. After the name we have the number of operations occurring in that session. Immediately after we have a line for each operation with the following information: 1) identification number of the patient – names are not allowed because of confidentiality, 2) interval time where the patient occupies the operating room [start time of the operation, end time of operation, end time of cleaning], 3) priority on the list, 4) number of days on the list (column en_lista), 5) surgical discipline, 6) doctor that diagnosed the patient, 7) illness and 8) procedure that is going to be applied.

```
11/ 3/2013
LUNES MAÑANA
QUIRÓFANO Q15:3 operaciones
 ENFERMO [inicio, fin,limpio] prior en_lista   sección_unidad    MÉDICO                    DIAGNÓSTICO                  PROCEDIMIENTO
 6078390 [ 8: 0, 9:15, 9:35]    1    47             TODOS_   noconsta              CARCINOMA_DE_VEGIGA          RTU_DE_TUMOR_VESICAL
 1709185 [ 9:35,12:25,12:45]   2A    41      Onco_Laparoscopia   noconsta  ADENOCARCINOMA_MICROACINAR_PROSTATA       LINFADENECTOMIA_ PELVICA
 1152939 [12:45,14: 0,14:20]    2    43      Onco_Laparoscopia   noconsta                TUMOR_VESICAL                  RTU_VEJIGA

QUIRÓFANO Q40:4 operaciones
 ENFERMO [inicio, fin,limpio] prior en_lista   sección_unidad    MÉDICO                    DIAGNÓSTICO                  PROCEDIMIENTO
 6319976 [ 8: 0, 9:15, 9:35]    3   254             TODOS_   noconsta              HIDROCELE_IZQUIERDO          HIDROCELECTOMIA (UCSI)
 6378122 [ 9:35,10:25,10:45]    2   124             TODOS_   noconsta        UROPATIA_OBSTRUCTIVA_BILATERAL     COLOCACION_DE_CATETER_DOB (UCSI)
 1599851 [10:45,12:25,12:45]    3   176      TVI_Laparoscopia   noconsta                      HBP          FOTOVAPORIZACION_PROSTATI (UCSI)
 1449123 [12:45,14:25,14:45]    3   160           TVI_Laser   noconsta          HBP-HIPERACTIVIDAD_DETRUSOR      FOTOVAPORIZACION_PROSTATA


12/ 3/2013
MARTES MAÑANA
QUIRÓFANO Q15:2 operaciones
 ENFERMO [inicio, fin,limpio] prior en_lista   sección_unidad    MÉDICO                    DIAGNÓSTICO                  PROCEDIMIENTO
 1550226 [ 8: 0,10:50,11:10]   2A    42        Onco_Abierta   noconsta                 LITIASIS_RENAL       NEFRECTOMIA_PARCIAL_IZQUI
 6133643 [11:10,14: 0,14:20]   2A    35        Onco_Abierta   noconsta                CARCINOMA_RENAL       NEFRECTOMIA_RADICAL_IZQUI


13/ 3/2013
MIÉRCOLES MAÑANA
QUIRÓFANO Q15:2 operaciones
 ENFERMO [inicio, fin,limpio] prior en_lista   sección_unidad    MÉDICO                    DIAGNÓSTICO                  PROCEDIMIENTO
 1547446 [ 8: 0,10:50,11:10]   2A    34      Onco_Laparoscopia   noconsta                   MASA_RENAL      NEFRECTOMIA_PARCIAL_DEREC
 6449984 [11:10,14: 0,14:20]   2A    33      Onco_Laparoscopia   noconsta          TUMOR_RENAL_IZQUIERDO          NEFRECTOMIA_RADICAL
TARDE
QUIRÓFANO Q15:2 operaciones
 ENFERMO [inicio, fin,limpio] prior en_lista   sección_unidad    MÉDICO                    DIAGNÓSTICO                  PROCEDIMIENTO
 2818151 [15: 0,17:50,18:10]   2A    29      Onco_Laparoscopia   noconsta          TUMOR_RENAL_BILATERAL               NEFRECTOMIA
 6102812 [18:10,18:60,19:20]    2    69             TODOS_   noconsta        TUMOR_VESICAL_PRIMARIO                 TM_VESICAL
```

Figure 1. Output of the algorithm for the real instance

To create the candidate solutions, we apply the exact method with the objective function of the waiting time indicator (WTI), obtaining the best (minimum) possible value of WTI, WTI_B, and a value of UR. Then, we exactly solve the problem, maximising the UR, with an additional constraint of WTI ≥ WTI_B. The outcome of this problem is an efficient solution, with the best UR possible, UR_0, for the WTI_B. Then, we solve a problem maximising UR, without any additional restriction, obtaining UR_B, the best (maximum) possible value of UR. In general, $UR_0 < UR_B$. We calculate nsol values equidistantly in the interval $[UR_0 , UR_B]$, $UR_0 < UR_1 < UR_2 < ... < UR_{nsol} < UR_B$. Afterwards we exactly solve the problems maximising WTI with an additional restriction, $UR \geq UR_i$. With these problems we obtain nsol possible efficient solutions. From these nsol+2 solutions we extract the efficient solutions which we present to the decision maker.

Figure 2 shows the Pareto front obtained in the example. The x-axis is the UR, which goes from 81% to 89.4%, while the y-axis represents the WTI, and goes from 55000 to 80600.

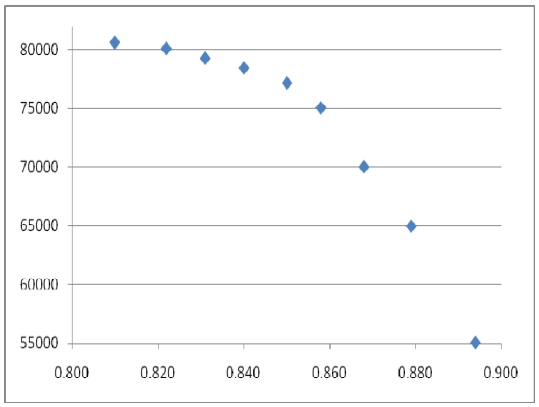


Figure 2. Pareto Front for the real instance

Acknowledgements

This research was partially supported by Ministerio de Ciencia e Innovación, MTM2011-23546.

References

Agnetis A., A. Coppi, M. Corsini, G. Dellino, C. Meloni and M. Pranzo, 2012, "Long term evaluation of operating theater planning policies", Operations Research for Health Care Vol. 1(4), pp. 95-104.

Cardoen, B., E. Demeulemeester and J. Belien, 2010. "Operating room planning and scheduling: A literature review". Eur. J. Operational Res., 201: 921-932.

Ehrgott M, Gandibleux X. "Approximative solution methods for multiobjective combinatorial optimization". TOP 2004; 12(1): 1-88.

F. Guerriero, R. Guido, "Operational research in the management of the operating theatre: a survey", Health Care Management Science 14 (1) (2011), 89–114.

Horn J (1997) Multicriteria decision making. In: Back T, Fogel DB, Michalewicz Z, editors. Handbook of evolutionary computation. 97/1, F1.9, IOP Publishing Ltd. and Oxford University Press

Hwang CL, Masud AS (1979) "Multi-objective decision making, methods and applications. A state of the art survey". Lecture notes in economics and mathematical systems, Springer-Verlag, Berlin

Zitzler E. "Evolutionary algorithms for multiobjective optimization: Methods and applications". Ph.D. thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.

Lean and efficient project management: new concepts and tools

A.J. Baptista[1], Inácio Ribeiro[3], J.P. Pereira[1], A. Sousa Ribeiro[2], João Bastos [3]

[1]INEGI, Porto
e-mail: abaptista, jptp@inegi.up.pt

[2]SISTRADE, Porto
e-mail: sousa.ribeiro@sistrade.com
[3]ISEP, Porto
e-mail: jab@isep.ipp.pt

Keywords: Project Management, Lean Principles, Efficiency Assessment, Effectiveness Management, Efficiency Management.

1. Introduction

In the last decades new approaches to project management have been developed in order to deal simultaneously with: short delivery times, high quality of deliverables, increased difficulty on objectives preposition or milestones scheduling, and budgetary constraints. In innovation driven projects and environments the management tasks are even more complex, since that the intrinsic uncertainty of R&D activities can easily put at risk both delivery times and budget limits.

Meanwhile, "Lean" practices and principles have shown their advantage and applicability in this area and especially in new product development projects. Several literature works can be cited, much of it related to the so called "Toyota Way" (Liker et. al. 2006 and Letens et. al. 2011). However, when the subject is directly related to lean project management, there are not many approaches that have addressed the topic and typically they are more related to construction field, such as Ballard et. al. (2003), or the Large Scale Industrial project as Karim and Nekoufar (2012) In the field of R&D projects Sánchez et. al. (2002) presented a work were the project efficiency was assessed in Spanish manufacturing companies.

Project management workflows are in most of the cases information based and thus not all-time visible and often elusive. Project specifications and variability hinders the process of standardization and reuse of work/product previously executed and wastes are hard to visualize and account for. Project efficiency is not an easy measurable concept, especially in innovation driven projects, since the measure of R&D efficiency includes not only information about the output and outcomes, such as patents, new products and profits, but also about the processes leading to them (Sánchez et. al. 2002).

Swink et. al. (2006) develop a theory of efficiency and performance trade-offs for new product development (NPD) projects, were they have highlighted the trade-offs: speed-quality; time–cost; and time-quality. Amongst the findings the authors point the importance of project management experience, balanced management commitment, and cross-functional integration to achieving high levels of NPD project efficiency.

Bouras (2013) developed a method for the evaluation of project management efficiency in the case of industrial projects execution. The method combines the objectives of cost and time completion, as well as other success criteria related to the operation and maintenance of the unit, and subsequently the relevant earnings into a so called unique relation using only non-dimensional quantities. Despite the well suited solution presented to relate the key performance indicators (KPI) with adimensional quantities for the efficient management of a given project, the method lacks a more integrated view of the different KPI.

2. New concepts and tools for lean and efficient project management

It is not an easy task to combine and optimize simultaneously the flow of a process (time management to reduce the total lead time) and the related resources. The so called Multi-Layer Stream Mapping (MSM) concept address the difficulties encountered on the efficiency assessment of complex systems, by analysing simultaneously the flow of a process and its resources Lourenço et. al. (2013). The root for the method can be related with the well-known Value Stream Mapping tool from the lean methodologies. Nonetheless, the concept opens a different and higher level

perspective to the value versus waste assessment for a system, by adding a multi layered variable analysis of different nature (time, energy, mass, water, etc.), with the combination of dimensionless ratios, which allows the comparison of efficiency ratios (Figure 1).

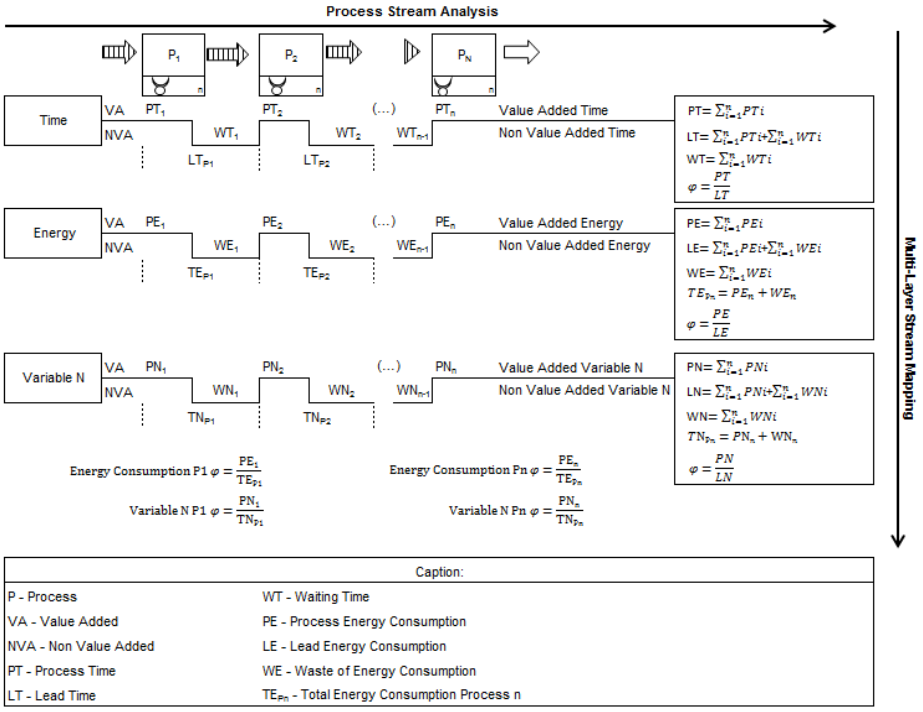


Figure 1: MSM global efficiency concept basis.

The values located below of the MSM line are those which do not add value to the product or deliverable, i.e. representing the "waste/misuses" of time (Lead Stages), or any kind of resources. On the other hand, the values that are presented above the VSM line (Process Stages) are those that add value to the product, thus representing the “useful consumption” of stream flow that can be analyzed in order to assess, evaluate and quantify production efficiency. One key feature of the MSM approach consists in considering dimensionless ratios, so the higher the result for the ratio, the better the performance of the energy, mass or time flow or other key variable of a process or system. It is possible to quantify the aggregated efficiency performance of a given unit processes (P1, P2, PN), by following the Multi-Layer Stream Mapping direction P1 φ (column direction of the diagram) or the overall efficiency of a given variable by analyzing each row direction. Finally, the overall efficiency of a process sequence or system can be evaluated by calculating the average (or other weighted formula) of the several φ of the unit processes.

In this work, the concept of MSM is applied in the attempt of solving the issues of combining the common trade-offs of project management, creating both an individual or integrated view of the KPI of a project through the project stages or its processes. Since the MSM method relies on evaluating systematically if one variable of a given process acts towards the creation of value, or not (and should be classified as waste), and it delivers lean principles base attitude for the project management domain, as well as, an efficiency measurement. The application of the MSM method to the project management field brings new challenges, as for the measurement of the valuable part of some variables. On the other hand, the definition of the aggregated efficiency of a given project considers not only pure efficiency KPI, where both the accomplishment results and the inherent efforts are measured, but also effectiveness KPI measurement. The aggregated efficiency complies in this way with the combination of efficiency and effectiveness assessment. In Table 1 are presented a compiled a group of representative variables and associated KPI for project

management evaluated by MSM method. A specific constraint is related with the KPI definition, which it should be a value between [0-100%].

Table 1 – Compilation of a group of variables and project management KPI to be measured with MSM.

Variable	KPI	Observations
Availability of human resources	Actual resources allocated / Planned resources	This KPI can be expressed in Hours and should be limited to 100% if Actual > Planned
Tasks completed on time	nº Tasks completed on time / nº of tasks	
Tasks accomplished under the budget time effort	Tasks accomplished under the budget time effort / nº of tasks	Direct measure for effort control (and indirectly of costs)
Milestones attained on time	nº Milestones attained on time / nº of milestones	This can indirectly evaluate the lead durations of stages and schedule accomplished
Deliverables completed on time	nº Deliverables completed on time / nº of deliverables	This can indirectly evaluate the lead durations of stages and schedule accomplished
Scope - Objectives	Fraction of Objectives attained / Total amount of Objectives	The objectives should be fractioned per task and each fraction
Quality of supply	Average quality evaluation of Deliverables / Max punctuation value	Should be applied an inquire to the client with a [0-5] scale
Client Satisfaction	Average client evaluation / Max punctuation value	Should be applied an inquire to the client with a [0-5] scale
Team Morale or Stamina	Average team member evaluation / Max punctuation value	Should be applied an inquire to team members with a [0-5] scale

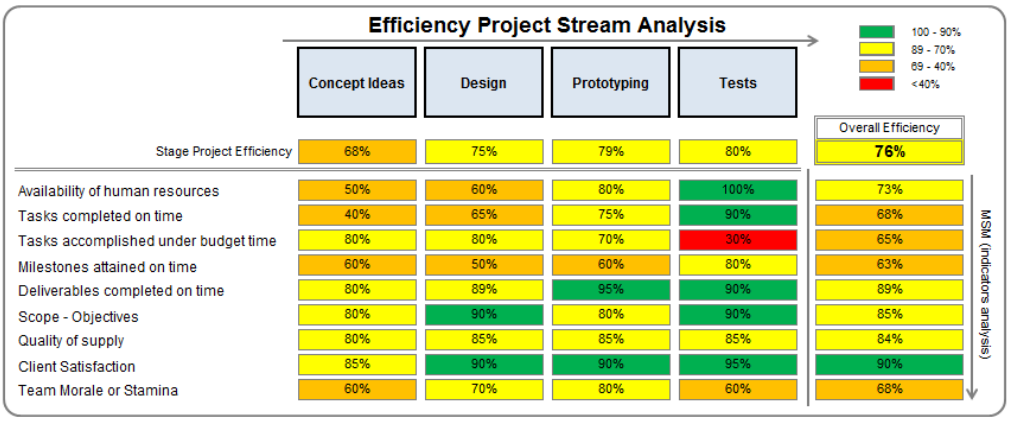
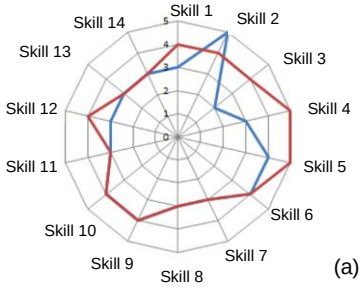


Figure 2. Example of efficiency map or dashboard of a product development project.

Besides the MSM concept application (example in Figure 2), the lean principles of the Toyota Way related to respect for the collaborator, namely to increase the collaborators intrinsic motivation and empowerment, were used in order to achieve positive commitment inside the project team. Figure 3a shows a radar chart example for skill assessment (blue line) and intrinsic motivation (red line), representing the output of the assessment carried out to know better the knowledge level (skill) of the collaborators in predetermined areas and in what areas they feel more motivated to participate. The method that was applied to map the team skills/motivation consisted in biannual inquires to the team (self-assessment with scale from 0-5) and the results were stored in a so called skill/competence matrix. Figure 3.b shows an extracted view example for one specific collaborator.



(a)

Type	Speciality	Skill	Intrinsic Motivation
Key Competence	CAD	3	2
Key Competence	CAE	3	4
Key Competence	CAM	3	3
Key Competence	Design	3	4
Key Competence	Materials	4	4
Key Competence	Prototyping	5	4

(b)

Figure 3 – Radar chart example (a) for skill assessment (blue line) and intrinsic motivation (red line) and table data example for one team member (b).

3. Conclusions

The described methods and tools were developed in order to assess the overall efficiency of a project, or group of projects, based in lean principles and in the novel framework MSM –Multi-layer Stream Mapping. The approach is proving to be a very powerful tool towards an easy and intuitive interpretation of KPI metrics of aggregated efficiency in project management. Since the number of KPI is not restricted, and should be set accordingly to the management rules of each organization, it can also be of great usefulness to handle the common trade-offs that exist in project management so that the project manager can see the consequences of a decision that puts more emphasis on one aspect of management, possibly neglecting other aspects. The proposed approach facilitates ways to compare groups of projects, as for portfolio management or multi-project management. In this type of analysis the approach present by itself a new kind of simple dashboard evaluation that can be both used by top-management, middle management, and also by the collaborators.

The other methods and tools, related to team management such as the described skill/intrinsic motivation matrix and radar charts are proving to be, not only popular within the project teams, but are also effective and positive to improve team motivation, identifying skill needs for project execution, decision support for training events and even to discover hidden talents.

The described methods are being implemented in a new application by an IT software company (SISTRADE) in order to be available as an independent product that can instantiate the created tools and methods allowing a more straightforward use of it in project management daily activities.

Acknowledgements

This work was developed within the frame of the Project PRODUTECH PTI, nº 13851 of the Program of Incentives to Technological Research & Development, supported by the Portuguese Agency of Innovation, to whom the financial support is to be acknowledged.

References

Angel Martínez Sánchez, Manuela Pérez Pérez. (2002). R&D project efficiency management in the Spanish industry. International Journal of Project Management. 20(7), 545–560.

Azharul Karim and Saviz Nekoufar (2012). "Standardisation of steel plant building in a portfolio of projects." Steel Times International 36(3): 42-48.

E. J. Lourenço, A. J. Baptista, J. P. Pereira, Celia Dias-Ferreira. (2013). Multi-Layer Stream Mapping as a Combined Approach for Industrial Processes Eco-efficiency Assessment. Re-engineering Manufacturing for Sustainability. 427-433

Geert Letens, Jennifer A. Farris, Eileen M. Van Aken. (2011). A Multilevel Framework for Lean Product Development System Design. Engineering Management Journal. 23(1), March 2011. 69-85.

Glenn Ballard, Gregory A. Howell. (2003). Lean project management. Building Research & Information (2003) 31(2), 119–133.

Jeffrey K. Liker and James M. Morgan. (2006). The Toyota Way in Services: The Case of Lean Product Development. Academy of Management Perspectives, 20 (2), 5-20.

Morgan Swink, Srinivas Talluri, Temyos Pandejpong. (2006). Faster, better, cheaper: A study of NPD project efficiency and performance tradeoffs. Journal of Operations Management. 24. 542–562

Vassilis K. Bouras. (2013). A Method for the Evaluation of Project Management Efficiency in the Case of Industrial Projects Execution. Procedia - Social and Behavioral Sciences. 74, 285–294.

Using real-life baseline schedules for evaluating project control techniques

J. Batselier[1] and M. Vanhoucke[2]

[1]Ghent University, Belgium
e-mail: jordy.batselier@ugent.be

[2]Ghent University and Vlerick Business School, Belgium
e-mail: mario.vanhoucke@ugent.be

Keywords: baseline schedule, earned value management, project control, real-life project database.

1. Introduction

The baseline schedule plays a crucial role in the managing of a project, as it provides the point of reference for both risk management and project control (Vanhoucke 2012). A project's baseline schedule is primarily defined by the project network. Therefore, it is essential to obtain realistic and diverse project networks that span the full range of problem complexity so that accurate evaluations of project management techniques can be performed. For this purpose, many network generators have been presented in literature (e.g. Kolisch et al. 1995, Schwindt 1995, Agrawal et al. 1996, Tavares 1999, Demeulemeester et al. 2003, Vanhoucke et al. 2008). The most complete network generators can take into account both network topology and resource-related characteristics. Moreover, these network generators have enabled the construction of several benchmark datasets (see e.g. Boctor 1993, Kolisch and Sprecher 1996, Van Peteghem and Vanhoucke 2013), which can be and have been employed as a basis for the validation and evaluation of novel project management techniques.

However, it can be argued that these techniques should not only be evaluated on generated project datasets, but also on a large and diverse database of projects with real-life baseline schedules. This view has already been expressed by many authors (i.a. Zwikael et al. 2000, Henderson 2003, 2004, Lipke 2009, Vanhoucke 2011). Therefore, the construction of a large and diverse project database lies at the heart of the current study. These data can then be used for various research topics. In this study, our focus is on the earned value management (EVM) control technique. More specifically, the accuracies of the most commonly used EVM forecasting methods for both project duration and cost are assessed.

2. Project database construction procedure

During the process of project database construction, it is crucial to ensure the quality and completeness of the real-life project data. Therefore, so-called project cards were developed, which provide a tool for project data evaluation, categorization and acquisition. Figure 1 illustrates how the project cards can be used during the database construction procedure. Moreover, the figure visualizes the classification into generated and real-life project data.

We mention that most of the acquired data were gathered from case studies performed by master students in Business Engineering and Civil Engineering of Ghent University. At the end of October 2013, the project database consisted of 47 real-life projects, a number that will be further increased by April 2014 when the 14th PMS conference takes place. Furthermore, it is our goal to continuously extend the project database so that ever more generalizable conclusions could be obtained for current as well as future research topics.

Moreover, within the framework of the project cards, novel data evaluation and project characterization metrics are proposed. In that way, the data can more accurately be evaluated and categorized, and gaps in the database regarding completeness and diversity can more easily be identified. A project card also contains information about, for example, whether the project's baseline schedule includes cost and resource data and, if present, how many of which kind of resources (renewable or non-renewable) there are. Furthermore, the network topology of each project is described based on various indicators (Tavares et al. 1999), like e.g. the serial/parallel-indicator (SP). Of course, the different aspects of risk analysis, mainly concerning project

sensitivity measures (Williams 1992, Elmaghraby 2000), and project control are also presented on a project card.

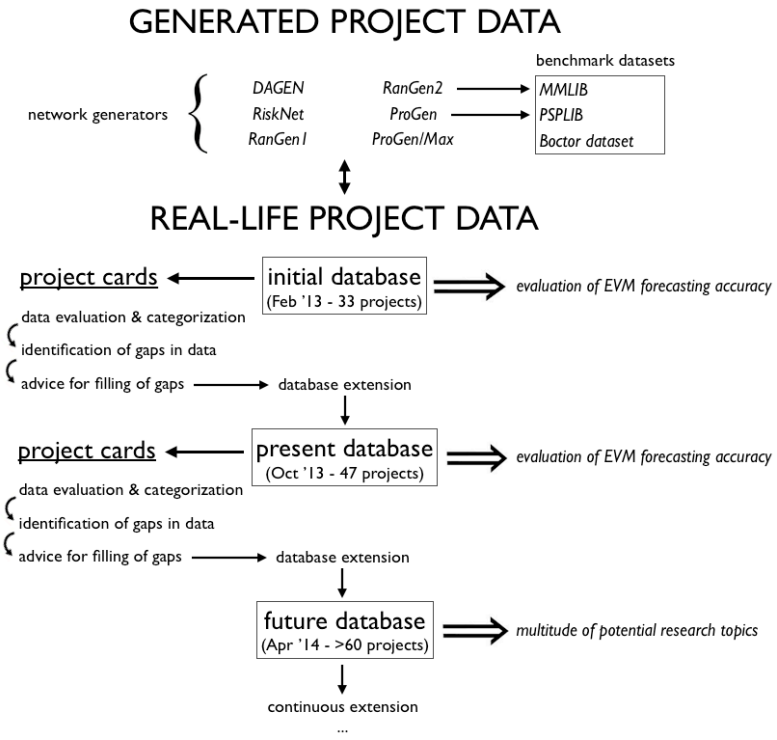


Figure 1. Situation and overview of our research approach based on real-life project data

3. Research design

The current study based on the real-life database concerns the evaluation of the accuracy of the most commonly used EVM forecasting methods for both project duration and cost. More specifically, the nine time and eight cost forecasting methods presented in Vanhoucke (2012) are evaluated. The performed study builds upon the work of many authors who have assessed either the cost (Bright and Howard 1981, Covach et al. 1981, Riedel and Chance 1989, Zwikael et al. 2000) or time (Henderson 2003, 2005, Vandevoorde and Vanhoucke 2006, Hecht 2007, Lipke 2009, Rujirayanyong 2009) forecasting accuracy of a selection of EVM methods.

However, whereas previous work focused on either the cost or the time dimension, this study considers both dimensions and therefore allows a quantitative comparison (based on the mean absolute percentage error or MAPE) between the two main aspects of project forecasting. Furthermore, the obtained results are based on a real-life project database that surpasses all existing real-life datasets in project management literature in both size and diversity, by which the recommendations of many authors in the research field are followed (i.a. Zwikael et al. 2000, Henderson 2003, 2004, Lipke 2009, Vanhoucke 2011).

Apart from the central theme of overall EVM forecasting accuracy evaluation, the influence of the network structure on the forecasting accuracy is also assessed. This topic extends the research of Vanhoucke and Vandevoorde (2007) as they performed a comparable analysis, however, based on a simulation study instead of real-life data.

Moreover, as real-life baseline schedule data can also be used as a basis for simulations, simulated and real forecasting results - the latter being based on actual instead of simulated project progress - can be compared. This comparison can of course not be made for generated projects, as such projects do not contain actual progress data and are therefore limited to simulated forecasting only.

References

Agrawal M.K., S.E. Elmaghraby and W.S. Herroelen, 1996, “DAGEN: A generator of testsets for project activity nets", European Journal of Operational Research, Vol. 90, pp. 376-382.

Boctor F., 1993, “Heuristics for scheduling projects with resource restrictions and several resource-duration modes", International Journal of Production Research, Vol. 31, pp. 2547-2558.

Bright H., T. Howard, 1981, “Weapon system cost control: Forecasting contract completion costs", Comptroller/Cost Analysis Division, US Army Missile Command, Redstone Arsenal, Alabama.

Covach J., J. Haydon and R. Reither, 1981, “A study to determine indicators and methods to compute estimate at completion (EAC)", ManTech International Corporation, Virginia.

Demeulemeester E., M. Vanhoucke and W. Herroelen, 2003, “RanGen: A random network generator for activity-on-the-node networks", Journal of Scheduling, Vol. 06, pp. 17-38.

Elmaghraby S., 2000, “On criticality and sensitivity in activity networks", European Journal of Operational Research, Vol. 127, pp. 220-238.

Hecht L., 2007, “Case study of earned schedule to do predictions", The Measurable News, Winter, pp. 16-18.

Henderson K., 2003, “Earned schedule: A breakthrough extension to earned value theory? A retrospective analysis of real project data", The Measurable News, Summer, pp. 13-17 & 21-23.

Henderson K., 2004, “Further developments in earned schedule", The Measurable News, Spring, pp. 15-17 & 20-22.

Henderson K., 2005, “Earned schedule in action", The Measurable News, Spring, pp. 23-28 & 30.

Kolisch R., A. Sprecher, 1996, “PSPLIB - a project scheduling problem library", European Journal of Operational Research, Vol. 96, pp. 205-216.

Kolisch R., A. Sprecher and A. Drexl, 1995, “Characterization and generation of a general class of resource-constrained project scheduling problems", Management Science, Vol. 41, pp. 1693-1703.

Lipke W., 2009, “Project duration forecasting... a comparison of earned value management methods to earned schedule", The Measurable News, Issue 2, pp. 24-31.

Riedel M., J. Chance, 1989, “Estimates at completion (EAC): A guide to their calculation and application for aircraft, avionics, and engine programs", Aeronautical Systems Division, Wright-Patterson AFB, Ohio.

Rujirayanyong T., 2009, “A comparison of three completion date predicting methods for construction projects", Journal of Research in Engineering and Technology, Vol. 6, pp. 305-318.

Schwindt C., 1995, “A new problem generator for different resource-constrained project scheduling problems with minimal and maximal time lags", WIOR-Report-449, Institut für Wirtschaftstheorie und Operations Research, University of Karlsruhe.

Tavares L.V., 1999, “Advanced models for project management", Kluwer Academic Publishers, Dordrecht.

Tavares L.V., J.A. Ferreira and J.S. Coelho, 1999, “The risk of delay of a project in terms of the morphology of its network", European Journal of Operational Research, Vol. 119, pp. 510-537.

Vandevoorde S., M. Vanhoucke, 2006, “A comparison of different project duration forecasting methods using earned value metrics", International Journal of Project Management, Vol. 24, pp. 289-302.

Vanhoucke M., 2011, “On the dynamic use of project performance and schedule risk information during project tracking", Omega The International Journal of Management Science, Vol. 39, pp. 416-426.

Vanhoucke M., 2012, “Project management with dynamic scheduling: Baseline scheduling, risk analysis and project control", Springer, Berlin.

Vanhoucke M., J.S. Coelho, D. Debels, B. Maenhout and L.V. Tavares, 2008, “An evaluation of the adequacy of project network generators with systematically sampled networks", European Journal of Operational Research, Vol. 187, pp. 511-524.

Vanhoucke M., S. Vandevoorde, 2007, “A simulation and evaluation of earned value metrics to forecast the project duration", Journal of the Operational Research Society, Vol. 58, pp. 1361-1374.

Van Peteghem V., M. Vanhoucke, 2013, “An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances", European Journal of Operational Research, doi: http://dx.doi.org/10.1016/j.ejor.2013.10.012.
Williams T., 1992, “Criticality in stochastic networks", Journal of the Operational Research Society, Vol. 43, pp. 353-357.
Zwikael O., S. Globerson and T. Raz, 2000, “Evaluation of models for forecasting the final cost of a project", Project Management Journal, Vol. 31, pp. 53-57.

An MILP Formulation for Scheduling of Work-Content-Constrained Projects

Philipp Baumann and Norbert Trautmann

University of Bern, Switzerland
philipp.baumann@pqm.unibe.ch and norbert.trautmann@pqm.unibe.ch

Keywords: Work-Content Constrained Project Scheduling, MILP Formulation

1 Introduction

Most models for project scheduling assume that all activities have constant durations and resource requirements. In contrast, in practical applications, e.g., pharmaceutical and software development projects, the project manager may change the resource usage of an activity over time in order to use the scarce project resources more efficiently.

This gives rise to the following project scheduling problem. One renewable work-content resource is available, and for each activity a work content is prescribed. The number of units of the work-content resource allocated to an activity may vary over time, as long as the required total work content is met. During execution, the number of units allocated must met a lower and an upper bound. Moreover, a minimum time lag between consecutive changes of the usage of the work-content resource (minimum block length for short) is prescribed. Besides the work-content resource an activity often requires further resources such as tools or supplies; the requirement for these resources depends on the amount of the work-content resource used. Furthermore, the activities are subject to finish-start precedence relationships. The planning problem consists in scheduling the activities of such a project such that the project duration is minimized.

For this problem, Fündeling (2006) presents a branch-and-bound method. Even though the approach is tailored to the specific problem setting, only few and small-sized problem instances can be solved to optimality. To tackle large-scale instances, Fündeling and Trautmann (2010) develop a priority-rule method that schedules activities iteratively using a specific schedule-generation scheme. The quality of the generated schedules could not have been evaluated thoroughly because neither optimal solutions nor strong lower bounds are known. Both approaches address the problem setting in which in a feasible schedule, the total number of resource units allocated to each activity must coincide with its prescribed work content. Besides, the discrete time/resource trade-off problem has been discussed in the literature (cf., e.g., Demeulemeester *et al.* 2000 or De Reyck *et al.* 1998). In that problem, only the work-content resource is considered, and an execution mode must be selected for each activity; each mode corresponds to a combination of a duration and a constant resource usage such that the total number of resource units allocated to the activity is equal to or larger than its prescribed work content. Recently Naber and Kolisch (2013) have presented four different mixed-integer linear programming (MILP) formulations for a relaxation of the problem discussed in this chapter; they assume that the total number of resource units allocated to an activity is to be greater than or equal to its prescribed work content, and that the capacity of the renewable resources can be divided continuously among the activities.

In this paper, we present an MILP formulation of the above-described problem. From the 480 problem instances with 10 activities that were introduced in Fündeling and Trautmann (2010), this model formulation solves 81% to optimality within short CPU times. We also analyze how the performance improves when the integrality constraint for the

resource allocation is dropped. For further details about our model formulation, we refer to Baumann and Trautmann (2013).

The remainder of the paper is organized as follows. In Section 2, we present the MILP formulation. In Section 3, we summarize our computational results. Section 4 concludes.

2 MILP SCHEDULING MODEL

We base our formulation on a discrete representation of time and on binary variables X_{it} that are equal to 1 if activity i is processed in period t, and 0 otherwise. We use the following notation:

i	Activity
t	Time period
k	Resource
k^*	Work-content resource
V	Activities
T	Periods
$\mathcal{R}$	Resources
V^r	Real activities
V^r_{kt}	Real activities that require resource $k \in \mathcal{R}$ and can be processed in period $t \in T$
T_i	Relevant periods for activity i ($T_i = \{EST_i + 1, \ldots, LFT_i, LFT_i + 1\}$)
$\mathcal{R}_i$	Resources required by activity i
P_i	Immediate predecessors of activity i
w_i	Work content of activity i
$\overline{r}_{ik}$	Upper bound on amount of resource $k \in \mathcal{R}$ used by activity i
$\underline{r}_{ik}$	Lower bound on amount of resource $k \in \mathcal{R}$ used by activity i
m	Minimum block length
s_{ik}	Requirement for resource $k \in \mathcal{R} \setminus k^*$ by activity i per unit of work-content
R_k	Capacity of resource $k \in \mathcal{R}$
R_{ikt}	Amount of resource $k \in \mathcal{R}$ used by activity $i \in V^r_{kt}$ in period $t \in T_i$
X_{it}	$\begin{cases} = 1, \text{if activity } i \text{ is processed in period } t \in T_i \\ = 0, \text{otherwise} \end{cases}$
D_{it}	$\begin{cases} = 1, \text{if usage of resource } k^* \text{ by activity } i \in V^r \text{ in period } t \in T_i \text{ differs from } t-1 \\ = 0, \text{otherwise} \end{cases}$

For each activity $i \in V$, we determine the set of available periods T_i based on the respective earliest start and latest finish times (EST_i, LFT_i). The decision variables related to activity i are only defined for these periods. The earliest start times are computed by forward recursion. Thereby the earliest start time of an activity is set to the latest earliest finish time of all immediate predecessor activities. The earliest finish time of an activity is computed by adding the lower bound on its duration $\lceil w_i / \overline{r}_{ik^*} \rceil$ to its earliest start time. The latest finish times are computed similarly using backward recursion.

The dummy activity $n+1$ is scheduled within the planning horizon, i.e.,

$$\sum_{t \in T_{n+1}} X_{n+1,t} = 1 \tag{1}$$

Activities may not be interrupted; more precisely, if activity i is executed in period $t-1$ and its work content has not been entirely processed by the end of this period, then activity i must also be processed in period t, i.e.,

$$X_{i,t-1} - \sum_{t' \in T_i : t' < t} \frac{R_{ik^*t'}}{w_i} \leq X_{it} \qquad (i \in V^r;\ t \in T_i :\ t > EST_i + 1) \tag{2}$$

Activity i can only be started if the total work content of all of its predecessors has been processed, i.e.,

$$X_{it} \leq \sum_{t' \in T_{i'}: t' < t} \frac{R_{i'k^*t'}}{w_{i'}} \qquad (i \in V;\ i' \in P_i;\ t \in T_i) \tag{3}$$

The work content of each activity is to be processed exactly during its execution, i.e.,

$$\sum_{t \in T_i} R_{ik^*t} = w_i \qquad (i \in V^r) \tag{4}$$

The lower and the upper bound on the usage of the work content resource must be met, i.e.,

$$\underline{r}_{ik^*} X_{it} \leq R_{ik^*t} \qquad (i \in V^r;\ t \in T_i) \tag{5}$$

$$\overline{r}_{ik^*} X_{it} \geq R_{ik^*t} \qquad (i \in V^r;\ t \in T_i) \tag{6}$$

Variable D_{it} must be equal to 1 if the usage of the work-content resource by activity i in period $t-1$ differs from the usage in period t. When an activity is completed in period $t = LF_i$, the variable D_{it} of the period $t = LF_i + 1$ will capture the last jump in the usage of the work-content resource.

$$R_{ik^*t} \leq \overline{r}_{ik^*} D_{it} \qquad (i \in V^r;\ t = EST_i + 1) \tag{7}$$

$$R_{ik^*t} - R_{ik^*,t-1} \leq \overline{r}_{ik^*} D_{it} \qquad (i \in V^r;\ t \in T_i:\ t > EST_i + 1) \tag{8}$$

$$R_{ik^*,t-1} - R_{ik^*t} \leq \overline{r}_{ik^*} D_{it} \qquad (i \in V^r;\ t \in T_i:\ t > EST_i + 1) \tag{9}$$

An activity i must not be processed after its latest finish time, i.e.,

$$X_{it} = 0 \qquad (i \in V^r;\ t = LFT_i + 1) \tag{10}$$

The minimum block length m implies that only one jump in the usage of the work-content resource during m consecutive periods $t \in T_i$ is allowed, i.e.,

$$\sum_{t'=0}^{m-1} D_{i,t+t'} \leq 1 \qquad (i \in V^r;\ t \in T_i:\ t \leq LFT_i - (m-2)) \tag{11}$$

The requirement of the non-work-content resources depends on the usage of the work-content resource. Here, a linear relation between the requirement of the non-work-content resources and the usage of the work-content resource is assumed, i.e., an increase (decrease) in the usage of the work-content resource results in a proportional increase (decrease) of the requirement for further resources. However, since we assume that all resources are discrete in nature, fractional requirements have to be rounded up. Thus, the requirement R_{ikt} of resource $k \in \mathcal{R} \setminus k^*$ by activity i in period t is $R_{ikt} = \lceil \underline{r}_{ik} + s_{ik}(R_{ik^*t} - \underline{r}_{ik^*}) \rceil$, where $s_{ik} = \frac{\overline{r}_{ik} - \underline{r}_{ik}}{\overline{r}_{ik^*} - \underline{r}_{ik^*}}$ represents the requirement per unit of the work-content resource k^*. The requirement of resource $k \in \mathcal{R} \setminus \{k^*\}$ of activity i in period t is computed by constraints (12). As this requirement needs to be an integer value, we use integer variables R_{ikt} to round up fractional values.

$$\underline{r}_{ik} X_{it} + s_{ik}(R_{ik^*t} - \underline{r}_i k^*) \leq R_{ikt} \qquad (i \in V^r;\ k \in \mathcal{R}_i \setminus \{k^*\};\ t \in T_i) \tag{12}$$

The total requirement for any resource $k \in \mathcal{R}$ must not exceed its capacity R_k, i.e.,

$$\sum_{i \in V^r_{kt}} R_{ikt} \leq R_k \qquad (k \in \mathcal{R};\ t \in T) \tag{13}$$

The objective is to minimize the duration of the project. The dummy activity $n+1$ represents the end of the project as it can only be scheduled after the completion of all real activities V^r. We write the objective function to be minimized as $\sum_{t \in T_{n+1}} t X_{n+1,t} - 1$.

3 Computational Results

We implemented the proposed model in AMPL and used Gurobi 5.5 with a prescribed time limit of 10 minutes to solve the 480 problem instances introduced in Fündeling and Trautmann (2010). The computations were performed on a standard workstation with two Intel Xeon 3.1GHz CPUs and 128GB RAM. For each problem instance we computed an upper bound on the makespan using the method of Fündeling and Trautmann (2010). To facilitate the generation of feasible solutions, we set the latest finish time of the dummy activities $n+1$ to the respective upper bound multiplied by a factor of 1.05.

Table 1 summarizes the results for the novel model with and without integrality constraints on the resource allocation variables. We report the number of instances for which a feasible solution was found within the time limit, and the number of instances for which optimality was proven within the time limit. For all instances for which a feasible solution has been devised, but the optimality of the best solution found could not be proven, we state the average MIP gap. In columns 5 and 6, we state the average relative deviation of our feasible solutions to the lower bounds computed in Fündeling and Trautmann (2010), and to the solutions found by the method of Fündeling and Trautmann (2010). The proposed model compares favorably to the method of Fündeling and Trautmann (2010). Moreover, when the integrality constraints are relaxed, almost all instances are solved to optimality.

Table 1. Computational Results

Resource allocation	# Feasible	# Optimal	∅ MIP gap [%]	∅ ΔLB [%]	∅ ΔFueTra10 [%]
Integer	430	389	5.02	16.15	-2.68
Continuous	480	476	1.60	6.95	–

4 CONCLUSIONS

We presented a novel MILP formulation for the work-content constrained project scheduling problem. Our model covers all particularities of this problem, in particular the minimum block-length constraint and the dependent requirement for non-work-content resources. Our computational results indicate that the proposed model outperforms the state-of-the-art method. We have analyzed the impact of the integrality constraint on the resource allocation; the computational burden is reduced considerably when this constraint is dropped. In future research, work-content based precedence relationships should be considered.

References

Baumann P., N. Trautmann, 2013, "Optimal Scheduling of Work-Content-Constrained Projects", *PROC 2013 IEEE IEEM CONF*, Laosirihongthong T Jiao R Xie M Sirovetnukul R (eds.)

Demeulemeester E., B. de Reyck, W. Herroelen, 2000, "The discrete time/resource trade-off problem in project networks: A branch-and-bound approach", *IIE TRANS*, Vol. 32, pp. 1059-1069.

De Reyck B., E. Demeulemeester, W. Herroelen, 1998, "Local search methods for the discrete time/resource trade-off problem in project networks", *NAV RES LOG*, Vol. 45, pp. 553-578.

Fündeling C., N. Trautmann, 2010, "A priority-rule method for project scheduling with work-content constraints.", *EUR J OPER RES*, Vol. 203, pp. 568-574.

Fündeling C., 2006, "Ressourcenbeschränkte Projektplanung bei vorgegebenen Arbeitsvolumina.", *Gabler, Wiesbaden.*

Naber A., R. Kolisch, 2013, "MIP models for resource-constrained project scheduling with flexible resource profiles", Unpublished working paper, TU München.

A Rough-Cut Capacity Planning Model with Overlapping

Baydoun G.[1], Haït A.[2] and Pellerin R.[1]

[1] École Polytechnique de Montréal, Montréal, Canada
georges.baydoun, robert.pellerin@polymlt.ca
[2] Université de Toulouse, Institut Supérieur de l'Aéronautique et de l'Espace, Toulouse, France
alain.hait@isae.fr

Keywords: Rough-Cut Capacity Planning, Concurrent Engineering, Overlapping.

1 Introduction

The overlapping of activities is a common practice to accelerate the execution of engineering projects. This technique consists in executing in parallel two sequential activities by allowing a downstream activity to start before the end of an upstream activity based on preliminary information. However, the overlapping of activities can entail rework tasks and modifications further to the transmission of complementary information after the start of the downstream activity (Grèze *et. al.* 2012). Activity overlapping thus allows reducing the total duration of project execution, at the expense of additional workload and execution cost associated to rework tasks. In this paper, we propose a Rough-Cut Capacity Planning (RCCP) model that determines the order of execution in time of a set of work packages (WPs) so as to minimize the total project duration and/or project cost, while respecting precedence relations, resource constraints and considering overlapping possibilities. The model was developed for supporting the project planning function in the early phases of projects by considering variable activity intensities, and aggregate resource capacities. The reminder of the paper is organized as follow. We first give a brief state of the art of existing RCCP and Resource-Constrained Project Scheduling Problem (RCPSP) models and overlapping models. We then introduce our RCCP model with multiple overlapping modes. In section 4, we present preliminary results before concluding the paper in section 5.

2 Related work

Several authors have studied the relation between rework and the amount of overlap in project conducted in a concurrent engineering context but only few papers have incorporated overlapping in the RCPSP. For instance, Gerk and Qassim (2008) proposed a linear model for project acceleration using crashing, overlapping and activity substitution. This model assumes that the relation between overlap amount and rework is continuous and linear. Grèze *et. al.* (2012) proposed a more realistic approach by restricting overlapping possibilities to a set of feasible overlap durations for each couple of overlappable activities, instead of considering a continuous and linear relation between overlap amount and rework. This assumption is more realistic as scheduling is performed in practice on a period-by-period basis (i.e. resource availabilities and allocations are determined per period). Also, overlapping points between activities is defined through clear document or information exchange in a concurrent engineering context, which limits the overlapping modes to a reduced and discrete set of possibilities.

However, these RCPSP models are not suited for planners in the early phases of projects as detailed activity content and resources are not known with precision and as work intensity is assumed to be constant over execution time. Indeed, planners tend to adopt an

aggregate planning approach in large engineering projects (Cherkaoui *et. al.* 2013) where WPs are broadly defined as group of multiple activities that could extend on a long period (weeks or months). In that context, RCCP models are better suited by dividing the planning horizon into time buckets (or periods) used to evaluate critical resource usage and by allowing resource allocation to WP to vary from one period to another (De Boer 1998). As such, a WP may start or end during a period; it is therefore possible to plan a WP and its successor within the same period.

Among existing RCCP models, Hans (2001) proposed an exact approach that consists in determining the periods where each job can be executed, and then specifying the fractions of the WP contents that are actually executed in each period. Several heuristic approaches have also been proposed to solve the RCCP problem including constructive heuristics (De Boer 1998) and linear-programming based heuristics (Gademann and Schutten 2005). However, none of these models considers overlapping and rework. To fill this gap, we propose a mixed integer linear-programming model, that is an extension of the RCCP model proposed by Haït and Baydoun (2012), where predecessor-successor WPs can overlap according to multiple overlapping modes. This extended model assumes that each overlapping mode can be defined by the percentage of execution of predecessor WP that needs to be reached in order to start the successor, as well as the amount of reworks on both WPs. The model is further explained in the following section.

3 RCCP model with overlapping

The model of Haït and Baydoun (2012) combines a continuous time representation of events and a discrete time evaluation of resources. A set of binary variables ensures the relation between starting time, ending time and durations over periods. These durations give minimum and maximum workload that can be assigned to the period. Our new model is based on the same representation of time and events of WP start and WP end, but adds a third type of events: intermediate milestone attainment.

Table 1. Nomenclature : sets & parameters / variables

P, D, H	Set of time periods ($p \in P$), duration of a period, time horizon $H = D \cdot \|P\|$
I	Set of work packages ($i \in I$)
$Succ_i, C_i$	Set of successors of i that can overlap on i ($j \in Succ_i$), $C_i = \|Succ_i\|$
M_i	Set of overlapping modes between i and its direct successors ($m \in M_i$)
Pos_{ijm}	Position of j among the successors of i according to mode $m \in M_i$
L^c_{im}	Required workload of part c of WP i in mode m
$L^{pred}_{ijm}, L^{succ}_{ijm}$	Required rework on i (respectively on j) in mode $m \in M_i$ due to overlapping between i and j
$t^0_i, t^{C_i+1}_i$	Starting time and ending time of i
t^c_i	For $c \in 1..C_i$: ending time of part c of i ($t^{c+1}_i \geq t^c_i \quad \forall i \in I, c \in \{0..C_i\}$)
e_{im}	Binary variable that equals 1 if mode m is chosen for i ($\sum_{m \in M_i} e_{im} = 1 \quad \forall i \in I$)
d^c_{ip}	Duration of part c of i within period p
d_{ijp}	Duration of overlapping between i and j within period p
l^c_{ip}	Workload of part c of i during period p
$l^{pred}_{ijp}, l^{succ}_{ijp}$	Rework on i (resp. j) during p due to overlapping between i and j

An overlapping mode between a WP i and its successor j is defined regarding the attainment – in terms of workload – of a milestone within WP i; the successor can start

once the milestone is reached. Each WP i is therefore divided into $C_i + 1$ parts, the end of each part matching the time when a milestone of i is reached. Each part has its own set of durations over the periods, denoted d^c_{ip}, and assigned workload during the periods, l^c_{ip}. Variables d^c_{ip} are used to guarantee that workloads l^c_{ip} always respect the minimum and maximum intensities.

Constraints (1) ensure that the required workload is attained for each part of WP i according to the selected mode.

$$\sum_{p \in P} l^c_{ip} \geq L^c_{im} \cdot e_{im} \qquad \forall i \in I, c \in \{1..C_i + 1\}, m \in M_i \tag{1}$$

The end of each part gives the time t^c_i when a successor in $Succ_i$ can start. Constraints (2) ensure that successor j begins after the correct milestone of i. For successors $\widetilde{j}$ that cannot overlap on i, a classical end-to-start constraint is defined: $t^0_{\widetilde{j}} \geq t^{C_i+1}_i$.

$$t^0_j \geq t^c_i - H \cdot (1 - e_{im}) \qquad \forall i \in I, m \in M_i, j \in Succ_i, c \in \{1..C_i + 1 | c = Pos_{ijm}\} \tag{2}$$

Figure 1 presents an example of a WP (A) with two successors (B, C) that can overlap. A is therefore divided into three parts. In the depicted mode, the milestone corresponding to C comes before the one corresponding to B. Consequently C can start after the end of the first part of A, while B can start after the end of the second part of A.

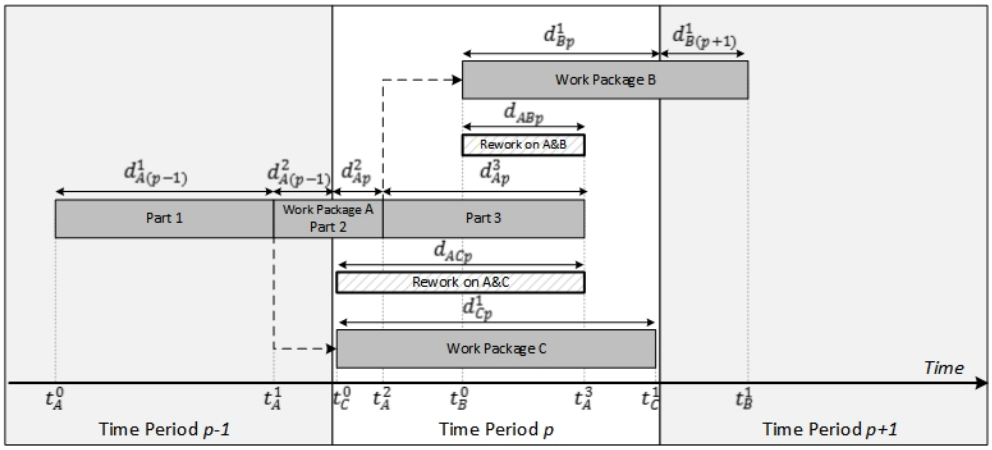


Fig. 1. A work package (A) with two successors (B, C) that can overlap.

In addition to initial workloads, reworks are required on predecessors and successors that overlap. The reworks should be executed during the overlapping time between predecessors and successors. Workload variables l^c_{ip}, l^{pred}_{ijp}, and l^{succ}_{ijp} are linked to durations d_{ijp} and d^c_{ip} to ensure the respect of minimum and maximum allowed intensities. Constraints (3) and (4) make sure that the total executed reworks match the required reworks in selected modes. As our objective functions minimise project cost and/or delivery time, inequalities are used instead of equalities for constraints 1, 3 and 4.

$$\sum_{p \in P} l^{pred}_{ijp} \geq L^{pred}_{ijm} \cdot e_{im} \qquad \forall i \in I, m \in M_i, j \in Succ_i \tag{3}$$

$$\sum_{p \in P} l^{succ}_{ijp} \geq L^{succ}_{ijm} \cdot e_{im} \qquad \forall i \in I, m \in M_i, j \in Succ_i \tag{4}$$

4 Main results

In order to test our model, we used instances that were generated by De Boer (1998) and that are commonly used to test RCCP models. These instances were modified in order to handle overlapping. Half of predecessor-successor couples were randomly selected and were allowed to overlap. Each selected couple has a maximum of two overlapping modes (80% and/or 60% of the workload of predecessors) and reworks were fixed proportionally (factor of 0.4) to the maximum workload that can be overlapped.

We tested our model using CPLEX 12.5 on instances, with and without overlapping. We solved the problem with two objective functions: project cost (cost of using external resources and adding reworks), and project delivery time. The results for one instance of 10 WPs (table 2) show that overlapping is beneficial in both cases. Note that the objective can easily be adapted in order to make a trade-off between project cost and delivery time.

Table 2. Results for instance $RCCP125$

	Without overlapping			**With overlapping**			
	Cost	Delivery time	CPU time	Cost	Delivery time	Total rework	CPU time
Minimise cost	3269.9	31	1.75 s	3261.5	30.313	33.84	5 s
Minimise delivery time	3540.2	29	1.5 s	3673.4	27.54	119.04	2.57 s
Minimise delivery time with a target cost of 3540.2				3540.2	28.031	119.04	4.28 s

5 Conclusion

In conclusion, we proposed an interesting extension of the RCCP that allows overlapping of work packages. Preliminary results show that overlapping adds a flexibility in distributing the workload for earlier delivery date or smaller cost of project.

Future work will focus on improving the performance of our model, and conduct an extended analysis on the benefits of overlapping, over a sufficient number of instances.

References

Cherkaoui, K., Pellerin, R., Baptiste, P. and Perrier, N., 2013 "Planification hiérarchique de projets EPCM", 10ème Conférence Internationale de Génie Industriel 2013, June 12-14, La Rochelle, France.

De Boer, R., 1998, "Resource-constrained multi-project management : a hierarchical decision support system", PhD thesis, University of Twente, Enschede, Netherlands.

Gademann, N. and Schutten, J.M.J., 2005, "Linear-programming-based heuristics for project capacity planning", *IIE Transactions*, Vol 37(2), pp. 153-165.

Gerk, J. and Qassim, R., 2008, "Project Acceleration via Activity Crashing, Overlapping, and Substitution", *IEEE Transactions on engineering management*, Vol. 55(4).

Grèze, L., Pellerin, R., Leclaire, P. and Perrier, N., 2012, "A heuristic method for resource-constraint project scheduling with activity overlapping", to appear in *Journal of Intelligent Manufacturing*, DOI 10.1007/s10845-012-0719-5.

Haït, A. and Baydoun, G., 2012, "A new event-based MILP model for the resource-constrained project scheduling problem with variable intensity activities", The IEEE International Conference on Industrial Engineering and Engineering Management 2012, Hong Kong, 10-13 December.

Hans, E., 2001, "Resource loading by Branch-and-Price techniques", PhD thesis, University of Twente, Enschede, Netherlands.

Minimizing total tardiness in two-machine permutation flowshop problem with availability constraints and subject to release dates

Abdelaziz Berrais[1], Mohamed Ali Rakrouki[1], and TalelLadhari[2]

[1]Taibah University, Al-Madinah, Saudi Arabia
[2]University of Tunis, Tunis, Tunisia
e-mail: aberrais@taibahu.edu.sa, mrakrouki@taibahu.edu.sa, ladhari_talel2004@yahoo.fr

Keywords: flowshop, total tardiness, availability constraints, heuristics.

1. Introduction

Scheduling problems consider often that machines are available during the scheduling, or in practice machines may be unavailable during several periods of time due to machine breakdown or preventive maintenance. In this work we consider a two-machine permutation flowshop problem with release dates of jobs in the deterministic case where unavailability periods of machines are fixed in advance. Moreover, we consider the resumable scheduling case (*r-a*) when the pre-emption is allowed. The objective is to minimize the total tardiness of jobs under the above mentioned assumptions.

The total tardiness criterion is very important in industries because the not respect of due dates can affect the reputation of the company and can cause lack of confidence, costs increasing, and losing of costumers.

The rest of this paper is organized as follows: the problem under consideration is described in Section2. In Section 3 we propose a mixed-integer formulation. In Section 4 we present five constructive heuristics. The experimental results and a comparative study are provided in Section 5. Finally, conclusions are given in Section 6.

2. Problem description

In this paper we study the two-machine flowshop problem with availability constraints in deterministic case and resumable machine in aim to minimize the total tardiness criterion subject to release dates. To the best of our knowledge this is the first attempt to solve this problem. In effect only makespan criterion has been widely studied for the two-machine flowshop scheduling problem with availability constraints. This problem can be stated as follows.

We are given a set of n jobs (j=1,...,n) must be processed during p_{1j} time units firstly on machine M_1 and then during p_{2j} time units on machine M_2. The two machines are available at time zero and can process at most one job at a time. The processing of job j cannot be started before a release date r_j and is subject to a due date d_j known in advance. We assume that at least one unavailability period (hole) can occur on each machine and there aren't two or more holes overlapping on the same machine. Also, job processing can be interrupted and resumed after. The aim is to compute a sequence of jobs which minimizes the $\sum_{j=1}^{n} T_j$criterion, with $T_j=max(0,C_j-d_j)$ where T_j and C_j are the tardiness and completion time of job j, respectively. This problem is denoted by $F2,h_{lo}|r\text{-}a,r_j|\sum T_j$, where h_{lo} is the number of holes (h) both on machine 1 (l) and on machine 2 (o). It is an extension of the $F2||\sum T_j$ known to be NP-hard in the strong sense (Lenstra et al., 1977).

3. Mixed-integer formulation

The problem under consideration can be formulated as follows. Let:

- X_{ij}: A binary variable which is equal to 1 if job j is assigned to position i and 0 otherwise. i=1,2,…,n and j=1,2,…,n
- T_k: tardiness at position k = 1,2,...,n

- u,v: the number of holes on M_1 and M_2, respectively
- t_l^i: duration of the hole number l on machine M_i (i=1,2).
- h_{lj}^i:a binary variable which is equal to 1 if the hole l occurs during the processing of the job assigned to the position j on machine M_i and 0 otherwise, with i=1,2 and j=1,2,...,n
- C_k: the completion time of job scheduled at position k.

Then, the problem can be formally formulated as follows:

$$\text{Minimize } \sum_{k=1}^{n} T_k \quad (1.1)$$

Subject to:

$$\sum_{j=1}^{n} X_{kj} = 1, \forall\, k = 1,\dots,n \quad (1.2)$$

$$\sum_{k=1}^{n} X_{kj} = 1, \forall\, j = 1,\dots,n \quad (1.3)$$

$$C_1 \geq \left(r_j + p_{1j} + p_{2j}\right)X_{1j} + \sum_{l=1}^{u} t_l^1 . h_{l1}^1 + \sum_{l=1}^{v} t_l^2 . h_{l1}^2, \forall\, j = 2,\dots,n \quad (1.4)$$

$$C_k \geq C_{k-1} + p_{2j}.X_{kj} + \sum_{l=1}^{v} t_l^2 . h_{lk}^2, \forall\, k = 2,\dots,n, \quad (1.5)$$

$$C_k \geq \sum_{i=1}^{k}\sum_{j=1}^{n} p_{1j}\, X_{ij} + \sum_{i=1}^{k}\sum_{l=1}^{u} t_l^1 . h_{li}^1 + \sum_{j=1}^{n} p_{2j} X_{kj} + \sum_{l=1}^{v} t_l^2 . h_{lk}^2, \forall\, k = 2,\dots,n \quad (1.6)$$

$$T_k \geq C_k - \sum_{j=1}^{n} d_j X_{kj}\,, \forall k = 1,\dots,n \quad (1.7)$$

$$T_k \geq 0, \forall k = 1,\dots,n \quad (1.8)$$

Constraints (1.2) imply that at each position there is only one job. Constraints (1.3) imply that each job must be scheduled at least at one position. Constraint (1.4) means that the completion time at the first position is greater than or equal to the sum of processing times and the release date and the durations of holes which occur at this position. Constraints (1.5) mean that the completion time of the job in position k is greater than or equal to the completion time of the job in position k-1 on the second machine plus the processing time of the job in position k plus the sum of the durations of holes which occur at this position. Constraints (1.6) mean that the completion time of the job at the position k is greater than or equal to the sum of processing times of jobs scheduled until position k on the first machine M_1 plus the sum of the durations of holes that occur until position k on the first machine plus the sum of processing times on machine M_2 of the job scheduled at position k plus the sum of the durations of holes which occur on machine M_2 in position k. Constraints (1.7) mean that the tardiness at position k is greater than the completion time at position k minus the due date of the job scheduled at this position.

4. Constructive heuristics

In this section we present five constructive heuristics for the problem under consideration. These heuristics are based on the well-known NEH algorithm (Nawaz et al., 1983) originally developed for minimizing the maximum completion time.

We denote by σ a partial sequence of scheduled jobs and $\bar{J}$the set of unscheduled jobs. The *Hi* (i=1,2,3,4,5)heuristics can be described as follows:

- Step 0: $\sigma = \emptyset; \bar{J} = J$
- Step 1: Arrange the jobs in $\bar{J}$ according to R_i(i=1,2,3,4,5) rule. Let $\pi = (\pi(1),\pi(2),\dots,\pi(n))$ be the resulting sequence
- Step 2: Among the partial sequences σ=$\pi(1),\pi(2)$ and σ=$\pi(2),\pi(1)$ select the one with the minimum partial total tardiness. Set $\bar{J} = \bar{J}\backslash\{\pi(1), \pi(2)\}$ and k=2
- Step 3: Select the job $\pi(k)$ and insert it to the k+1 possible position of σ. Among k+1 sequences, select the one with the minimum partial total tardiness and set it as the current σ. Set $\bar{J} = \bar{J}\backslash\{\pi(k)\}$
- Step 4: Repeat Step 3 until $\bar{J} = \emptyset$

Modifying the Step 1 to the following five alternatives yields five different versions.

H1: The jobs are ranked according to EDD (Earliest Due Date or Jackson's rule). We sort the jobs in non-decreasing order of their due date d_j(Jackson 1955).

H2: The jobs are ranked according to MDD (Modified Due Date). At each time *t*, we select the job *j* having the minimum value of $max\{d_j, C_j(\sigma)\}$, where $C_j(\sigma)$ is the completion time of job *j* if it is scheduled at the end of σ.

H3: The jobs are sorted using ST (Slack Time). At each time *t*, the job *j* with the minimum value of d_j-$C_j(\sigma)$ is selected.

H4: The jobs are sorted according to STRW (ST per Remaining Work) rule. At each step the job with minimum value of $\frac{d_j - C_j(\sigma)}{p_{1j}+p_{2j}}$ is selected.

H5: The sequence having the minimum total tardiness obtained with *Hi*(*i*=1,2,3,4) is considered.

5. Experimental results

This section describes the computational tests which have been conducted to evaluate the empirical performance of the proposed algorithms. Our algorithms have been coded in C++ and run on a Pentium (R) Dual-Core 2.30 GHz PC with 3 GB RAM.

5.1. Test problems

In order to assess the quality of the different proposed algorithms, we carried out a series of experiments on 30 randomly generated instances for 14 problem sizes n∈{10,20,30,40,50,60,70,80,90,100,200,300,400,500}. Thus, we solved a total of 420 test problems.

The processing times and release dates are uniformly distributed between [1,100], $\left[0, \frac{\sum_{j=1}^{n} p_{1j}}{2}\right]$, respectively. The due dates generating is inspired from the scheme of Pinedo and Singer (1999), d_j will be uniformly distributed in $[b_i, b_s]$ with $b_i = r_j + p_{1j} + p_{2j}$ and $b_s = r_j + 3(p_{1j} + p_{2j})$. Moreover, we consider 5 holes randomly generated on each machine.

5.2. Performance of the proposed heuristics

The performance analysis is based on the average relative percentage deviation (ARPD) from the best-known solution. The percentage deviation is defined as $\left[\frac{UB - UB^*}{UB^*}\right] \times 100$, where *UB* is the solution provided by the heuristic H_i and $UB^* = \min(UB_i)$; (*i*=1,...,5).

The results of the computational study of the proposed constructive heuristics are summarized in Table 1. The headings have the following meaning: *n*: number of jobs, *ARPD*: average relative percentage deviation from the best-known solution and *Time*: mean CPU time (in sec).

*Table 1.*Computational performance of the proposed constructive heuristics

n	H1		H2		H3		H4		H5	
	ARPD	**Time**	**ARPD**	**Time**	**ARPD**	**Time**	**ARPD**	**Time**	**ARPD**	**Time**
10	4.19	0.00	9.89	0.00	2.57	0.00	2.62	0.00	0.00	0.00
20	4.50	0.00	5.56	0.00	4.45	0.00	3.46	0.00	0.00	0.00
30	6.36	0.00	2.37	0.00	6.23	0.00	4.76	0.00	0.00	0.01
40	6.51	0.00	1.92	0.01	6.15	0.01	6.26	0.00	0.00	0.03
50	6.98	0.01	1.20	0.02	7.10	0.01	7.54	0.01	0.00	0.05
60	9.14	0.01	0.56	0.02	9.13	0.02	10.03	0.02	0.00	0.08
70	8.94	0.02	0.47	0.03	9.09	0.04	12.08	0.02	0.00	0.13
80	11.05	0.03	0.47	0.05	10.83	0.06	10.35	0.03	0.00	0.19
90	12.34	0.04	0.28	0.07	11.76	0.08	11.87	0.05	0.00	0.27
100	12.13	0.06	0.49	0.11	11.88	0.12	11.69	0.07	0.00	0.37
200	12.65	0.50	0.71	0.80	12.59	0.91	13.07	0.50	0.00	3.01
300	13.89	1.57	1.10	2.63	14.06	2.58	13.87	1.51	0.00	10.39
400	13.54	3.81	1.02	5.78	13.39	6.01	13.44	3.82	0.00	23.60
500	13.74	7.15	1.23	11.04	13.82	11.44	13.64	6.95	0.00	46.00
Avg.	***9.71***	***0.94***	***1.95***	***1.47***	***9.50***	***1.52***	***9.62***	***0.93***	***0.00***	***6.01***

The results of Table 1 provide strong evidence that H5 heuristic is very effective and outperforms all the proposed heuristics. Indeed H5 presents lower values of average relative percentage deviation from the best-known solution compared with the remaining heuristics for all instances.

Table 1 shows that the proposed algorithms can solve all problem sizes less than n=100 in a negligible mean CPU time (<1 sec). From Table 1, we see that H5 algorithm is able to solve very large instances with up 500 jobs within a moderate CPU time. For instance all the instances with 500 jobs were solved within an average CPU time equal to 46 sec.

6. Conclusions

This paper investigates the two-machine permutation flowshop with the objective of minimizing the total tardiness with availability constraints and subject to release dates. Despite its theoretical and practical importance, this NP-hard problem has not been investigated before. We proposed a mixed-integer formulation as well as some constructive heuristics. Our computational experiments that were carried out on a large set of randomly instances show that the proposed heuristics provides interesting results. An interesting issue which deserves further investigation is to develop lower bounds and an exact method for the problem under consideration.

References

Jackson, J.R., 1955 "Scheduling a production line to minimize maximum lateness", Research Report 43, Management science research report, University of California, Los Angeles.

Nawaz M., E.E.Enscore and I. Ham, 1983 "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem", OMEGA, Vol. 11, pp.91-96.

Lenstra J.K., A.H.G. Rinnooy Kan and P. Brucker1977 "Complexity of machine scheduling problems", Annals of Discrete Mathematics, Vol. 1, pp. 343-62.

Pinedo, M. and Singer, M., 1999 " A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. Naval Research Logistics, 46(1):1–17.

Resource Levelling in Project Scheduling with Generalized Precedence Relationships and Variable Execution Intensities

Lucio Bianco, Massimiliano Caramia and Stefano Giordani

University of Rome "Tor Vergata", Italy
{bianco,caramia,giordani}@dii.uniroma2.it

Keywords: Project scheduling, Generalized precedence relationships, Resource levelling.

1 Introduction

Generalized Precedence Relations (GPRs) are temporal constraints in which the starting/finishing times of a pair of activities have to be separated by at least or at most an amount of time denoted as "time lag" (minimum time lag and maximum time lag, respectively). GPRs can be classified into *Start-to-Start* (SS), *Start-to-Finish* (SF), *Finish-to-Start* (FS) and *Finish-to-Finish* (FF) relations. These four constraint types double when minimum and maximum time lags are concerned. In particular, a minimum time lag generates $(SS_{ij}^{\min}(\delta), SF_{ij}^{\min}(\delta), FS_{ij}^{\min}(\delta), FF_{ij}^{\min}(\delta))$ which specify that activity j can start (finish) only if its predecessor i has started (finished) at least δ time units before. Analogously, a maximum time lag $(SS_{ij}^{\max}(\delta), SF_{ij}^{\max}(\delta), FS_{ij}^{\max}(\delta), FF_{ij}^{\max}(\delta))$ imposes that activity j can be started (finished) at most δ time slots beyond the starting (finishing) time of activity i.

We study Resource Constrained Project Scheduling Problem (RCPSP) with GPRs. From the complexity viewpoint, the problem is strongly NP-hard and also the easier problem of detecting whether a feasible solution exists is NP-complete (Bartusch *et. al.*, 1988). Almost all the exact procedures presented in the literature for such a problem are conceived to work for the project completion time minimization, i.e., the branch and bound algorithms by Bartusch *et. al.* (1988), Demeulemeester and Herroelen (1997), De Reyck and Herroelen (1998), Schwindt (1998), Fest *et. al.* (1999), Dorndorf *et. al.* (2000), and Bianco and Caramia (2012).

Only a few exact approaches have been introduced in the literature to level resource profiles in projects with GPRs. Engelhardt and Zimmermann (1998), exploiting an idea of Ahuja (1976), proposed a method that enumerates all combinations of activity start times to minimize the sum over time of the squared changes in the resource utilization and several other objective functions. In 2001, Nübel presented a tree-based enumeration approach for the resource renting problem, and Neumann *et. al.* (2003) outlined how this approach can be used to solve the resource levelling problem. Gather and Zimmermann (2009) have sketched some weaknesses of the latter approach and developed a new and more efficient procedure, relying on the paper of Gabow and Myers (1978). Additional literature on RCPSP with GPRs can be found in the book of Neumann *et. al.* (2003).

In our work, we study a particular RCPSP with GPRs. Indeed, we consider the problem of levelling resources in a project with GPRs, given a deadline for the completion of all the activities and variable execution intensity of the activities. RCPSP with variable execution intensity has been taken into account firstly by Kis (2005) applied to a real world scenario in which, due to the physical characteristics of some manufacturing processes, the effort associated with a certain activity for its execution may vary over time. An example is that of the human resources that can be shared among a set of simultaneous activities in

proportion variable over time. In this case, the amount of work per time unit devoted to each activity and, consequently, also its duration are not univocally defined. Therefore, the problem tackled in our work is as follows: given a set of K types of renewable resources, with a_k and c_k being, respectively, the available amount of resource type k and its additional resource unitary cost per time period, a set of activity $i = 1, \ldots, n$, with activity i requiring a total amount $\bar{r}_{ik}$ of resource k and having a duration ranging between a minimum and a maximum value d_i^{min} and d_i^{max}, respectively, and a set of GPRs constraints, we want to schedule these activities levelling the usage of resources over time, e.g., minimizing the total additional resource cost, while respecting a deadline D. To the best of our knowledge this is a novel problem. In the following section, we propose a mathematical formulation for such a problem.

2 The Mathematical Model

In the following, we assume that there is a planning horizon within which all the activities may be carried out. In particular, we denote such a planning horizon as $[0, T)$, where T is an upper bound on the minimum project completion time. Trivially, we consider $T \geq D$. Moreover, we assume, without loss of generality, that the time horizon is discretized into T unit-width time periods $[0, 1), [1, 2), \ldots, [T-1, T)$, indexed by $t = 1, \ldots, T$, respectively. The project is formed by n real activities, $i = 1, \ldots, n$, and by two dummy activities, 0 and $n + 1$, corresponding to the source and sink nodes of the project network. The latter is assumed to be in an activity-on-nodes standardized form (see, Bartusch *et. al.* 1998) without positive directed cycles. Let us define the following parameters:

- K, the number of renewable (continuously divisible) resources, each one available in an amount of a_k units, with $k = 1, \ldots, K$;
- c_k, the unitary cost of additional resources of type k per time period;
- $\bar{r}_{ik}$, the overall amount of units of resource k necessary to carry out activity i;
- d_i^{max}, the maximum duration of activity i;
- d_i^{min}, the minimum duration of activity i;
- D, the deadline to complete all the activities;
- E, the set of ordered pairs of activities constrained in the standardized activity-on-nodes project network defined by GPRs.

Furthermore, let us consider the following decision variables:

- u_{kt}, the type k resource usage during time period t;
- x_{it}, the fraction of activity i executed within the end of time period t;
- s_{it}, a binary variable that assumes value 1 if activity i has started within the beginning of time period t, and assumes value 0 otherwise;
- f_{it}, a binary variable that assumes value 1 if activity i has finished within the end of time period t, and assumes value 0 otherwise.

The mathematical model is as follows:

$$\min \sum_{t=1}^{T} \sum_{k=1}^{K} c_k \max(0, u_{kt} - a_k) \tag{1}$$

$$s.t. \quad x_{it} - x_{i,t-1} \geq \frac{1}{d_i^{max}}(s_{it} - f_{i,t-1}), \quad i = 1, \ldots, n; \quad t = 1, \ldots, T \tag{2}$$

$$x_{it} - x_{i,t-1} \leq s_{it} - f_{i,t-1}, \quad i = 1, \ldots, n; \quad t = 1, \ldots, T \tag{3}$$

$$\sum_{t=1}^{T} s_{it} \geq \sum_{t=1}^{T} s_{jt} + \delta_{ij}, \quad \forall (i, j) \in E \tag{4}$$

$$s_{it} \leq s_{i,t+1}, \quad i = 0, \ldots, n+1; \quad t = 1, \ldots, T \tag{5}$$

$$f_{i,t-1} \leq f_{it}, \quad i = 0, \ldots, n+1; \quad t = 1, \ldots, T \tag{6}$$

$$x_{iT} = f_{iT} = s_{iT} = 1, \quad i = 1, \ldots, n \tag{7}$$

$$x_{i0} = f_{i0} = 0, \quad i = 1, \ldots, n \tag{8}$$

$$f_{it} \leq x_{it}, \quad i = 1, \ldots, n; \quad t = 1, \ldots, T \tag{9}$$

$$x_{it} \leq s_{it}, \quad i = 1, \ldots, n; \quad t = 1, \ldots, T \tag{10}$$

$$f_{00} = s_{01} = 1 \tag{11}$$

$$f_{n+1,T} = s_{n+1,T+1} = 1 \tag{12}$$

$$\sum_{i=1}^{n} \bar{r}_{ik}(x_{it} - x_{i,t-1}) = u_{kt}, \quad k = 1, \ldots, K; \quad t = 1, \ldots T \tag{13}$$

$$t(s_{n+1,t+1} - s_{n+1,t}) \leq D, \quad t = 1, \ldots, T \tag{14}$$

$$1 + \sum_{t=1}^{T}(s_{it} - f_{it}) \geq d_i^{min}, \quad i = 1, \ldots, n \tag{15}$$

$$\sum_{t=1}^{T}(x_{it} - x_{i,t-1}) = 1, \quad i = 1, \ldots, n \tag{16}$$

$$u_{kt} \geq 0, \quad k = 1, \ldots, K; \quad t = 1, \ldots, T \tag{17}$$

$$x_{it} \geq 0, \quad i = 1, \ldots, n; \quad t = 1, \ldots, T \tag{18}$$

$$s_{it} \in \{0, 1\}, \quad i = 0, \ldots, n+1; \quad t = 1, \ldots, T+1 \tag{19}$$

$$f_{it} \in \{0, 1\}, \quad i = 0, \ldots, n+1; \quad t = 0, \ldots, T \tag{20}$$

The objective function (1) minimizes the total cost of the additional resource usage. Constraints (2) and (3) regulate the minimum and maximum fraction of activity i to be executed in time period t. Constraints (4) model *Start-to-Start* precedence constraints with time-lags δ_{ij}, $(i, j) \in E$, where δ_{ij}'s depend in general also on the durations of activities i and j, respectively, being the duration of activity i equal to $1 + \sum_{t=1}^{T}(s_{it} - f_{it})$. Constraints (5) and (6) are congruency constraints on the values assumed by variables s_{it} and f_{it} over time. Constraints (7) say that every activity i must start and finish within the planning horizon. Constraints (8) represent the initialization conditions for variables x_{it}, and f_{it} when $t = 0$. Constraints (9) and (10) force f_{it} to be zero if $x_{it} < 1$, and x_{it} to be zero if $s_{it} = 0$. Constraints (11) and (12) are initialization conditions for dummy activities 0 and $n+1$, respectively. Resource usage is represented by relations (13). Constraints (14) impose that the project must finish within the deadline D. Constraints (15) force the duration of each activity i to be not less than d_i^{min}. Constraints (16) say that activity i must be completely executed. Constraints (17), (18), (19), and (20) limit the range of variability of the variables.

In order to linearize the objective function, we introduce additional variables $\bar{u}_{kt} \geq 0$, $k = 1, \ldots, K, t = 1, \ldots, T$, and rewrite (1) as $\min \sum_{i=1}^{T} \sum_{k=1}^{K} c_k \bar{u}_{kt}$, with the additional constraints $\bar{u}_{kt} \geq u_{kt} - a_k, \quad k = 1, \ldots, K; \quad t = 1, \ldots, T$, which, by constraints (13), may be rewritten as $\bar{u}_{kt} \geq \sum_{i=1}^{n} \bar{r}_{ik}(x_{it} - x_{i,t-1}) - a_k, \quad k = 1, \ldots, K; \quad t = 1, \ldots, T$.

We solve the proposed model by means of an exact branch and bound algorithm where the lower bound is based on a Lagrangian relaxation of the latter resource constraints.

We conducted preliminary tests on two sets of instances. First of all, we used the well known rlp_j10 test set devised in Weglarz (1998); this test set contains 270 test instances with 10 activities and 1-5 resources. Furthermore, we considered 45 instances of the rlp_j20 test set (see Weglarz, 1998) with 20 activities and 1-5 resources (instances 136-180). In the

columns of Table 1, we reported: the test type, the average objective value `AOV` obtained, the average value `AQCV` of the function $\sum_{k=1}^{K}\sum_{t=1}^{T} c_k u_{kt}^2$ associated with the optimal solutions achieved by our model, the average branch and bound nodes `B&B_Nodes`, the average CPU time (in seconds) to solve at the optimum the instances (when optimality is achieved), the number `#_opt` of instances solved at the optimum within a time limit of 7200 seconds, the number `#_opt_300_s` of instances solved within 300 seconds.

Table 1. Computational results

Test Type	AOV	AQCV	B&B_Nodes	Average_CPU_Time	#_opt	#_opt_300_s
rlp_j20	380	4378	4402	89s	270	270
rlp_j20 (instances 136-180)	805	8213	9225	2512s	35	18

References

1. Ahuja H.N., 1976, "Construction Performance Control by Networks", John Wiley and Sons, New York.
2. Bartusch M., R.H. Möhring and F.J. Radermacher, 1988, "Scheduling Project Networks with Resource Constraints and Time Windows", *Ann Oper Res*, Vol. 16(1-4), pp. 201–240.
3. Bianco L., M. Caramia, 2012, "An exact algorithm to minimize the makespan in project scheduling with scarce resources and generalized precedence relations" *Eur J Oper Res*, Vol. 219(1), pp. 73–85.
4. Demeulemeester E.L., W.S. Herroelen, 1997, "A branch-and-bound procedure for the generalized resource-constrained project scheduling problem", *Oper Res*, Vol. 45(2), pp. 201–212.
5. De Reyck B., W. Herroelen, 1998, "A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations", *Eur J Oper Res*, Vol. 111(1), pp. 152–174.
6. Dorndorf U., E. Pesch and T. Phan-Huy, 2000, "A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints", *Manage Sci*, Vol. 46(10), pp. 1365–1384.
7. Engelhardt H., J. Zimmermann, 1998, "Lower Bounds and Exact Methods for Resource Levelling Problems", Technical Report WIOR (Universität Karlsruhe), 517.
8. Fest A., R.H. Möhring, F. Stork and M. Uetz, 1999, "Resource-constrained project scheduling with time windows: A branching scheme based on dynamic release dates", Technical Report 596, Technical University of Berlin.
9. Gabow H.N., Myers E.W., 1978, "Finding all spanning trees of directed and undirected graphs", *Siam Journal on Computing*, Vol 7, pp. 280–287.
10. Gather T., J. Zimmermann, 2009, "Exact methods for the resource levelling problem", Proceedings of MISTA Conference 2009, pp. 811-820.
11. Kis T., 2005, "A branch-and-cut algorithm for scheduling of projects with variable-intensity activities", *Mathematical Programming*, Vol. 103(3), pp. 515–539.
12. Neumann K., C. Schwindt and J. Zimmerman, 2003, "Project scheduling with time windows and scarce resources", *Lect Notes Econ Math*, Vol. 508, 2nd edition, Springer Verlag, Berlin.
13. Nübel H., 2001, "The resource renting problem subject to temporal constraints", *OR Spektrum*, Vol. 23, pp. 359–381.
14. Schwindt C., 1998, "Verfahren zur Lösung des ressourcenbeschränkten Projektdauerminimierungsproblems mit planungsabhängigen Zeitfenstern", Shaker, Aachen.
15. Weglarz J., 1999, "Project Scheduling: Recent Models, Algorithms, and Applications", Kluwer, Boston.

A single machine scheduling problem with bin packing constraints

Jean-Charles BILLAUT[1], Federico DELLA CROCE[2] and Andrea GROSSO[2]

[1] L.I.,Université François-Rabelais Tours, France jean-charles.billaut@univ-tours.fr
[2] D.A.I., Politecnico di Torino, Italy federico.dellacroce@polito.it
[3] D.I., Universitá di Torino, Italy grosso@di.unito.it

Keywords: Scheduling, Bin packing, Heuristics.

We consider a single machine scheduling problem with additional bin packing constraints. The origin of the problem comes from the production of chemotherapy drugs (Mazier *et. al.* 2010). In this production environment, raw materials are called monoclonal antibodies and can be stored in vials for a long time before use (Billaut 2011). However, once a vial is opened or once the active agent has to be mixed with some water, it must be used before a given time limit, in order to keep intact the properties of anticancer active agents. The maximum delay of use after opening depends on the agent and may vary between several hours to several days. In the mean time, the product has to be stored in a fridge for temperature and darkness reasons. The cost of these drugs is not negligible and the economic impact of saving these products is very important. The preparations that are scheduled and which use the same vial have to be completed before the product perishes. In other words, the total processing time of the jobs assigned to a same vial cannot be greater than the life time of the raw material. Furthermore, the total consumption cannot be greater than the total volume of the vial. Because each preparation has to be delivered to a patient for a given due date, one objective of the problem is to minimize the maximum lateness related to these due dates. Notice that in such a context, the deadline for the use of the raw material becomes a variable of the problem, which is directly related to the production scheduling decisions: once the life duration or the capacity of the vial is exceeded, a new vial is opened.

1 Problem statement and notations

We consider a simplified version of the above problem, with only one machine and one type of raw material. We consider a set of n jobs to schedule on a single machine. W.l.o.g., the jobs are supposed to be numbered in EDD order, i.e. $d_1 \leq d_2 \leq ... \leq d_n$. To each job $j \in \{1, ..., n\}$ is associated a processing time p_j, a consumption b_j and a due date d_j. The life duration of the product after opening is equal to T and the volume of one vial is equal to V. We assume always w.l.o.g. that $p_j < T$ and $b_j < V$, $\forall j$, $1 \leq j \leq n$. The number of vials is not limited but supposed to be bounded by n. We denote by C_j the completion time of j, L_j the lateness defined by $L_j = C_j - d_j$. The maximum lateness is defined by $L_{max} = \max_{1 \leq j \leq n} L_j$. We assume that the maximum lateness is bounded by Q. Minimizing the quantity of lost raw materials is equivalent to minimize the number of vials that are opened. Therefore, the problem is a mixed between a scheduling problem and a two-constraint bin packing problem. Without due dates (or with extremely large due dates), the problem is a bin packing problem. For this reason, the problem is clearly NP-hard. With a huge T and a huge V, the problem is the trivial single machine problem with the L_{max} minimization. In the following, we call a "bin", the set of jobs performed with the same vial.

Example: We consider a set of six jobs with the following data and $T = 10$ and $V = 10$.

j	1	2	3	4	5	6
p_j	3	4	4	5	3	1
b_j	1	2	5	3	1	4
d_j	7	9	11	13	14	16

The schedule $(1, 3, 5, 2, 4, 6)$ is represented in Fig. 1. In this two dimensional Gantt chart, a job j is represented by a rectangle with the duration p_j on the x-axis and the consumption b_j on the y-axis. Jobs of the same bin are connected by the north-west corner of the rectangle. The first job of a bin is put on the x-axis. In Fig. 1, one can see that 1 and 2 are the first jobs of the bins $(1, 3, 5)$ and $(2, 4, 6)$. The maximum lateness of this sequence is equal to $L_{max} = \max(-4, 5, -4, 6, -4, 4) = 6$ and this schedule requires 2 bins.

We denote by $u_k \in \{0, 1\}$ a boolean variable equal to 1 if bin k is used, and 0 otherwise and by $x_{j,k} \in \{0, 1\}$ a boolean variable equal to 1 if job j is assigned to bin k, and 0 otherwise. Also, H_V denotes a very large (high value) constant in order to determine a standard *big-M* constraint in ILP modeling. The problem can be formalized as follows.

$$\text{MIN} \sum_{k=1}^{n} u_k$$

s.t.

$$\begin{cases} \sum_{k=1}^{n} x_{j,k} = 1, \forall j \in \{1, ..., n\} & (1) \\ \sum_{j=1}^{n} p_j x_{j,k} \leq T u_k, \forall k \in \{1, ..., n\} & (2) \\ \sum_{j=1}^{n} b_j x_{j,k} \leq V u_k, \forall k \in \{1, ..., n\} & (3) \\ \sum_{h=1}^{k-1} \sum_{i=1}^{n} p_i x_{i,h} + \sum_{i=1}^{j} p_i x_{i,k} \leq d_j + Q + H_V(1 - x_{j,k}), & \\ \qquad \forall j \in \{1, ..., n\}, \forall k \in \{1, ..., n\} & (4) \\ u_{k+1} \leq u_k, \forall k \in \{1, ..., n\} & (5) \end{cases}$$

Constraints (1) ensure that each job is performed by using one vial (is assigned to one bin). Constraints (2) and (3) correspond to the temporal and capacity limits. Constraints (4) suppose that job j is in bin k and correspond to the bound on the L_{max}: $\sum_{h=1}^{k-1} \sum_{i=1}^{n} p_i x_{i,h}$ is the completion time of the $k-1^{th}$ bin, $\sum_{i=1}^{j} p_i x_{i,k}$ is the completion time of job j in its bin (remember that the jobs are numbered in EDD order). Constraints (5) ensure that the bins are used in their index increasing order.

2 Solution approaches

2.1 Recovering beam search algorithm

The *Beam Search* algorithm is a truncated branch-and-bound method where a subset of w nodes at each level are selected for branching. w is called the *beam width*. This method was first proposed in (Ow and Morton 1988). For the selection of nodes, each node is evaluated by a combination of a lower bound (LB) and an upper bound (UB), generally a weighted sum $WS = (1 - \alpha)LB + \alpha UB$. Because the selected nodes are not necessarily the bests at a given level of the tree, among the set of possible nodes of a pure branch-and-bound algorithm, a recovering phase is applied in the *Recovering Beam Search* algorithm (RBS). The aim of this phase is to recover from wrong decisions jumping to a better node at the same level of the search tree. For a detailed description of RBS we refer to (Della Croce *et. al.* 2004). A node σ of the tree is defined by a partial sequence of jobs $S(\sigma)$, a

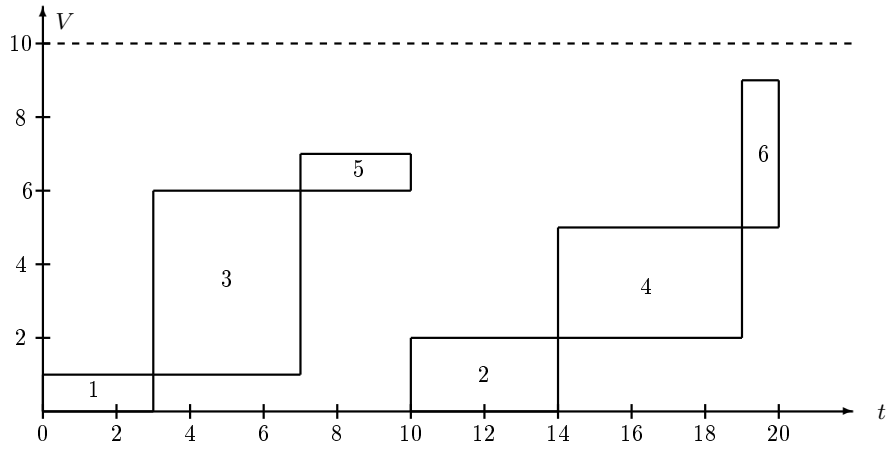


Fig. 1. Gantt chart of schedule $(1, 3, 5, 2, 4, 6)$

list of unscheduled jobs $\overline{S}(\sigma)$, a lower bound $LB(\sigma)$, and an upper bound $UB(\sigma)$. At the root node, the initial sequence of unscheduled jobs is determined as follows. Starting from EDD sequence, a steepest descent algorithm is used to reduce the number of bins, without violating the constraint on the L_{max}. The initial sequence which is obtained is called INIT. Alg. 1 and Alg. 2 describe the lower bound and upper bound computation. We denote by $Bin(S)$ a function that computes in $O(n)$ the number of bins used by a sequence S and the sum of jobs processing time and of jobs consumption in the last bin (respectively called $RestP$ and $RestB$). $u//v$ stands for the concatenation of u and v.

Algorithm 1 $LB(\sigma)$

$NbBins = Bin(S(\sigma)) - 1$
$SumP = RestP + \sum_{j \in \overline{S}(\sigma)} p_j$
$SumB = RestB + \sum_{j \in \overline{S}(\sigma)} b_j$
$NbBins = NbBins + \max(\lceil SumP/T \rceil, \lceil SumB/V \rceil)$
Return($NbBins$)

Algorithm 2 $UB(\sigma)$

$S'(\sigma) = S(\sigma)//\overline{S}(\sigma)$
$NbBins = Bin(S'(\sigma))$
Return($NbBins$)

The evaluation of a node is given by $WS(\sigma) = (1-\alpha)LB(\sigma) + \alpha UB(\sigma)$. The recovering phase is composed by two types of neighborhood called SWAP and EBSR (extraction and backward shift reinsertion) proposed in (Della Croce *et. al. 1* 2004).

2.2 Matheuristic algorithm

A matheuristic procedure (Della Croce *et. al.* 2013) can be seen as a local search approach for MIPs, expecially suited for $0-1$ variables, using a generalization of the k-exchange neighborhood. Consider a general MIP ($\min c^T X$ subject to $AX \leq b$, $X \in$

$\{0,1\}$), where $X^T = (x_1, x_2, .., x_n)$ is a vector of n variables of the problem and $\bar{X}^T = (\bar{x_1}, \bar{x_2}, .., \bar{x_n})$ is a feasible solution to the MIP. If this is the case, it is always possible to define a subset S of a defined size of variables indices $\{1, 2, .., n\}$. The neighborhood $N(\bar{X})$ consists of all solutions of the MIP where the j^{th} variable is equal to the value of the j^{th} variable in $\bar{X}$ for all $j \notin S$, namely $N(\bar{X}) = \{X \mid x_j = \bar{x}_j,\ \forall j \notin S\}$. The resulting neighborhoods $N(\bar{X})$ can then be searched for an improving solution using a MIP-solver both optimally or approximately. The main idea stems in representing the MIP as a permutation problem where variables belonging to the current solution are partitioned into two sets. A first set ($\bar{X}/S$) is then reoptimized by means of a MILP solver generating a permutated assignment while variables in the second set (S) keep the same assignment as in the current solution. For the considered problem, the incumbent solution returned by the RBS algorithm induces correspondingly a sequence of the bins where we assume that γ bins are used. The neighborhood exploration works as follows. Starting with the first bin, consider the r-th bin in the sequence along with bins $r+1$, $r+2$,..., $r+H-1$ with item set $S = S_r \cup S_{r+1} \cup ... \cup S_{r+H-1}$. Solve the problem of rescheduling the items in S so that $w(S_{r+H-1}) = \sum w_i : i \in S_{r+H-1}$ is minimized where we use $w_i = \max\{v_i, p_i\}$ (this performance measure has been selected experimentally). The rationale is to empty as much as possible bin $r+H-1$. If a (sub)sequence with $w(S_{r+H-1}) = 0$ is obtained, then one bin is saved, γ is reduced by one unit and the process can restart with $r = 1$. Alternatively, the new subsequence is kept anyway as the space of bin S_{r+H-1} has been optimized and will be used in the next iterate. The approach is then iterated for $r = 1, 2, ..., \gamma - H + 1$. Whenever $r = \gamma - H + 1$ is reached, the process restarts with $r = 1$ until a time limit is exceeded. The problem of rescheduling the items in S is done by solving by means of an ILP solver that adapts the model presented to above in order to take into account the fact that bins $1, ...r-1$ and $r+H, ..., \gamma$ are not rescheduled and that the objective is to minimize the weight of the items assigned to bin $r+H-1$.

3 Computational experiments

Detailed computational experiments will be presented at the Conference and relate to the bin packing instances considered in (Capara and Toth 2001) modified in order to take into account also the sequencing part of the problem. Here, we simply stress that the proposed approach combining RBS and MH reaches in limited time good quality solutions being able to improve also some of the benchmarks results on the original bin packing instances of (Capara and Toth 2001).

References

J-C Billaut, 2011, "New scheduling problems with perishable raw materials constraints", *16th IEEE Conference on Emerging Technologies and Factory Automation*, Vol. 01, pp. 1-5.

A. Caprara and P. Toth, 2001, "Lower bounds and algorithms for the 2-dimensional vector packing problem", *Discrete Applied Mathematics*, Vol. 111, pp. 231-262.

F. Della Croce, M. Ghirardi and R. Tadei, 2004, "Recovering Beam Search: enhancing the beam search approach for combinatorial optimization problems", *Journal of Heuristics*, Vol. 10, pp. 89-104.

F. Della Croce, A. Grosso and F. Salassa, 2013. "Matheuristics: Embedding MILP Solvers into Heuristic Algorithms for Combinatorial Optimization Problems", in Heuristics, Theory and Applications, P. Siarry (ed.), Nova Publishers, 31-52.

A. Mazier, J-C. Billaut and J-F. Tournamille, 2010. "Scheduling preparation of doses for a chemotherapy service", *Annals of Operations Research*, Vol. 178(1), pp. 145-154.

P. S. Ow and T. E. Morton, 1988,"Filtered beam search in scheduling", *International Journal of Production Research*, Vol 26, pp. 297–307.

THE COLD ROLLING MILL SCHEDULING PROBLEM

FAYEZ F. BOCTOR
Centre Interuniversitaire de Recherche sur le Réseau d'entreprises, la Logistique et le Transport
Faculté des sciences de l'administration, Université Laval, Québec, Canada.

Keywords: job scheduling, Heuristics.

1. INTRODUCTION

This paper deals with the problem of scheduling rolling jobs on a single rolling mill which hereafter will be called the *Cold Rolling-Mill Scheduling Problem* (CRMSP). The problem studied is a real one that has been observed in a Canadian aluminium rolling facility where the usual practice is to construct an operations schedule for the coming five to seven days on a sliding horizon basis. This plan is updated every two or three days to reschedule the remaining jobs of the previous plan along with new ones that are expected to arrive during the following five to seven working days. The arrival date, the due date, the processing time, the weight, the final width and the final thickness of the aluminium coils to be rolled are known and deterministic. The scheduling objective is to minimize the sum of setup costs, tardiness penalties and inventory holding costs. A setup is needed whenever the width of the coil to be rolled is larger than the width of the previous one as the edges of the rolled coil leave marks and scratches on the rollers. In this case, the rollers should be grinded and polished before rolling the larger job. The work-in-process holding cost depends on the alloy composition, the weight of the coil to be processed and its flow time in the shop.

In the studied industrial setting, the setup cost is estimated at $300, the setup time is about 1.5 hours and the tardiness penalty per hour is set at $30 per 1000 pounds. The average inventory holding cost per 1000 pounds and per hour is $0,025, the width of jobs varies between 40 and 62 inches and the job processing time is from 1 to 5 hours.

2. LITERATURE REVIEW

Very few published papers deal with the rolling mill scheduling problem and all but one address the hot not the cold mill scheduling problem. The objective in this case is to synchronize the operations of the furnaces or the smelters with the operations of the rolling mill facility. This objective is usually expressed as maximizing the throughput of the milling facility or minimizing the make span of processing the set of rolling jobs. As underlined by Nicholls (1994), if the rolling mill facility is not able to handle all the required jobs in time, the entire plant can be disrupted causing substantial cost increase.

The sole paper that deals with the cold rolling-mill scheduling problem is by Mayrand *et al* (1995) who propose a non-linear formulation and an adaptation of the genetic algorithm to solve the problem. The cost elements of the objective function are discounted in function of the arrival date of each job. They do not consider due dates as either a soft or a hard constraint. However, there is no guarantee that the genetic algorithm they propose will find a feasible schedule even if such a schedule exists.

This current paper rather, deals with the cold rolling mill scheduling problem (CRMSP) where all jobs should be processed and there is no need to synchronize the mill operations with the smelter. The paper provides a new and linear formulation of the problem considering due dates explicitly. It also provides a Hybrid Greedy Randomized Adaptive Sampling Procedure (HGRASP) to solve it. Using randomly generated problem instances having the same characteristics of the real-life problem, it will be shown that the proposed heuristics perform remarkably well.

3. MATHEMATICAL FORMULATION

This section provides a new and linear formulation of the CRMSP using the following notation:

N number of jobs to be scheduled
j job index; j=1,..., N
S cost of one set-up
s time required to perform one set-up
H a large number
h_j unit inventory holding cost per unit of time and per unit of weight of job j
α earliest date where the rolling mill is available to process the considered set of jobs
a_j arrival date of job j
d_j due date of job j
w_j total weight of job j (e.g. in pounds)
L_j slab width of job j (e.g. in inches)
L the largest width that can be processed at the beginning of the planning horizon without any prior set-up
e_j thickness of job j (e.g. in inches)
p_j processing time of job j (e.g. in hours)
P_j tardiness penalty per unit of time and unit of weight for job j
t_j starting date of job j
r_j tardiness of job j; $r_j = max\ (0, t_j+p_j-d_j)$
X_{jp} binary variable equals 1 if job j is to be in position p in the execution sequence;
Y_p binary variable equals 1 if a setup is needed before the job in position p

Due dates will be considered as soft constraints. A penalty P_j per unit of job weight and for each tardiness time unit is added to the total cost. To formulate the problem we add a dummy job, numbered 0, with $L_0=L$, $t_0= a_0= \alpha$, $p_0=0$, $d_0= \infty$ and assign it to position 0 in the sequence (i.e.: we set $X_{00}=1$). Using the above given notation our cold rolling mill scheduling problem (CRMSP) can be modelled as follows:

Find: $t_j \geq a_j;\ r_j \geq 0, Y_p \in \{0,1\}$ and $X_{jp} \in \{0,1\};\ j = 1,..N, P = 1,..,N,$ which

Minimize: $\sum_{p=1}^{N} SY_p + \sum_{j=1}^{N}\{h_j w_j\left(t_j + p_j - a_j\right) + P_j w_j r_j\}$ (1)

Subject to: $\sum_{p=1}^{N} X_{jp} = 1;\ j = 1,..,N$ (2)

$\sum_{j=1}^{N} X_{jp} = 1;\ p = 1,..,N$ (3)

$t_i - t_j - p_j - sY_p \geq H\left(X_{j,p-1} + X_{ip} - 2\right);$
$j = 0,..,N, i = 1,..,N, i \neq j, p = 1,..,N$ (4)

$HY_p \geq H\left(X_{j,p-1} + X_{ip} - 2\right) + \left(L_i - L_j\right);$
$j = 1,..,N, i = 1,..,N, i \neq j, p = 1,..,N$ (5)

$r_j \geq t_j + p_j - d_j;\ j = 1,..,N$ (6)

The first term of the objective function (1) is the sum of setup costs; the second term is the total inventory holding cost, and the third term gives the overall tardiness penalty. Constraints (2) state that each job has one and only one position in the execution sequence while constraints (3) make sure that there is only one job in each position. Constraints (4) imply that, if job j immediately precedes job i, then the starting time of i is greater than or equal to the finish time of j plus the setup time s if a setup is necessary. Constraints (5) set the value of setup variable Y_p to 1 if the job in position p is larger than the job in position p-1. Finally constraints (6) determine the tardiness r_j of each job j.

This model contains $2N$ continuous variables, $N(N+1)$ binary variables and $N(2N^2-N+1)$ linear constraints. It can be solved using available MIP codes for up to about 10 jobs only. However, real-life instances contain up to 50 jobs over a horizon of 5 days.

4. OPTIMUM SOLUTION AND EXECUTION DATES FOR A GIVEN SEQUENCE

The heuristic proposed in this paper enumerate some production sequences and for each sequence determine the corresponding optimal execution dates and costs. Let α be the earliest date where the mill is available and assume that jobs are numbered according to the given execution sequence. Then the earliest start times can be determined by:

$$t_1 = max\ (a_1\ ,\ \alpha+\theta_1 s), \quad (7)$$

$$t_j = max\ (a_j\ ,\ t_{j-1} + p_{j-1} + \theta_j s);\ j=2,...,N, \quad (8)$$

$$r_j = max\ (0,\ t_j + p_{j-1} - d_j);\ j=1,...,N. \quad (9)$$

Where: θ_1 =1 if $L_1>L$ and θ_1=0 otherwise

θ_j =1 if $L_j>L_{j-1}$ and θ_j=0 otherwise

The minimal total cost corresponding to this sequence σ is:

$$C_\sigma = S\sum_{j=1}^{N}\theta_j + \sum_{j=1}^{N} h_j w_j (t_j - a_j\ ) + \sum_{j=1}^{N} P_j r_j \quad (10)$$

This cost is the smallest possible (minimum) cost for the given sequence σ as once the sequence is chosen the number of setups is fixed, the corresponding setup cost is determined and cannot be reduced. Also, using the earliest execution dates minimizes both the sum of the inventory holding costs and the sum of tardiness penalties.

5. THE HYBRID GREEDY RANDOMIZED ADAPTIVE SAMPLING PROCEDURE

The proposed heuristic enumerates a number of production sequences and retains the one producing the smallest cost. Each sequence is obtained by a construction heuristic which starts at time $t=\alpha$ (the earliest time where the rolling mill is available), attributes to each of the jobs schedulable at time t a score, denoted s_j, and randomly choose among the three jobs having the highest score the one to schedule. Then we move to the time moment where this job should finish and choose the next one; and so on. The score is a weighted sum of three of job characteristics: time slack, processing time and width.

To choose among the three jobs having the highest score, we use the standard roulette wheel scheme. We arrange the 3 jobs in the descending order of their score, attribute to each one a probability f_j of being selected, which equals its score divided by the sum of the three scores, and then draw a random number x from a uniform distribution between 0 and 1. The job with the highest probability is then selected if $x \leq f_1$, the second job is selected if $f_1 < x \leq f_1+f_2$, and the third one is selected otherwise.

This proposed sequence construction procedure is repeated G times producing G solution. The optimal execution dates and cost for each sequence is determined (as shown in Section 4) and the best of the G sampled sequences is then retained.

Two improvement procedures, $I1$ and $I2$, are then applied. These improvement procedures are hill climbing procedures. The difference between these two procedures is their searched neighbourhood. The neighbourhood of $I1$ is all neighbour sequences where only one job is moved to a different position. The neighbourhood of $I2$ is all the neighbour sequences where one pair of jobs exchanges their position in the sequence.

6. PERFORMANC EVALUATION

One hundred test instances were randomly generated and solved. These instances have the same characteristics as the real-life problem observed in the Aluminium rolling facility but with only 15 jobs to process. Table 1 presents parameters values used to generate the test instances.

Table 1: Parameters of the randomly generated test-instances

Parameter	Values	Parameter	Values
Set up cost S	300	Job width L_j	41,45,49,53,57,61
Setup time s	1.5	Job weight w_j	20,25,30,..,225
Initial width L	57	Job arrival date a_j	0,6,12,18
Tardiness penalty P_j	30	Job due date d_j	6,12,18,24
Inventory holding cost h_j	0.025	Processing time p_j	1,2,3,4,5

We tried to solve the test problems to optimality using the MIP code GUROBI 5.0.1 and a computer equipped with a 3.50 GHz, 8 cores Intel Xeon processor. After 24 hours of computation the first problem was not solved. Thus it was decided to put a limit of 3600 seconds (one hour) on the run time. No problem was solved to optimality within this time limit. Solutions obtained after one hour was better than the best solutions obtained by the tested heuristic for only 21 problems. The average deviation of the best solution obtained by GUROBI from the best obtained solution was 3.39%.

The 100 test-instances were solved using the Hybrid greedy randomized adaptive sampling procedure with G (the number of sampled solutions) equals to 1000, 2000, and 5000. The best sampled solution is then improved by applying the two improvement procedure $I1$ and $I2$. Table 2 summarizes the results obtained. From this table we can see that increasing G decreases the percentage deviation from the best solution found. However, the computation time also significantly increases. The table also shows that the HGRASP with G=5000 and followed by the two improvement procedures gave the best results as expected.

Table 2*:* Summary of the obtained results

	GUROBI	HGRASP with*		
		G=1000	G=2000	G=5000
Average percentage deviation from the best found solution	3.39%	2.19%	2.13%	1.70%
Maximal percentage deviation from the best found solution	14.84%	13.24%	18.53%	13.17%
Average computation time (seconds)	3600	2.00	3.84	9.53
Number of times the best solution is found	21	46	49	55
Number of times is the only method to find the best solution	21	10	7	11

* Only the best sampled solution is improved by applying $I1$ and $I2$.

7. OTHER SOLUTION METHODS

Two other solution methods were developed within this research project to solve the CRMSP. A detailed description of these methods and the results they produced are given in Boctor (2014).

Acknowledgement

This research work was partially supported by the Canadian Natural Sciences and Engineering Research Council (NSERC) under Grant OPG0036509. This support is gratefully acknowledged.

REFERENCES

Mayrand, E, P Lefrançois, O Kettani, and M-H Jobin, 1995. A Genetic search algorithm to optimize job sequencing under technological constraint in a rolling-mill facility. *Operations Research Spektrum*, 17, 2/3, 183-191.

Nicholls, M G, 1994. Optimizing the operations of an ingot mill in an aluminium smelter. *Journal of the Operational Research Society*, 45, 9, 987-998.

Boctor, F F, 2014. Methods to solve the cold rolling mill scheduling problem. Document de travail, FSA, Université Laval.

Bounds on single processor scheduling with time restrictions

Oliver Braun[1], Fan Chung[2], and Ron Graham[2]

[1] Environmental Campus Birkenfeld, Trier University of Applied Sciences, Germany
e-mail: o.braun@umwelt-campus.de

[2] University of California, San Diego, USA
e-mail: fan@ucsd.edu, graham@ucsd.edu

Keywords: scheduling, worst-case analysis, time restrictions.

1. Problem description

We are initially given a set $S = \{S_1, S_2, \ldots, S_n\}$ of jobs. Each job S_i has associated with it a length s_i where we assume $0 \leq s_i \leq 1$. The jobs are all processed sequentially on a single processor. Only one job can be worked on at any point in time. A job S_i will be processed during a (semi-open) time interval $[\alpha, \alpha + s_i)$ for some $\alpha \geq 0$.

Given some permutation π of S, say $\pi(S) = T = (T_1, T_2, \ldots, T_n)$ with corresponding job lengths $(t_1, t_2, \ldots, t_n)$, the jobs are placed sequentially on the real line as follows. The initial job T_1 in the list T begins at time 0 and finishes at time t_1. In general, T_{i+1} begins as soon as T_i is completed, provided the following constraint is always observed [1].

$$\begin{gathered}\text{For every real } x \geq 0, \text{ the unit interval} \\ [x, x+1) \text{ can intersect at most } B \text{ jobs.}\end{gathered} \tag{1}$$

Constraint (1) reflects the condition that each job needs one of B additional ressources for being processed and that a ressource has to be renewed after the processing of a job has been finished. The preceding procedure results in a unique placement (or *schedule*) of the jobs on the real line. We define the *finishing time* $\tau(T)$ to be the time at which the last job T_n is finished. A natural goal might be for a given job set S, to find those permutations $T = \pi(S)$ which minimize the finishing time $\tau(T)$.

2. Worst-case analysis of the *List Scheduling* heuristic

In [1] it has been shown that the problem is NP-hard in general. (The authors polynomially reduce the NP-hard problem PARTITION (see [2]) to a special case of our scheduling problem.) Let us denote by $\tau_w(S)$ the largest possible finishing time for any permutation of S, and let $\tau_o(S)$ denote the optimal (i.e., the shortest possible) finishing time for any

permutation of S. We are interested in seeing how much worse $\tau_w(S)$ can be compared to $\tau_o(S)$. The following bounds from [1] are similar in spirit to some of the bounds in the very early *List scheduling heuristic* and *LPT heuristic* literature ([3], [4]).

Theorem 1 *For $B = 2$ and any set S,*

$$\tau_w(S) - \frac{4}{3}\tau_o(S) \leq 1.$$

The factor $\frac{4}{3}$ is best possible.

Theorem 2 *For $B \geq 3$ and any set S,*

$$\tau_w(S) - \left(2 - \frac{1}{B-1}\right)\tau_o(S) \leq 3.$$

The factor $2 - \frac{1}{B-1}$ is best possible.

3. Analysis of the LPT heuristic

For this heuristic, we rearrange our set S of jobs into *decreasing* order to form the permutation $L = (L_1, L_2, \ldots, L_n)$ where l_i denotes the length of L_i and $l_i \geq l_{i+1}$ for $1 \leq i \leq n-1$. As usual (see [1]), we let $s(l_i)$ denote the *starting* time of L_i when L is scheduled, and we let $f(L_i)$ denote the corresponding *finishing* time. We write $\tau(L) = \tau_{LPT}(S)$ for the makespan of an LPT-schedule. We start with the following observation.

Lemma 1 *If the jobs are ordered according to the LPT rule, then we have:*

$$s(L_i) = f(L_{i-2}) + 1, \ i \geq 3$$

Proof: It follows from (1) that

$$s(L_i) \geq f(L_{i-2}) + 1, \ 2 \leq i \leq n. \tag{2}$$

In fact, we claim that (2) holds with *equality* for all i if the jobs are given in LPT order. This is certainly true for $i = 2$ (since $l_2 \leq 1$). Assume that it holds for some value $i \geq 3$. The only reason that we could have $s(L_i) > 1 + f(L_{i-2})$ is if

$$f(L_{i-1}) > f(L_{i-2}) + 1. \tag{3}$$

But we know by induction that $f(L_{i-1}) = s(L_{i-1}) + l_{i-1} = f(L_{i-3}) + 1 + l_{i-1}$. Hence, by (3), $f(L_{i-3}) + 1 + l_{i-1} > f(L_{i-2}) + 1$, i.e., $f(L_{i-3}) + l_{i-1} > f(L_{i-2}) = s(L_{i-2}) + l_{i-2}$. However, $s(L_{i-2}) \geq f(L_{i-3})$. Therefore, $s(L_{i-2}) + l_{i-1} > s(L_{i-2}) + l_{i-2}$, or $l_{i-1} > l_{i-2}$, which is a contradiction. Thus we have $s(L_i) = 1 + f(L_{i-2})$ for all i.

Theorem 3 *For any set S,*

$$\tau(L) - \tau_o(S) \leq \begin{cases} \frac{1}{2}(1 + l_1 - l_{n-1} - l_n) & \text{if } n \text{ is odd,} \\ \frac{1}{2}(l_1 + l_2 - l_{n-1} - l_n) & \text{if } n \text{ is even.} \end{cases}$$

Proof: We know from [1] that the optimal schedule for S satisfies

$$\begin{aligned} \tau_o(S) &\geq \frac{1}{2}\big(s_1 + s_n + \sum_{i=1}^{n} s_i + n - 2\big) \\ &\geq \frac{1}{2}\big(l_{n-1} + l_n + \sum_{i=1}^{n} l_i + n - 2\big) \end{aligned} \tag{4}$$

(note that l_{n-1} and l_n are the lengths of the two smallest jobs in the LPT order).

Now we consider two cases:

(i) $n = 2m + 1$. From Lemma 1 we see that

$$\tau(L) = l_1 + 1 + l_3 + 1 + \ldots + l_{2m-1} + 1 + l_{2m+1} = \sum_{i=1}^{m} l_{2i-1} + m.$$

Because of $l_i \geq l_{i+1}$ for $1 \leq i \leq n-1$, we have consequently

$$\tau(L) \leq l_1 + 1 + l_2 + 1 + \ldots + l_{2m-2} + 1 + l_{2m} = l_1 + \sum_{i=1}^{m-1} l_{2i} + m.$$

Thus,

$$\begin{aligned} \tau(L) &\leq \frac{1}{2}\big(l_1 + \sum_{i=1}^{2m+1} l_i + 2m\big) \\ &= \frac{1}{2}\big(l_1 + \sum_{i=1}^{n} l_i + n - 1\big). \end{aligned}$$

Hence, by (4),

$$\tau(L) - \tau_o(S) \leq \frac{1}{2}\big(1 + l_1 - l_{n-1} - l_n\big).$$

(ii) $n = 2m$. From Lemma 1 we see that

$$\tau(L) = l_1 + 1 + l_2 + 1 + l_4 + 1 \ldots + l_{2m-2} + 1 + l_{2m} = l_1 + \sum_{i=1}^{m} l_{2i} + m - 1.$$

Because of $l_i \geq l_{i+1}$ for $1 \leq i \leq n-1$, we have consequently

$$\tau(L) \leq l_1 + 1 + l_2 + 1 + l_3 + 1 + \ldots + l_{2m-3} + 1 + l_{2m-1} = l_2 + \sum_{i=1}^{m} l_{2i-1} + m - 1.$$

Thus,

$$\begin{aligned}\tau(L) &\leq \frac{1}{2}\big(l_1 + l_2 + \sum_{i=1}^{2m} l_i + 2m - 2\big) \\ &= \frac{1}{2}\big(l_1 + l_2 + \sum_{i=1}^{n} l_i + n - 2\big).\end{aligned}$$

Hence, by (4),

$$\tau(L) - \tau_o(S) \leq \frac{1}{2}\big(l_1 + l_2 - l_{n-1} - l_n\big).$$

It follows immediately that LPT-schedules are at most one unit interval longer than optimal schedules. We give the following example that shows that the bound is tight. Consider the set S consisting of t jobs of length 1 and t zero-jobs for some integer t. We denote a job of length 1 by 1 and a zero-job by 0. The optimal permutation $0[1]^t[0]^{t-1}$ has finishing time $\tau_o(S) = (3t-2)/2$. On the other hand, the *LPT*-order 1^t0^t has a finishing time $\tau(L) = 3t/2$. Thus, $\tau(L) - \tau_o(S) = 1$.

References

[1] Braun, O., Chung, F., Graham, R.L. (2013). Single processor scheduling with time restrictions. *Journal of Scheduling*, DOI 10.1007/s10951-013-0342-0.

[2] Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman.

[3] Graham, R. L. (1966). Bounds on certain multiprocessing anomalies. *Bell System Technical Journal, 45*, 1563–1581.

[4] Graham, R. L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics, 17*, 416–429.

Criticalness and the threat of project tasks

Brozová, Helena[1], Dvorak, Petr[2], and Subrt, Tomas[1]

[1]CULS Prague, DSI FEM, Czech Republic
e-mail: brozova@pef.czu.cz, subrt@pef.czu.cz

[2]CULS Prague, DICT, Czech Republic
e-mail: dvorakpetr@oikt.czu.cz

Keywords: Task criticalness, threatening task, multiple criteria decision making.

1. Introduction

The project management is a tool providing successful realization of a project within planned time, planned costs and planned quality. To ensure this it is necessary to analyse critical and threatening factors both explicitly and implicitly existing in the project.

The principle of the project management is expressed in the form of a project magic triangle (triple imperative) where the dimension of costs, time and quality towards the project objective are monitored (Kerzner, 1979, Warburton, 2011, PMI, 2013). This magic triangle expresses the dynamics of the project management: it is possible to reduce costs, however, at the expense of time and quality, or it is possible to reduce time but at the expense of costs and decreasing quality or finally, it is possible to improve the quality but to increase time and costs. Unfortunately, the objectives of a project can never be reached with minimal costs, in minimal time and with the highest quality possible (Kerzner, 1979, Warburton, 2011). Liberatore and Pollack-Johnson (2012) show the continuous model for maximization the quality subject to time and cost limits.

The riskiness, threat and criticalness of project activities are not only given by the surroundings and the environment of the project but also by the internal arrangement and structure of the project.

Quantitative estimation serves for the evaluation of the task criticalness. Although such quantitative estimation of the project is performed in a pre-project phase, generally only traditional time and resource analysis of the project is provided. Primarily, the activities are evaluated from a time perspective only as critical or uncritical activities, but many other quantitative characteristics can be used. Such a multiple attribute analysis enables not to ignore the impact of time-uncritical activities which may have the high criticalness from many different reasons (Brozova et al, 2013).

On the other side, more attention is paid to the analysis of the risks which can occur during the project realization and result in the project failure (Wideman, 1992). The risk management covers a number of different techniques and approaches which have been derived in order to reduce the risk of a project and its partial activities (Wideman, 1992). The risk management is the identification, assessment and reduction of risks (Gonen, 2012). Sadeh and Zwikael (2012) show that in order to improve project success organizations have to use a proper measurement method for riskiness evaluation. Risks can come from uncertainty in quantitative attributes and from qualitative and intangible aspects, which often cover also human and social elements. That is the reason why the subjective judgement methods or fuzzy approach are used for the evaluation of riskiness of the project (Gonen, 2012).

According to our point of view, the criticalness, the riskiness and the threat of the project tasks should be evaluated and managed. Threatening tasks can lead to the project failure in all three aspects of the project magic triangle. The task threat is more qualitative than the task criticalness; however, it is based on similar parameters, mainly duration, cost and quality. Therefore the evaluation of the activities threat should be based on the fuzzy evaluation from the perspective of the project magic triangle aspects.

The main aim of the paper is to compare the qualitative and quantitative approach analysing threats and criticalness of the project tasks. We suggest the application of the fuzzy linguistic system into the qualitative (soft) estimation of the threat of the project tasks and we use multiple attribute approaches in the quantitative (hard) evaluation of the activities criticalness. We compare both approaches and discuss their advantages and disadvantages.

2. Threatening tasks

The suggested overall evaluation of the threat of the project activities is based on experts' knowledge and experts' experience of characters of tasks. This evaluation method is based on the soft system approach (Checkland, Scholes, 1990) because many factors influencing the project success have social or human roots or are based on uncertainty, probability and possibility. Such managerial evaluation of the task threatening can be made by expressing linguistic uncertainty using linguistic terms

and fuzzy scales. A fuzzy measure is a subjective scale for the degrees of fuzziness and is suitable in analysing human subjective judgement (Ross, 2010, Wang and Klir, 1992). Therefore, we define the fuzzy linguistic system with the linguistic state variables describing the level of activity threat, their values are words, and meanings of these words are fuzzy sets. The evaluation of the threat is based on a fuzzy conversion, which corresponds to the conversion of linguistic labels into fuzzy numbers. In this study we created a non uniform six point fuzzy scale allowing realistic subjective evaluation of the task hazard where uncertainty decreases towards the extreme values.

Task can be (timely, costly or qualitatively)	
not at all hazardous	(0; 0; 0; 0.1)
usually not hazardous	(0; 0.1; 0.2; 0.3)
rather not hazardous	(0.2; 0.3; 0.4; 0.6)
rather hazardous	(0.4; 0.6; 0.7; 0.8)
usually hazardous	(0.7; 0.8; 0.9; 1)
always hazardous	(0.9; 1; 1; 1)

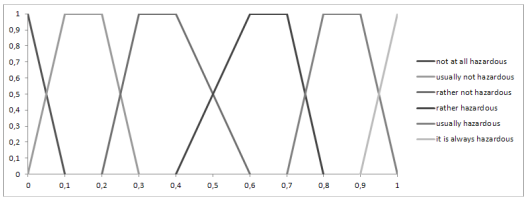


Figure 1. Linguistic fuzzy scale for task hazard evaluation

Partial evaluations of the tasks are aggregated using a fuzzy logic system representing a knowledge system (its simplest form uses fuzzy sum). For the overall evaluation is used a non uniform five point fuzzy scale with uncertainty increases towards the evaluation Extremely threatening task values.

Task can be	
Non-threatening	(0; 0; 0.05; 0.15)
Weakly threatening	(0.05; 0.15; 0.25; 0.35)
Rather threatening	(0.25; 0.35; 0.5; 0.6)
Strongly threatening	(0.5; 0.6; 0.75; 0.85)
Extremely threatening	(0.75; 0.85; 1; 1)

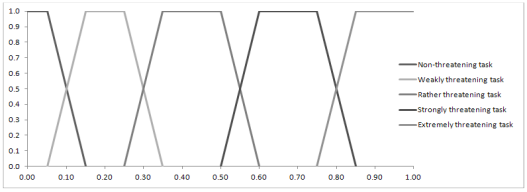


Figure 2. Linguistic fuzzy scale for overall task threat evaluation

The application of the suggested fuzzy system enables to obtain an approximative overall evaluation of the task threat for all project tasks.

3. Task criticalness

A hard system approach to the analysis of the task in the relation to the project success is based on the crisp exact quantitative evaluation of the task. Brozova et al (2013) suggested to provide the overall evaluation of the task criticalness without soft knowledge about character of the tasks.

The estimation of the overall criticalness of the project activities is based on the multiple attributes decision making method using five indicators of the criticalness as probability that the activity will lie on critical path, time duration, time slack, cost, and work. The weights of these indicators are evaluated by the decision maker. The values of these indicators are then converted into interval $\langle 0,1 \rangle$, the value 0 corresponds to the lowest partial criticalness and the value 1 to the highest criticalness, and used as the input for the Simple additive weighted mmethod (Hwang, Yoon, 1981) using formula

$$C_I = v_1 \frac{p_I - \min\limits_K p_K}{\max\limits_K p_K - \min\limits_K p_K} + v_2 \frac{t_I - \min\limits_K t_K}{\max\limits_K t_K - \min\limits_K t_K} + v_3 \frac{s_I - \max\limits_K s_K}{\min\limits_K s_K - \max\limits_K s_K} + v_4 \frac{c_I - \min\limits_K c_K}{\max\limits_K c_K - \min\limits_K c_K} + v_5 \frac{w_I - \min\limits_K w_K}{\max\limits_K w_K - \min\limits_K w_K} \qquad (1)$$

where C_I is the global evaluation of the activity criticalness, p_I is the evaluation of probability of the critical path, t_I is the duration of the activity, s_I is the time slack of the activity, c_I is the cost of the activity, v_I is the work amount of the activity, and $v_1,\ldots,v_5$ are the weights of the components of criticalness.

The probability that the activity lies on critical path is related to the project structure, i.e. these probabilities of the predecessors and the numbers of their successors and calculated as

$$p_1 = 1 \text{ and } p_i = \sum_{j \text{ predecessor of } i} \frac{p_j}{h_j},\ i \neq 1 \qquad (2)$$

where p_1, p_i, p_j are the probabilities that the activities 1, i, j will be on the critical path, and h_j is the number of activities following activity j, (activity 1 is the starting task).

4. Example of the evaluation of the tasks - Tender for the purchase of software

The project shows the tasks of the public tender for the acquisition of software in simplified sub-limit mode. The tender starts with preparing the documentation. The first aggregate activity consists of the initial analysis of the client's needs, processing task itself, detailed specifications of the subject, and criteria evaluation. The next step is the preparation of the tender documentation, made mostly by the legal department. The tender call and the receipt of bids are the following steps. Once the receipt of bids is closed, the envelopes will be opened and their formal correctness will be checked and then the bids are compared according to the tender criteria. Based on information from previous tasks, the evaluation will be sent to tender participants and finally the selected supplier will be asked to sign the contract.

The project was firstly evaluated using quantitative data according to the formula (1). The results are in the table 1, the criticalness of the activities are in the last column.

Table 1. Analysis of the task criticalness using Simple additive weighted method

Tasks	Probability of the critical path	Duration (days)	Slack (days)	Work (hours)	Costs (Kč)	Probability of the critical path	Duration	Slack	Work	Costs	Criticalness
Purchase of the software - Tender		**62.2**		**537.6**	**399360**						
Analysis of the client's needs	1	5	0	60	70000	1	0.495	1	0.372	0.582	**0.652**
Formulation of the client's needs	0.33	5	0	80	46000	0	0.495	1	0.497	0.381	**0.434**
Market research	0.33	2	3	16	8000	0	0.192	0.559	0.095	0.064	**0.144**
Selection of the procurement mode	0.33	0.1	6.8	0.8	400	0	0	0	0	0	**0**
Processing of the internal documents	0.33	0.1	6.8	1.6	560	0	0	0	0.005	0.001	**0.002**
Specifications of the subject matter	0.67	2	0	32	24000	0.5	0.192	1	0.196	0.197	**0.357**
Specification of the evaluation criteria	1	1	0	16	8000	1	0.091	1	0.095	0.064	**0.373**
Tender documents	0.5	3	0	24	24000	0.25	0.293	1	0.146	0.197	**0.315**
Commenting on the tender documents	0.5	1	1	8	4000	0.25	0.091	0.853	0.045	0.030	**0.191**
Approval of the tender final version	1	1	0	8	4000	1	0.091	1	0.045	0.030	**0.352**
Notice of the tender	1	1	0	8	8000	1	0.091	1	0.045	0.064	**0.362**
Opening of the bids and checking of form	1	0.1	0	2.4	1600	1	0	1	0.010	0.010	**0.323**
Selection of the bid according to the eval	1	5	0	120	80000	1	0.495	1	0.749	0.666	**0.763**
Award notification	1	0.1	0	0.8	800	1	0	1	0	0.003	**0.319**
The contract	1	10	0	160	120000	1	1	1	1	1	**1**
MIN	0.333333333	0.1	0	0.8	400						
MAX	1	10	6.8	160	120000	0.189	0.164	0.129	0.23	0.288	**Weights**

For soft evaluation the linguistics terms from the table 1 are used. As an example: the activity Analysis of the client's needs is often underestimated in terms of time and resources. Therefore, the project manager evaluated its duration as usually hazardous, and costs as rather hazardous. Because it is necessary to reach the good analysis to set the consequent parameters and specifications, in terms of quality, these activities are always hazardous.

Table 2. Analysis of the threat of the task using fuzzy linguistic system

Tasks	Time hazardous	Cost hazardous	Quality hazardous	Threatening	
Purchase of the software - Tender					
Analysis of the client's needs	usually hazardous	rather hazardous	always hazardous	***Extremely threatening task***	**1**
Formulation of the client's needs	usually not hazardous	usually not hazardous	always hazardous	***Rather threatening task***	**5**
Market research	usually not hazardous	not at all hazardous	rather hazardous	***Weakly or rather threatening task***	**6**
Selection of the procurement mode	not at all hazardous	not at all hazardous	not at all hazardous	***Non-threatening task***	**9**
Processing of the internal documents	rather not hazardous	not at all hazardous	usually not hazardous	***Weakly threatening task***	**7**
Specifications of the subject matter	rather not hazardous	rather not hazardous	always hazardous	***Rather or strongly threatening task***	**4**
Specification of the evaluation criteria	usually hazardous	rather not hazardous	always hazardous	***Strongly threatening task***	**3**
Tender documents	rather not hazardous	not at all hazardous	rather hazardous	***Weakly or rather threatening task***	**6**
Commenting on the tender documents	usually hazardous	rather not hazardous	usually hazardous	***Strongly threatening task***	**3**
Approval of the tender final version	usually not hazardous	usually not hazardous	rather not hazardous	***Weakly threatening task***	**7**
Notice of the tender	usually not hazardous	usually not hazardous	rather hazardous	***Weakly or rather threatening task***	**6**
Opening of the bids and checking of for	rather not hazardous	rather not hazardous	usually hazardous	***Rather or strongly thretening task***	**4**
Selection of the bid according to the eva	rather hazardous	rather not hazardous	always hazardous	***Strongly threatening task***	**3**
Award notification	usually not hazardous	usually not hazardous	rather not hazardous	***Weakly threatening task***	**7**
The contract	usually hazardous	usually not hazardous	always hazardous	***Strongly threatening task***	**3**

The table 2 consists of the expert linguistic evaluation of the activities and the overall evaluation using fuzzy rule basis is in the last column, these terms are ordered according to their natural meanings.

5. Results and discussion

The results are in the figure 3, where the activities are ranked according to the task criticalness. The graph shows that some activities are evaluated differently according to the both evaluations.

The biggest difference is seen in criticalness and the threat of the **Commenting on the tender documents**. This difference is based on the fact that this activity is not time critical and also is not really consuming in the other parameters, but it is inherently very risky, mainly the quality is very threatening and also it can be time consuming. The difference in the evaluation of the activities **Analysis of the client's needs**, **Specification of the evaluation criteria**, **Opening of the bids and checking of formal correctness** and **Specifications of the subject matter** can be explained similarly.

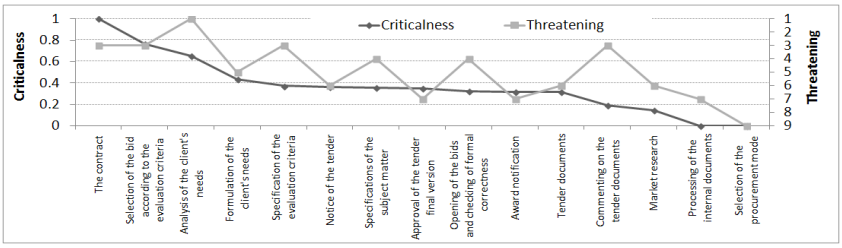


Figure 3. Comparison of the task criticalness and the task threatening

6. Conclusion

The example of quantitative and qualitative evaluation of project activities demonstrates the need for both forms of the project analysis. In this way the comprehensive evaluation of the project activities includes all their attributes.

The quantitative (hard) evaluation of task criticalness does not respect the soft factors influencing activities; it is good evaluation in the cases of low impact of these factors on the realization of the project. This approach is subjectively influenced only by setting the weights of the parameters of the criticalness.

The task qualitative evaluation of the task threat using fuzzy approach enables to include also randomness, social aspects and other soft aspects influencing the tasks and the project realization. Used linguistic terms, however, allow a large degree of subjectivity evaluation. This fact is not only negative; its positivity lies in the possibility of expert evaluation of the factors based on the content of each project activity.

Therefore the crisp data and evaluation has to be compared with the subjective, intuitive and soft evaluation. Different project managers can evaluate the tasks and the weights differently and thus the results are different. Putting different parameters and rules will lead to different results.

Acknowledgements

The research is supported by the Internal Grant Agency of the University of Life Sciences Prague – project IGA PEF 20131028 - Multicriterial Analysis of Criticality of Project Activities.

References

Brozova, H., Bartoska, J., Subrt, T., Rydval, J., 2013, "Multiple criteria evaluation of the activities criticalness in the Project management", Proceedings of the 31st International conference on Mathematical Methods in Economics, Jihlava.

Checkland, P., Scholes, J., 1990, "Soft System Metodology In Action", John Wiley, Chichester.

Gonen, A., 2012, "Project Risks selection under budget constraints", Proceedings of the 13th International Conference on Project Management and Scheduling, Leuven.

Hwang, Ch-L., Yoon, K., 1981, "Multiple Attribute Decision Making". Springer, Berlin Heidelberg, New York, 1981.

Kerzner, H., 1979, "Project Management: A Systems Approach to Planning, Scheduling and Controlling". New York: Van Nostrand Reinhold.

Liberatore, M.J., Pollack-Johnson, B., 2013, "Improving Project Management Decision Making by Modelling Quality, Time, and Cost Continuously", *IEEE Trans. Eng. Manag.*, Vol. 60, pp. 518-528.

Madadi, M., Iranmanesh, H., 2012, "A management oriented approach to reduce a project duration and its risk (variability)". European Journal of Operational Research 219, pp. 751-761.

PMI, 2013, Project Management Institute. "A Guide to the Project Management Body of Knowledge". 5th ed., Newtown Square (PA, USA), ISBN 978-1-935589-67-9.

Ross, T.J., 2010, "Fuzzy Logic with Engineering Applications", John Wiley and sons.

Sadeh, A., Zwikael, O., 2012, "Top Management Project Support: an Investigation into the Most Effective Antecedents for Project Success", Proceedings of the 13th International Conference on Project Management and Scheduling, Leuven.

Wang, Z., Klir, G.J., 1992, "Fuzzy Measure Theory", Plenum Press, New York.

Warburton, R.D.H., 2011, "A time-dependent earned value model for software projects", International Journal of Project Management, vol. 29, no. 8, pp. 1082-1090.

Wideman, R.M., 1992, "Project and Program Risk Management", Project Management Institute.

Proactive Project Scheduling with a Bi-Objective Genetic Algorithm in an R&D Department

Canan Capa[1], Gunduz Ulusoy[2]

[1]Concordia University, Canada
e-mail: c_capa@encs.concordia.ca

[2]Sabanci University, Turkey
e-mail: gunduz@sabanciuniv.edu

Keywords: proactive project scheduling, multi-objective genetic algorithm, R&D

1. Introduction

During project execution, especially in a multi-project environment unforeseen events arise that disrupt project plans resulting in deviations of project plans and budgets. Therefore, project schedules should also include solution robustness to cope with the uncertainties such that actually realized activity start times during project execution will not differ much from the baseline schedule. Constructing solution robust schedules requires proactive scheduling techniques.

The literature on proactive project scheduling is relatively scarce. Leus (2003) considers the objective of minimizing the total weighted instability of the schedules from a given deadline. Herroelen and Leus (2004) develop mathematical models for the generation of stable baseline schedules. Van de Vonder et. al. (2006) propose resource flow dependent float factor heuristic as a time buffering technique to produce robust schedules relying completely on the activity weights. Lambrechts et. al. (2008) focus on disruptions caused by stochastic resource availabilities and aim at generating stable baseline schedules. Van de Vonder et. al. (2008) introduce multiple algorithms to include time buffers in a given schedule while a predefined project due date remains respected. In a recent study, Lambrechts et. al. (2011) analytically determine the impact of unexpected resource breakdowns on activity durations and develop an approach for inserting explicit idle time into project schedules in order to protect them from possible resource unavailability. In addition to these proactive strategies, there are some risk-integrated procedures. Shatteman et. al. (2008) develop a methodology that relies on a computer supported risk management system that allows to identify, analyze and quantify the major risk factors and derive the probability of their occurrence and their impact on the duration of the project activities. Creemers et. al. (2013) propose a quantitative approach that allows to address the risk response process in a scientifically-sound manner and shows that a risk-driven approach is more efficient than an activity-based approach to analyze risks. Herroelen (2014) propose a methodology that integrates quantitative risk analysis with reliable proactive/reactive project scheduling procedures.

The problem on hand is the proactive scheduling of R&D projects with a priori assigned resources in a stochastic and dynamic environment present in the R&D Department of a leading home appliances company in Turkey. We develop a three-phase model incorporating data mining and project scheduling techniques to solve the problem. Phase I of the model, the uncertainty assessment phase, provides a systematic approach to assess uncertainty by identifying the most important factors of uncertainty, measuring the impacts of these factors to resource usage deviation levels of projects and their activities; and generating activity resource usage deviation distributions by using the well known data mining techniques: feature subset selection, clustering and classification. Phase II, the proactive project scheduling phase, proposes two scheduling approaches using a bi-objective genetic algorithm (GA). Phase III, the reactive project scheduling phase, aims at rescheduling the disrupted project activities during implementation using the scheduled order repair heuristic developed and enables the project managers to make what-if analysis and thus to generate a set of contingency plans for better preparation.

In this paper, our focus is limited to Phase II of the three-phase approach. In Section 2, the problem and the problem environment are explained. In Section 3, we present the solution methodology and in Section 4 we present the main results obtained by the implementation of the proposed proactive project scheduling approach with real data. Finally, in Section 5 we conclude and provide suggestions for future work.

2. Problem definition and environment

We consider the preemptive resource constrained multi-project scheduling problem (RCMPSP) in a stochastic and dynamic environment present in the R&D Department of a leading home appliances company in Turkey. No precedence relation is assumed between projects. A project consists of a number of events and activities that have to be performed in accordance with a set of precedence and resource constraints. Activities require two types of renewable resources: human resource and equipment. Human resource is multi-skilled. Equipment includes machines, mechanisms and laboratories. Non-renewable resources are not considered. This is due to relatively unrestricted availability of non-renewable resources. The resource requirement of activities and hence, the durations of activities are uncertain. The project network is of activity-on-node (AON) type with Finish-to-Start (FS) and Start-to-Start (SS) precedence relations with zero or positive time lags. The problem on hand can be considered an extension of the RCMPSP with generalized precedence relations and multi-skilled resources, preemption, stochastic activity durations and resource availabilities and dynamic arrival of projects. The objective is to generate solution robust baseline project schedules and minimizing the completion time for the overall project makespan. Solution robustness is a measure of the difference between the realized schedule and baseline schedule. In our case, we use the total sum of absolute deviations *(TSAD)* for solution robustness, where the absolute deviations are taken between the actual starting times and starting times realized in a set of K simulations over all activities. The problem environment differs from those in the literature in that a resource is required for the duration of its usage within an activity rather than for the whole duration of the activity requiring that particular resource. Resources can work on more than one activity in a time period (say, a week) and the duration of the usage of the resources can differ over the periods that the activity is executed. Additionally, the concept of preemption of a resource employed by an activity is also introduced.

3. Solution methodology

In this section, we present a bi-objective GA for the scheduling of a newly arrived project that uses the output of Phase I as input and employs two scheduling approaches: the single and multi-project scheduling approaches. The aim of these scheduling approaches is to generate non-dominated solution robust project schedules with the minimum makespan for the completion of all projects scheduled. Solution robustness is measured with *TSAD* of the schedule through K number of possible schedule realizations in both approaches. The single and multi-project scheduling approaches differ in that the single project scheduling approach considers the ongoing and not yet started activities and projects of the baseline schedule as fixed by the time of scheduling of the newly arrived project and uses the currently available resources to schedule it. The multi-project scheduling approach, on the other hand, schedules the ongoing and not yet started activities and projects of the baseline schedule anew together with the newly arrived project. Since the two scheduling approaches differ in the way they adopt for the scope of scheduling, the definitions of *TSAD* and makespan, thus, the objectives considered in the bi-objective GA also differ although they both try to minimize *TSAD* and makespan. Note that in the proactive project scheduling approaches, a set of non-dominated robust project schedules are generated. From these non-dominated robust schedules, the decision maker can choose the schedule that best fits the current project management environment in the system. Proposed bi-objective GA is an adopted version of NSGA-II suggested by Deb et al. (2002), which uses an explicit diversity generation procedure called *crowded-comparison* along with an elite-preservation procedure called *non-dominated sorting*. An individual is represented by a precedence feasible activity list. We make use of one-point crossover and swap mutation operators. Population management is the same as in NSGA-II. However, our bi-objective GA differs in the schedule generation scheme and chromosome evaluation procedures.

3.1. Schedule generation

Since the work of resources on activities is preemptive, a schedule is represented with the lists of resource, activity, week and amount *(r,a,t,k)* quadruple. Each *(r,a,t,k)* quadruple shows that resource r works on activity a at the time instant t for k working hours. Our resource schedule generation scheme starts with scheduling the resources of the first activity in the chromosome. Note that, resource order for scheduling is not important since all orders give the same work schedule for that activity. Considering the earliest precedence feasible starting time of activities

and starting at the first available time instant, resources are scheduled until they reach their required usage hours. After all the resources of the first activity in the chromosome are scheduled, starting and ending time of that activity is determined by simply checking the work schedules of the resources that activity requires. Then, the earliest starting time of the successor activities are updated. This procedure is repeated until all the activities in the chromosome are scheduled.

3.2. Chromosome evaluation

For a given order of activities both the overall makespan and solution robustness are assessed through a set of *K* realizations mimicking the implementation phase, where a realization corresponds to a sample instance obtained by a simulation run using the activities' percentage resource requirement deviation distributions, which is determined by Phase I. For this purpose, two alternative chromosome evaluation heuristics with the objective of quality robustness represented with makespan and solution robustness expressed in terms of *TSAD* value of the robust activity starting times from their counterparts in all *K* realizations, are considered: chromosome evaluation heuristic I (*CEH-I*), and chromosome evaluation heuristic II (*CEH-II*). *CEH-I* solves a *TSAD* minimization model by LP. Using the activity starting times realized in simulations, this *TSAD* minimization model aims at finding robust start times that minimize the *TSAD* value of the scheduled activities. Note that resulting activity start times might be completely different than the activity starting times in *K* realizations and they might be resource-infeasible. Thus, using the resulting robust starting times, first, feasibility of these starting times is checked and if infeasible, the schedule is fixed with deferring the infeasible activities. On the other hand, in *CEH-II, K* realizations are sorted in their non-domination ranks using the corresponding makespan and *TSAD* values. Among the schedules that are on the Pareto front, the schedule having the minimum *TSAD* is selected as the robust schedule of the chromosome. The makespan and the *TSAD* values of the resulting schedule are used as performance measures of the chromosome. Note that the *TSAD* in the multi-project scheduling approach includes the deviations of the starting times of the activities not yet completed as well.

4. Implementation with real data

For the implementation, 37 completed R&D projects varying in size all initiated by the R&D Department between 2007 and 2011 are used as test instances to compare the performances of the two proactive project scheduling approaches. Project networks are of AON type. FS and SS are generalized precedence relations with zero and positive time lags. There is no precedence relation between the projects. The number of activities vary between five and up to 39 and average network complexities are 1.2, 1.5, and 1.7 for small, medium and large networks, respectively. The two types of renewable resources are: Human resource and equipment. Activities require from one human resource to a total of more than 11 human resources and equipment. The weekly capacity of human resources is 45 working hours. For the resources in the equipment category, capacity values differ from nine working hours to 672 working hours. Resource availability may vary between resources and periods regarding the current workloads.

All codes are written in Microsoft Visual Studio C# and CPLEX 12.5 is used as the MILP solver. All tests are performed on a computer with a 3.20 GHz Intel(R) Core(TM) i7 CPU 960 processor and 8 GB of RAM.

The best combination of the parameters to be used in the bi-objective GA is determined through extensive experimentation resulting in the following set of parameters: the crossover rate of 0.95; mutation rate of 0.05; population size 50; the number of generations 50; and the number of schedule realizations for a chromosome 100. The results obtained by the use of *CEH-I* and *CEH-II* are compared with respect to CPU time, diversity of the solutions and solution quality. Because of space limitations, we do not present the tables showing results but we provide the main results obtained with the analysis of the results. It is seen that the CPU time required to schedule the projects is less for almost all projects when *CEH-II* is used instead of *CEH-I* since fitness of a chromosome is calculated using an already generated schedule in *CEH-II*. Thus, it seems sorting the schedules generated in the simulation with respect to their non-domination level requires less computational time than solving the *TSAD* minimization model and generating a new schedule using the output of the *TSAD* minimization model. It is also seen that when *CEH-I* is used, less number of non-dominated schedules are obtained for each project and it tends to find schedules with less *TSAD* while *CEH-II* tends to find schedules with smaller makespan values.

When we compared the results of the single and multi-project scheduling approaches, we saw that for most of the projects, single project scheduling approach gives better completion times. On the other hand, if we think all the projects as a composite project, the completion time of this composite project obtained with multi-project scheduling approaches approximately 3.5 months earlier using *CEH-I* and approximately 4.5 months earlier using *CEH-II* in a time horizon of 34months. Hence, if completing the composite project is more important than completing the projects individually, multi-project scheduling approach is better. On the other hand, a disadvantage of multi-project scheduling is that it re-schedules all the active activities with a new project initiation, so an activity is scheduled more than once even if there is no disruption affecting that activity. This re-scheduling increases system nervousness and decreases the morale of the human resources that work on the activities. An additional disadvantage is that the multi-project scheduling approach needs more CPU time than the single project scheduling approach.

5. Conclusion and future work

In this paper, we presented the proactive project scheduling phase of the three-phase approach developed for robust project scheduling. To the extent of our knowledge, this study is the first study considering multiple objectives on proactive project scheduling literature for the problem of the preemptive version of the RCMPSP with generalized precedence relations. To obtain robust baseline schedules, in the proactive project scheduling phase, we suggested two scheduling approaches each using a bi-objective GA with two different chromosome evaluation heuristics. Solution robustness is assured with *TSAD* minimization after a pre-specified number of schedule realizations are obtained for a chromosome. The other objective is the minimization of the makespan over all projects. The proactive project scheduling approaches are implemented on the real data from the R&D Department of a leading home appliances company in Turkey. Although we have used these two objectives, some other objectives could be used or added to the model as well. A further extension of our work could be considering the concepts of activity flexibility, project flexibility, activity priority and project priority while scheduling the projects.

References

Creemers, S., Demeulemeester, E., Van de Vonder, S. (2013). A new approach for quantitative risk analysis. *Annals of Operations Research*, 1-39.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. A. M. T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182-197.

Herroelen, W. (2014). A risk integrated methodology for project planning under uncertainty. In *Essays in Production, Project Planning and Scheduling*, 203-217. Springer US, New York, N.Y.

Herroelen, W., Leus, R. (2004). Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research*, 42, (8), 1599-1620.

Lambrechts, O., Demeulemeester, E., Herroelen, W. (2008). A tabu search procedure for developing robust predictive project schedules. *International Journal of Production Economics*, 111(2), 493-508.

Lambrechts, O., Demeulemeester, E., Herroelen, W.(2011).Time slack-based techniques for robust project scheduling subject to resource uncertainty. *Annals of Operations Research,*186(1), 443-464.

Leus, R. (2003). The generation of stable project plans. PhD Dissertation. Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.

Schatteman, D., Herroelen, W., Van de Vonder, S., Boone, A. (2008). Methodology for integrated risk management and proactive scheduling of construction projects. *Journal of Construction Engineering and Management*, 134(11), 885-893.

Van de Vonder, S., Demeulemeester, E., Herroelen, W. (2008). Proactive heuristic procedures for robust project scheduling: An experimental analysis. *European Journal of Operational Research*, 189(3), 723-733.

Van de Vonder, S., Demeulemeester, E., Herroelen, W., Leus, R. (2006). The trade-off between stability and makespan in resource-constrained project scheduling. *International Journal of Production Research*, 44(2), 215-236.

Batching and dispatching with limited queue time in wafer fabrication

Federica Ciccullo, Margherita Pero, Giovanni Pirovano, Andrea Sianesi

Politecnico di Milano – Department of Management, Economics and Industrial Engineering
Via Lambruschini 4B, 20156 Milano (Italy)

federica.ciccullo@polimi.it, margherita.pero@polimi.it, giovanniluca.pirovano@polimi.it, andrea.sianesi@polimi.it

Keywords: Time constraints, batching, wafer fab, dispatching.

1. Introduction

Electronics industry is one of the largest industries in the world. Key players in this industry are the semiconductor manufacturers, which produce integrated circuits on silicon wafers. According to a report by the Semiconductor Industry Association (SIA), the worldwide sales of semiconductors reached $25.53 billion for the month of July 2013, and an increase of 5.1 percent over July 2012. Semiconductor manufacturing process consists of four stages: wafer fabrication, wafer probe, assembly, and final test. Wafer fabrication is one of the most complex manufacturing processes. It takes place in fabs, that are very large systems with tens to hundreds machines moving hundreds to thousands wafers (Moench et al., 2011). The scheduling problems in a wafer fab is similar to scheduling flexible job shop but it is more complicated since there are several elements which have to be considered, i.e. (Dabbas and Fowler 2003): (1) Re-entrance of lots in some stages, i.e. lots must be processed by the same machine during multiple visits; (2) Different types of machines depending on the object processed, i.e. wafer, lot (a set of wafers) or batch (a set of lots); (3) Setup times: Changing between different operations needs setup actions in several stages in the fabs; (4) Auxiliary resources: In some stages, auxiliary resources are required, such as the reticles in the photolithography stage; (5) Multiple orders per lot: due to technology innovation and increase in the wafer size, customer order is reducing, so various customers' orders might be grouped in a lot; (6) Uncertainties and machine availability: the arrival of new orders, the cancellation of orders, and long machine failure are common unplanned events; (7) Time constraints: in some cases, the time between the end of an operation n and the start of the operation $n+q$ must be lower than a time-limit, in order to guarantee the lots' quality. Scheduling in such a context is challenging for managers. Therefore this work presents the preliminary results of a research carried on in the framework of a EU-funded project: "Integrated Solutions for Agile Manufacturing in High-mix Semiconductor Fabs" (INTEGRATE), aimed at developing scheduling algorithms to support wafers manufactures. The studied case is the one of the plant of ST-MICROELECTRONICS in Catania. In particular, the proposed approach support them in defining the batches and dispatching the lots among three consecutive operations (i.e. one cleaning and two diffusions) characterized by time constraints between them and no batches affinity between the first and the second operation. Moreover, the batches might be composed of lots of wafers of different families. Considering j as the lot's index, the problem can be described as in (1) (in line with the notation proposed in Bucker (2007)):

$$R\,J\left|\, p_j, d_j, p-batch, incompatible-batch, T_{lags}^{min}\right| \min(\bar{T}), \min(R_{work}), S_{scrap} = 0 \quad (1)$$

Where the first part of (1) defines the machine environment, which is unrelated parallel machines (R) with different operations (J). The second part describes the product characteristics. In particular: the studied situation is characterized by different processing time of the lots (p_j), relevant and different due dates of the lots (d_j), the time for processing a batch on a machine is the longest processing time among all the lots in the batch (p-batch), not all the lots can be batched together (incompatible-batch), and presence of time constraints between operations (Tl_{ags}^{min}). The last part represents the objectives, which are: minimizing the average flow time of the lots (T) and the number of re-cleaned lots (R_{work}), and avoid scrapped lots (S_{scrap}). To this aim, two heuristic algorithms have been proposed. We use the term "heuristic algorithms" to describe our approaches, since we propose two methods that are both composed by a series of steps (thus the term algorithm), but they support the decision maker in finding a good solution in a limited amount of time, but they do not assure to find the optimal solution (thus the term heuristic). They share the same trigger events, but not the

prioritisation rule, which for the former is based on the ATC-Index (Vepsalainen and Morton, 1987) while on BATC-Index (Cerekci and Banerjee, 2010) for the latter. A simulation model of ST-MICROELECTRONICS Catania's wafer fab has been developed on Arena™ and a simulation campaign performed in order to validate and compare the two heuristic algorithms to find the one which allows to reach better performance (i.e. minimum rework, average flow time and zero scrapped lots).

2. Literature review

Researchers have addressed the problem of batching and dispatching in different ways. Various elements can be considered in order to classify the methods: (i) machine environment e.g work area, parallel machines; (ii) process restrictions e.g. re-entrant flow, time constraints, sequence-dependent; (iii) objectives e.g. related to processing time, throughput, on time delivery (Mönch et al. 2011). Different solving techniques have been used, i.e. mathematical programming, simulation and heuristic algorithms (Mathirajan and Sivakumar 2006). Batching and dispatching problems are often solved by means of a rule based on the value of an index, that takes into account lot's features. Vepsalainen and Morton (1987) develops an index called Apparent Tardiness Cost (ATC), which assigns the priority on the basis of the expected tardiness cost per immediate processing requirements. The aim is to minimize the sum of weighted tardiness, in a job shop. An evolution of this index is presented by Cerekci and Banerjee (2010). They focus on mean tardiness performance of a batch processing machine in a two-stages system by including an upstream unit capacity machine. They propose two new control strategies, called BATC-I and BATC-II, in order to find the batches composition and then dispatch the batches. Akçali et al. (2000) focus on loading policy based on the minimum batch size, and different dispatching policies based on an index called critical ratio of lots, in diffusion area of a wafer fab. They suggest that by setting a variable minimum batch size, which depends on the production volume, the flow time can be reduced. They use simulation in order to test the different approaches, and the results show that the loading policy has a significant influence on the flow time and on the overall processing time of the products, while dispatching policy has less significant effect. Some approaches incorporate other information about the state of the fab. Solomon et al. (2002) develop a dispatching policy to be used in batch processing machines that incorporates information about future arrivals and the status of critical machines, so to balance the time that lots spend waiting at the batch processing machine and the time spent in setups, in order to improve the makespan. As far as time constraints is concerned, the relevance of the topic has been recognized by both practitioners and researchers. Solutions have been proposed in industries other than semiconductors and considering both tardiness and makespan as objectives. In fact, Gicquel, et al. (2012) study hybrid flow-shop scheduling problem arising from a bio-process industry, where a variety of constraints has to be taken into account, such as limited waiting time between processing stages. They propose an exact solution approach for minimizing the total weighted tardiness, based on a discrete time representation and a mixed-integer linear programming formulation. Joo and Kim (2008) face the problem of two stages single server with time constraints between the two stages. They apply the branch- and-bound algorithm using the dominance properties in order to minimize the makespan. Similarly, Li and Li (2007) study the hybrid flow-shop scheduling problem with limited waiting time constraint in a multi stage process, characterized by parallel unit capacity machines. The objective is to minimize the makespan for a given set of jobs. They propose a recursive backtracking algorithm, which schedules each job from the first stage to the last. Some works include also the problems related to batch processing machines. For example, Su (2003) proposes a heuristic algorithm to minimize the makespan on a two stages process, where at the first stage there is a batch processing machine. It should be noted that they model only lots belonging to the same family. Literature review shows that few papers consider both batching problem and time constraints, and none proposes a model applicable to ST-MICROELECTRONICS's plant situation.

3. Problem setting and proposed heuristic algorithms

3.1 Problem setting

The aim of this work is to provide the production managers of ST-MICROELECTRONICS Catania's plant, with reference to the cleaning and diffusion area, an heuristic algorithm to define: (1) Which lots should be included in a batch, (2) Which batch should be dispatched first, and (3)

When a batch should be dispatched. Only a part of ST Catania's Plant is considered, precisely the cleaning and diffusion areas. Three types of lots' flows go through these two areas: (i) lots which have to perform only cleaning, (ii) lots which have to perform cleaning and one diffusion operation, and (iii) lots which have to perform cleaning and two consecutive diffusion operations. In addition, the lots belong to different family, which means they are characterized by different recipes, therefore they cannot be processed together. Furthermore, the lots can have the same recipe at the cleaning operation, but different recipes at the diffusion operation. This means that lots loaded in the same batch at the cleaning operation not necessarily can be loaded directly in the same batch on the first diffusion operations. While the batches that have been loaded on the first diffusion operation can be loaded as they are also on the second diffusion operation, without any recombination of lots, provided that the capacity limit of the machine is respected. Furthermore, the two areas are characterized by a set of parallel batch processing machines, which can have different maximum batch size. Another characteristic is that not all the machines are qualified to process all the recipes. Moreover, it should be noted that the processing time of a batch is equal to the highest processing time of the lots belonging to the batch itself. Lastly there are minimum time constraints between the different operations. This means that a lot can wait in the queue in front of a furnace machine no longer than a threshold, given by the recipe of the lot itself. The queue time is calculated from the moment when the lot finishes to be processed at an operation to the moment when the lot is loaded on the machine at the next operation. If a lot exceeds the time constraint between a cleaning operation and diffusion operation, it must be reworked, while if a lot exceeds the time constraint between the two consecutive diffusion operations, it must be scrapped. The information related to the job, such as due date, processing time, arrival time at first stage, can be obtained from the information system of the fab. The same applies for the information related to the fab status, for example machine status or number of lots in the queue. Preemption is not allowed.

3.2 Proposed heuristic algorithms

Two heuristic algorithms have been developed and then compared. The objectives taken into account when developing them are: minimize the average flow time of lots, minimize the reworked and avoid the scrapped lots. Both methods were developed starting from two ideas, i.e.: (1) In order to avoid lots waiting at the batch processing machines due to lack of other lots to be included in the batch, the method tries to compose the batches at the beginning of the cleaning operation by grouping lots sharing the same flow type and recipes. (2) In order to avoid to overstep the time constraint the queues in front of the furnace machines are monitored and the batches are loaded on a machine at cleaning area only when the expected queue time at the following stage is lower than the time constraint.The proposed heuristics algorithms are characterized by three main elements: (a) prioritisation: which aims to compose the batches and calculate the priority index to prioritize the batches, (b) trigger events and dispatching: which aims to define the instant when prioritization should be performed, and then, based on the prioritisation result, the batch must be dispatched, and (c) queues control: which aims to limit the number of batches waiting at the second stage in order to avoid to overstep the time constraint

In this work two alternative methods for performing the prioritisation are presented, namely ATC-index (Vepsalainen and Morton 1987) and the BATCH-index (Cerekci and Banerjee, 2010).

The second task is to define the trigger event, which activates the heuristic algorithms. At the cleaning operation there are two trigger events, the first one is: "a cleaning machine is empty", while the second is: "there is a new lot in queue at the cleaning and a cleaning machine is empty". Whereas, the trigger event for the diffusion operation is: "there is at least one complete batch in front of the diffusion machine and at least one load-space of the diffusion machine is empty".

The queue control depends on the following equation (equation 2):

$$TC < \frac{Wip_i}{u_i * a_i} \tag{2}$$

The equation aims to check if a specific machine i can process all the batches in its queue without overstep the time constraint (TC), by considering the waiting batches (Wip_i), the service rate (u_i) and machine availability (a_i), in line with Little's law. Finally, the heuristic algorithms are divided in four steps: (a) Priority index calculation: this part is the same for all the lots, (b) Cleaning scheduling: applied when the chosen lot has to do only the cleaning operation, (c) Cleaning-Diffusion scheduling: used when the lot has to do one cleaning and one diffusion, and (d) Cleaning-Diffusion-Diffusion scheduling: implemented when the lot has to do one cleaning and two diffusions. The first part of the heuristic algorithms aims to select the lots to be dispatched first. The trigger events are

the two explained above. ATC is calculated in both heuristic algorithms, then only for the second also BATC is calculated. ATC allows for selecting the lot to be considered first for batching. BATC allows for selecting the batch to dispatch first on the cleaning machine. The three following parts of the heuristic algorithms, i.e. b, c and d, differ one from the other for the maximum batch capacity since in the first case is the capacity of the available cleaning machine which can perform the recipe of the considered batch; for the second is the capacity of the diffusion machine which can perform the recipe of the considered batch and respects (2), while for the third Bmax is the maximum among the maximum capacities of the machines which can perform the recipe of the considered batch and full fill the condition of the equation (2), respectively for the first and the second diffusion operations. In addition, the batches at diffusion are dispatched on the furnaces using FIFO (first in first out) logic.

3.3 Simulation experiment and results

A simulation model of the analysed system is developed in Arena^{TM} in order to validate and compare the two heuristics algorithms. The input data came from the real situation of ST-MICROELECTRONICS Catania plant. A simulation campaign is performed using as decision variables: (i) Δt: the look-ahead time; (ii) *Wtmax*: the maximum waiting time at the first operation; (iii) *Time_load_space*: time for when the load-space is booked, as control variables: (i) Waiting time at the diffusion operations (< time constraint); (ii) Waiting time at the cleaning operation, and as performance: (i) average flow time (ii) number of scrapped lots (iii) number of lots reworked. The performance of the proposed heuristic algorithms have been compared.

References

Akçali, E., Uzsoy, R., Hiscock, D., G., Moser, A., L., and Teyner, T., J., "Alternative loading and dispatching policies for furnace operations in semiconductor manufacturing: a comparison by simulation." Proceedings of the 32nd conference on Winter simulation. Society for Computer Simulation International, (2000).

Attar, S., Mohammadi, F., M. and Tavakkoli-Moghaddam, R. "Hybrid flexible flowshop scheduling problem with unrelated parallel machines and limited waiting times." The International Journal of Advanced Manufacturing Technology (2013): 1-17.

Brucker, Peter, ed. Scheduling algorithms, 5th ed., Springer (2007).

Cerekci, A. and Banerjee, A. "Dynamic control of the batch processor in a serial-batch processor system with mean tardiness performance." International Journal of Production Research 48.5 (2010): 1339-1359.

Dabbas, R., M. and Fowler., J., W. "A new scheduling approach using combined dispatching criteria in wafer fabs." Semiconductor Manufacturing, IEEE Transactions on 16.3 (2003): 501-510.

Gicquel, C., Hege, L., Minoux, M. and Van Canneyt, L."A discrete time exact solution approach for a complex hybrid flow-shop scheduling problem with limited-wait constraints." Computers & Operations Research 39.3 (2012): 629-636.

Joo, B., J., and Kim., Y., D. "A branch-and-bound algorithm for a two-machine flowshop scheduling problem with limited waiting time constraints." Journal of the Operational Research Society 60.4 (2008): 572-582.

Li, T. and Li, Y."Constructive backtracking heuristic for hybrid flowshop scheduling with limited waiting times." Wireless Communications, Networking and Mobile Computing WiCom 2007. International Conference on. IEEE (2007).

Mathirajan, M., and Sivakumar., A., I. "A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor." The International Journal of Advanced Manufacturing Technology 29.9-10 (2006): 990-1001.

Mönch, L., Fowler, J.,W., Dauzère-Pérès, S., Mason, S., J., and Rose O. "A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations." Journal of Scheduling 14.6 (2011): 583-599.

Solomon, A., Fowler, J.,W., Jensen, P., H."The inclusion of future arrivals and downstream setups into wafer fabrication batch processing decisions." Journal of Electronics Manufacturing 11.02 (2002): 149-159.

Su, L., H. "A hybrid two-stage flowshop with limited waiting time constraints." Computers & Industrial Engineering 44.3 (2003): 409-424.

Vepsalainen, Ari, P., J. and Morton., T., E., "Priority rules for job shops with weighted tardiness costs." Management science 33.8 (1987): 1035-1047.

A new approach to minimize the makespan of various resource-constrained project scheduling problems

José Coelho[1,2] and Mario Vanhoucke[3,4,5]

[1] Technical University of Lisbon, Lisbon (Portugal)
[2] Universidade Aberta, Lisbon (Portugal)
Jose.Coelho@uab.pt
[3] Faculty of Economics and Business Administration, Ghent University, Belgium
mario.vanhoucke@ugent.be
[4] Technology and Operations Management Area, Vlerick Business School, Belgium
mario.vanhoucke@vlerick.com
[5] Department of Management Science and Innovation, University College London, United Kingdom
m.vanhoucke@ucl.ac.uk

Keywords: Resource-constrained project scheduling, Multi-mode, SAT

1 Introduction

The resource-constrained project scheduling problem (RCPSP) is one of the most investigated scheduling problems in the project management literature. Resource-constrained project scheduling is the process of constructing a project schedule within the limited amount of resources available. It requires the examination of the possible unbalanced use of resources over time to resolve over-allocations (the so-called resource conflicts) when more resources are required than available. The critical path based scheduling methods will often schedule certain activities simultaneously, but when more resources such as machines or people are needed than there are available, these activities will have to be rescheduled concurrently or even sequentially to resolve the resource constraints. The resource-constrained project scheduling problem aims at resolving these resource conflicts such that the total project makespan is minimized.

This abstract presents a new solution approach to solve the resource-constrained project scheduling problem in the presence of multiple modes with mode identity constraints and two types of logical constraints. Apart from the traditional AND constraints with minimal time-lags, these precedences are extended to OR constraints. These logical constraints extend the set of relations between pairs of activities and make the RCPSP definition somewhat different from the traditional RCPSP research topics in literature. It is known that the RCPSP with AND constraints, and hence its extension to OR constraints, is NP-hard.

The new algorithm consists of a set of network transformation rules that remove the OR logical constraints and transforms them into AND constraints and extends the set of activities to maintain the original logic. A satisfiability (SAT) solver is used to guarantee the original precedence logic and is embedded in a meta-heuristic search to construct resource feasible schedules that respect both the limited renewable resource availability as well as the precedence logic. Computational results on a newly generated dataset that is publicly available show that the procedure is able to generate near-optimal solutions in reasonable time.

2 Problem description

In this abstract, a new solution approach will be proposed that will be able to solve various versions of the multi-mode resource-constrained project scheduling problem (section 2.1) and will take both AND and OR precedence relations (section 2.2) into account. The solution approach consists of a combined metaheuristic search and a SAT solver search (section 2.3), based on the principles used by [Coelho and Vanhoucke, 2011], to solve this new problem to near-optimality in reasonable time.

2.1 Multi-mode

Many research efforts have extended the RCPSP to the presence of multiple activity modes where each activity can be executed under a different duration and a corresponding renewable and nonrenewable resource use (the problem is further abbreviated as the MMRCPSP). Due to the complex nature of the problem, only a few exact algorithms have been presented in literature, and most algorithms presented in literature therefore consist of heuristic or meta-heuristic solution approaches. Moreover, a clear distinction can be made between algorithms incorporating both renewable and non-renewable resource constraints and algorithms limited to projects with only renewable resource constraints. In the paper written by [Van Peteghem and Vanhoucke, 2014], most algorithms have been benchmarked on existing and newly presented data instances. We will compare our solution approach on these data instances, extended with extra features such as mode identity constraints [Salewski et al., 1997] as well as the extended precedence relations that will be discussed in the next subsection.

2.2 AND/OR constraints

Scheduling activities with AND/OR constraints is not new in literature, and has been discussed in [Gillies and Liu, 1995], [Adelson-Velsky and Levner, 2002] and [Möhring et al., 2004]. They classify the traditional precedence relations to the class of AND constraints to model that an activity can only start when all predecessors have been finished. An OR constraint, however, is slightly different as it allows an activity to start as soon as any of its predecessors has been completed. Consequently, the OR precedence relations differ from the traditional AND constraints that they only require one predecessor to finish before the successor can start.

2.3 SAT approach

The solution approach to solve the existing and novel problems to near-optimality consists of the three-phased approach, including a set of network transformation rules, the use of a SAT solver and a metaheuristic search for resource feasible solutions, as briefly discussed along the following lines.

- Step 1. Network transformation: Including multi-modes and AND/OR relations into the RCPSP will be done by extending the network with various dummy nodes and dummy arcs that enable the algorithm to solve the problem using a traditional RCPSP solver. The specific details of the network conversions are not outlined in this document but will be shown in the presentation.
- Step 2. SAT Solver: In order to guarantee that the newly incorporated constraints will be satisfied, a satisfiability (SAT) problem solver will evaluate the logical conditions of the intermediate constructed schedules that act as a go/no go to further optimize the partial schedule in case all constraints are satisfied. A similar approach has been

proposed by [Coelho and Vanhoucke, 2011] to solve overallocations of non-renewable resources.
– Step 3. Metaheuristic search: A metaheuristic search procedure will evaluate thousands of solutions in reasonable time based on the traditional principles borrowed from genetic algorithms.

3 Computational design

During the presentation, the design for a new public dataset will be described and preliminary computational results for the computational experiments will be reported. These data instances will be generated by the network generators of [Demeulemeester et al., 2003, Vanhoucke et al., 2008] and will be used to test our solution approach.

Table 1 displays general and preliminary results for the new solution approach to solve the multi-mode RCPSP and shows its performance in comparison with the currently best performing algorithm from literature. To that purpose, we relied on the data instances of [Van Peteghem and Vanhoucke, 2014] that have been especially designed to solve MM-RCPSP instances and to benchmark the resulting solution in literature. In the table, the results for a 1,000, 5,000 and 50,000 schedules limit are shown for the instance set MM-LIB50, MMLIB100, and MMLIB+. For each set and stop criterion, the percentage of feasible instances, the deviation percentage from the lower bound, and the deviation percentage from the best solution known are displayed. The deviation percentages are calculated only for the feasible instances.

The results show that the procedure is competitive with the currently best performing algorithms since it stays at an average 2% from the best-known solutions. However, the newly presented solution approach can, unlike most algorithms presented in literature, be used to solve several extensions of the well-known MMRCPSP without any change in the algorithm. It should be noted that these results are only preliminary, and more detailed results will be presented on the workshop.

The infeasibility on some of the problem instances occur due to two reasons. Firstly, the number of activity modes might be too high for the current implementation of the RCPSP algorithm, and secondly, the number of backtracks limit during the SAT search was often reached. In the near future, an update of the data structure of the RCPSP solver is on the agenda in order to deal with a larger number of activity modes. Moreover, updating the SAT procedure to make a more specific search in order to find an activity mode assignments in a more efficient way also lie in our future research intentions.

Table 1. Preliminary results on a sample of the test data

Schedule Limit:		1,000	5,000	50,000
MMLIB50	feasibility	83%	83%	83%
	dev.LB	30%	21%	16%
	dev.Best	13%	5%	1%
MMLIB100	feasibility	83%	83%	82%
	dev.LB	40%	28%	19%
	dev.Best	19%	9%	2%
MMLIB+	feasibility	n.a.	50%	50%
	dev.LB	n.a.	67%	54%
	dev.Best	n.a.	10%	2%

4 Conclusions

In this abstract, the integration between resource-constrained project scheduling and a satisfiability (SAT) problem solver, as originally presented by [Coelho and Vanhoucke, 2011] has been extended to the presence of AND/OR constraints. Computational results tested on the datasets presented in [Van Peteghem and Vanhoucke, 2014] show promising results and show that the current state of the procedure can compete with some of the existing algorithms in literature. However, the newly presented solution approach can, unlike most algorithms presented in literature, be used to solve several extensions of the problem formulation without any change in the algorithm.

References

[Adelson-Velsky and Levner, 2002] Adelson-Velsky, G. M. and Levner, E. (2002). Project scheduling in and-or graphs: A generalization of dijkstra's algorithm. *Mathematics of Operations Research*, 27:504–517.

[Coelho and Vanhoucke, 2011] Coelho, J. and Vanhoucke, M. (2011). Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. *European Journal of Operational Research*, 213:73–82.

[Demeulemeester et al., 2003] Demeulemeester, E., Vanhoucke, M., and Herroelen, W. (2003). Rangen: A random network generator for activity-on-the-node networks. *Journal of Scheduling*, 6:17–38.

[Gillies and Liu, 1995] Gillies, D. and Liu, J. W. (1995). Scheduling tasks with and/or precedence constraints. *SIAM Journal on Computing*, 24:797–810.

[Möhring et al., 2004] Möhring, R., Skutella, M., and Stork, F. (2004). Scheduling with and/or precedence constraints. *SIAM Journal on Computing*, 33:393–415.

[Salewski et al., 1997] Salewski, F., Schirmer, A., and Drexl, A. (1997). Project scheduling under resource and mode identity constraints: Model, complexity, methods and application. *European Journal of Operational Research*, 102:88–110.

[Van Peteghem and Vanhoucke, 2014] Van Peteghem, V. and Vanhoucke, M. (2014). An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research*, 235:62–72.

[Vanhoucke et al., 2008] Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., and Tavares, L. (2008). An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, 187:511–524.

Minimizing the makespan of a project with stochastic activity durations under resource constraints

Creemers S

IESEG School of Management, France
s.creemers@ieseg.fr

Keywords: Resource constraints, Makespan minimization, Stochastic activity durations..

1 Introduction

The resource-constrained project scheduling problem (RCPSP) has been widely studied. Most of the literature focuses on the deterministic RCPSP with minimum-makespan objective. A fundamental assumption of the deterministic RCPSP is that activity durations are known in advance; they are certain. In reality, however, this is almost never the case. In this abstract, we develop an exact procedure to solve the stochastic resource-constrained scheduling problem (SRCPSP). A computational experiment shows that our approach works best when solving small-to medium-sized problem instances where activity durations have a moderate-to-high level of variability. For this setting, our model outperforms the existing state-of-the-art.

Most of the literature on resource-constrained project scheduling assumes that activity durations are known in advance (i.e., they are deterministic). In reality, however, activity durations are often uncertain (Ashtiani et al. 2011). The stochastic resource-constrained project scheduling problem (stochastic RCPSP or SRCPSP) studies the RCPSP when activity durations are stochastic. In contrast to the deterministic RCPSP, the SRCPSP has received only little attention in the literature (for a survey of the literature on the deterministic RCPSP, refer to to Demeulemeester and Herroelen (2002) and Neumann et al. (2003)). Because the deterministic RCPSP, and hence the SRCPSP, is known to be $\mathscr{NP}$-hard (Blazewicz et al. 1983), most of the researchers have focussed their efforts on heuristic procedures. Golenko-Ginzburg and Gonik (1997) propose two heuristics that both rely on solving a series of multi-dimensional knapsack problems (the first heuristic uses an exact procedure whereas the second procedure resorts to a heuristic solution of the knapsack problems). Tsai and Gemmill (1998) apply simulated annealing and tabu search procedures whereas Ballestín and Leus (2009) use a genetic algorithm and a GRASP algorithm respectively. Ashtiani et al. (2001) adopt a two-phase local-search procedure. Stork (2001), who builds on the work of Igelmund and Radermacher (1983) and Möhring (2000), is one of the few researchers that has developed exact procedures to solve the SRCPSP. In his PhD, he compares the performance of five different branch-and-bound algorithms.

In this abstract, we extend the work of Creemers et al. (2010) and use a backward stochastic dynamic-programming (SDP) recursion to determine the minimum completion time of resource-constrained projects with stochastic activity durations. We use acyclic phase type (PH) distributions to model activity durations and match the first two moments of the activity duration distributions. The complexity of the problem increases with the number of project activities and with decreasing levels of activity duration variability. Therefore, the model is intended for small-to medium-sized problem instances where activity durations have a moderate-to-high level of variability.

2 Definitions and problem statement

A project is a network of activities that can be represented by a graph $G = (V, E)$, where $V = \{0, 1, \ldots, n\}$ is a set of nodes and $E = \{(i, j) | i, j \in V\}$ is a set of arcs. The nodes represent project activities whereas the arcs connecting the nodes represent precedence relationships. Activities 0 and n are dummy activities and represent the start and completion of the project respectively. The duration of an activity i is a random variable $\tilde{p}_i$ and has expected value μ_i and variance σ_i^2. $\tilde{\mathbf{p}} = \{\tilde{p}_0, \tilde{p}_1, \ldots, \tilde{p}_n\}$ denotes the vector of the activity duration random variables. p_i is a realization (or random variate) of $\tilde{p}_i$ and $\mathbf{p} = \{p_0, p_1, \ldots, p_n\}$ is a realization of $\tilde{\mathbf{p}}$. An activity i can start when all of its predecessors are finished and if sufficient resources are available. There are K renewable resource types. The availability of each resource type k is denoted by R_k. Each activity i requires $r_{i,k}$ units of resource k, where $r_{0,k} = r_{n,k} = 0$ for all $k \in \mathscr{R} = \{1, 2, \ldots, K\}$.

A solution to the deterministic RCPSP is a schedule $S = \{S_0, S_1, \ldots, S_n\}$, where S_i is the starting time of an activity i, $S_0 = 0$ (i.e., the project starts at time 0), and S_n represents the completion time (or makespan) of the project. In addition, define $\mathscr{A}(S, t) = \{i \in V : S_i \leq t \wedge (S_i + p_i) \geq t)$, the set of activities in schedule S that are active at time t. A schedule S is feasible if:

$$S_i + p_i \leq S_j \qquad \forall (i, j) \in E, \tag{1}$$

$$\sum_{i \in \mathscr{A}(S,t)} r_{i,k} \leq R_k \qquad \forall t \geq 0, \forall k \in \mathscr{R}, \tag{2}$$

$$S_i \geq 0 \qquad \forall i \in V. \tag{3}$$

The optimal schedule S^* minimizes S_n subject to Constraints 1–3.

Because activity durations are not known in advance, a solution to the SRCPSP is a policy rather than a schedule. A policy Π is a set of decision rules that defines actions at decision times. Decision times are typically the start of the project and the completion times of activities. An action, on the other hand, corresponds to the start of a precedence-and-resource-feasible set of activities. In addition, decisions have to respect the non-anticipativity constraint. When executing a policy, a schedule is constructed as time progresses (i.e., activity starting times become known gradually). Consequently, a policy Π may be interpreted as a function $\mathbb{R}_{\geq 0}^{n+1} \mapsto \mathbb{R}_{\geq 0}^{n+1}$ that maps given realizations of activity durations $\mathbf{p}$ to vectors of feasible starting times $S(\mathbf{p}; \Pi) = \{S(p_0; \Pi), S(p_1; \Pi), \ldots, S(p_n; \Pi)\}$ (see for instance Igelmund and Radermacher (1983), Möhring (2000), and Stork (2001)). For a given realization $\mathbf{p}$ and policy Π, $S_n(\mathbf{p}; \Pi)$ denotes the makespan of schedule $S(\mathbf{p}; \Pi)$. The most common objective of the SRCPSP is to minimize $E(S_n(\mathbf{p}; \Pi))$ over a given class of policies (where $E(\cdot)$ is the expectation operator with respect to $\mathbf{p}$). Optimization over the class of all policies is computationally intractable. Therefore, we restrict our attention to the subclass of policies $\mathcal{P}$ that start activities only at the end of other activities (where activity 0 starts at time 0).

3 Stochastic activity durations

A project network with stochastic activity durations is often referred to as a PERT network, and a PERT network with independent and exponentially-distributed activity durations is also called a Markovian PERT network. For Markovian PERT networks, Kulkarni and Adlakha (1986) have developed an exact method for deriving the distribution of the earliest project completion time using a continuous-time Markov chain (CTMC). Buss and Rosenblatt (1997), Sobel et al. (2009), and Creemers et al. (2010) use the CTMC of Kulkarni and Adlakha (1986) as a starting point to develop scheduling procedures that

maximize an expected-NPV objective. All aforementioned studies, however, assume unlimited resources and exponentially distributed activity durations. In this abstract, we extend the work of Creemers et al. (2010) to accommodate: (1) resource constraints, (2) PH-distributed activity durations, and (3) a minimum-makespan objective.

4 Performance

In order to solve the SRCPSP, we extend the SDP recursion presented in Creemers et al. (2010). They use a SDP recursion to determine the maximum Net Present Value (NPV) when activity durations are exponentially distributed. In this abstract, we extend this recursion and (1) include resource constraints, (2) allow for general activity duration distributions, and (3) apply a minimum-makespan objective rather than a maximum-NPV objective. The SDP recursion is implemented in Visual Studio C++. All tests are performed on an Intel I5 2.53 GHz computer with 4 GB of RAM.

Various data sets are available in the literature. Tsai and Gemmill (1998), Ballestín and Leus (2009), and Ashtiani et al. (2011) assess the performance of their procedures using the instances of the Patterson data set (Patterson 1984). Stork (2001) evaluates his branch-and-bound algorithms on the J30 and J60 instances of the well-known PSPLIB data set (Kolisch and Sprecher 1996). Ballestín and Leus (2009) and Ashtiani et al. (2011) use the J120 instances of the same data set. Golenko-Ginzburg and Gonik (1997) use a single instance with 36 activities to evaluate their two heuristics. The same problem instance is also used in Ballestín and Leus (2009) and Ashtiani et al. (2011). In our experiments, we use the project instances of the Patterson data set and the J30 and J60 instances of the PSPLIB data set. Because our model is designed to solve small-to medium-sized problem instances, we do not use the J120 instances of the PSPLIB data set. We also do not use the example project presented in Golenko-Ginzburg and Gonik (1997) because its activities have a very limited duration variability.

In a first experiment we assume that activity durations are exponentially distributed. Table 1 summarizes the results. It is clear that project networks of up to 32 activities are analyzed with ease. The results also show that networks of 62 activities can often be solved, albeit at a larger computational cost. With respect to the Patterson data set, Tsai and Gemmill (1998) are able to solve 95 out of 110 instances to optimality if activity durations are deterministic. If activity durations are stochastic, optimality cannot be guaranteed. With respect to the J30 and J60 instances of the PSPLIB data set, Stork (2001) is able to solve 179 and 11 out of 480 instances respectively. It is clear that our model performs better. Next, we use the instances of the J30 data set to analyze the impact of different levels of activity duration variability on the performance of our model. Table 1 summarizes the results. The level of activity duration variability determines the complexity of the Markovian PERT network. For values of SCV smaller than 0.5, the size of the state space increases exponentially and computational performance plummets. For moderate-to-high levels of activity duration variability, however, the computational effort is acceptable.

5 Conclusions

In this abstract, we have presented a generic model for the optimal scheduling of activities with stochastic durations subject to precedence- and resource constraints. The model is an extension of the SDP recursion presented in Creemers et al. (2010). A computational experiment has shown that the model performs best when activities have moderate-to-high levels of activity duration variability, and that it can be used to solve project instances

Table 1. Computational performance

	Data set (SCV = 1.0)			Average SCV (J30)		
	Patterson	J30	J60	0.5	1.0	2.0
Instances in set	110	480	480	480	480	480
Instances solved	110	480	197	467	480	467
Avg CPU time (s)	0.00	0.49	3,727	18.6	0.49	18.8
Max CPU time (s)	0.06	13.1	697,368	321	13.1	390
Min CPU time (s)	0.00	0.00	3.07	0.03	0.00	0.03
Avg state-space size (10^3 states)	7.45	539	150,424	50,795	539	50,795
Max state-space size (10^3 states)	136	11,378	975,124	899,011	11,378	899,011

that have up to 62 activities. For this setting, the model outperforms the existing state-of-the-art.

References

1. Ashtiani, B., R. Leus and M.B. Aryanezhad, 2011, "New competitive results for the stochastic resource-constrained project scheduling problem: Exploring the benefits of pre-processing", *Journal of Scheduling*, Vol. 14, pp. 157–171.
2. Ballestín, F., R. Leus, 2009, "Resource-constrained project scheduling for timely project completion with stochastic activity durations", *Production and Operations Management*, Vol. 18, pp. 459-474.
3. Blazewicz, J., J.K. Lenstra and A.H.G. Rinnooy Kan, 1983, "Scheduling subject to resource constraints: Classification and complexity", *Discrete Applied Mathematics*, Vol. 5, pp. 11-22.
4. Buss, A.H., M.J. Rosenblatt, 1997, "Activity delay in stochastic project networks", *Operations Research*, Vol. 45, pp. 126–139.
5. Creemers, S., R. Leus and M. Lambrecht, 2010, "Scheduling Markovian PERT networks to maximize the net present value", *Operations Research Letters*, Vol. 38, pp. 51-56.
6. Demeulemeester, E., W. Herroelen, 2002, *Project Scheduling: A Research Handbook*, AH Dordrecht: Kluwer Academic Publishers Group.
7. Golenko-Ginzburg, D., A. Gonik, 1997, "Stochastic network project scheduling with non-consumable limited resources", *International Journal of Production Economics*, Vol. 48, pp. 29-37.
8. Igelmund, G., F.J. Radermacher, 1983, "Preselective strategies for the optimization of stochastic project networks under resource constraints", *Networks*, Vol. 13, pp. 1-28.
9. Kolisch, R., A. Sprecher, 1996, "PSPLIB - A project scheduling problem library", *European Journal of Operational Research*, Vol. 96, pp. 205–216.
10. Kulkarni, V., V. Adlakha, 1986, "Markov and Markov-regenerative PERT networks", *Operations Research*, Vol. 34, pp. 769-781.
11. Möhring, R.H., 2000, "Scheduling under uncertainty: Optimizing against a randomizing adversary", *Lecture Notes in Computer Science*, Vol. 1913, pp. 15-26.
12. Neumann, K., C. Schwindt and J. Zimmermann, 2003, *Project Scheduling with Time Windows and Scarce Resources*, Berlin: Springer-Verlag.
13. Patterson, J.H., 1984, "A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem", *Management Science*, Vol. 30, pp. 854-867.
14. Sobel, M.J., J.G. Szmerekovsky and V. Tilson, 2009, "Scheduling projects with stochastic activity duration to maximize expected net present value", *European Journal of Operational Research*, Vol. 198, pp. 697-705.
15. Stork, F., 2001, *Stochastic Resource-Constrained Project Scheduling*, PhD thesis, Technische Universität Berlin.
16. Tsai, Y.-W., D.D. Gemmill, 1998, "Using tabu search to schedule activities of stochastic resource-constrained projects", *European Journal of Operational Research*, Vol. 111, pp. 129-141.

Solving Scheduling Evacuation Problem with Mathematical Programming using Preprocessing

Kaouthar Deghdak, Vincent T'kindt and Jean-Louis Bouquard

Laboratory of Computer Science, Team Scheduling and Control (ERL CNRS 6305),
University François Rabelais Tours, 64 Av. J. Portalis, 37200 Tours, France
{deghdak, tkindt, bouquard}@univ-tours.fr

Keywords: Evacuation Scheduling, Time-indexed formulation, Preprocessing.

1 Introduction

Evacuation problems have received increasing attention in the last years. From the literature, it turns out that two main application scenarios have been considered: the scenario of building evacuation and the scenario of region evacuation. Evacuation of buildings is extensively discussed in (Hamacher and Tjandra 2001). Several reviews tackle the evacuation problem for large regions, e.g, (Sbayti and Mahmassani 2006), (Bish 2011) and (Bish *et. al.* 2013).

To analyse, evaluate and find an optimal evacuation plan, several optimization models have been developed. These models are generaly based on the Dynamic Traffic Assignment (DTA) methodology, e.g, (Kwon and Pitt 2005) and (Sbayti 2008)) or on dynamic network flow models, like universal maximum flow, minimum dynamic cost flow, maximum dynamic flow, quickest flow and quickest path, e.g, (Hamacher and Tjandra 2001). Both proposed optimization models aim at minimizing the total evacuation time or to maximize the number of people exiting the damaged area in a given time period.

In this work, we consider the evacuation of people due to a natural disaster, like earthquakes, where residents have to change their centre of lives from several days to several months with the eventual goal of returning back to home. In particular, we assume that the locations of gathering points (where people are evacuated, outside the damaged area), the locations of collection points (where people are gathered waiting to be evacuated) and the capacities of the transportation network are known. The goal is to define a macroscopic plan of evacuation which means that people are considered homogeneously: only common behaviours are taken into account, and we have to transport evacuees from the collection points to the gathering points in a minimal amount of time. The evacuation is assumed to be done by means of a fleet of buses, thus leading to schedule the evacuation operations with a fleet of buses (Bus Evacuation Problem, BEP). Originally, the BEP was introduced and studied in (Bish 2011), where a mathematical programming model and an heuristic are presented. BEP can be modeled as a cumulative scheduling problem with additional resource constraints: the gathering points are cumulative parallel machines and the jobs to schedule are the persons to evacuate. The additional resources are the buses which are disjunctive resources. One other important feature of this evacuation scheduling problem is that the processing time of the jobs are dependent on their starting time in the schedule. This is a consequence of the evolution of the transportation network through time due to events like earthquake replicates and road repairs.

Section 2 describes the evacuation scheduling problem in details and provides a mixed-integer programming formulation. In Section 3, we present a preprocessing method to improve the mathematical programming solution.

2 Problem definition and mathematical formulation

Consider that we have M parallel and cumulative machines, each machine j corresponding to a gathering location GP_j, with a capacity cap_j expressed as a number of buses that can bring evacuees. N evacuation operations have to be scheduled on the machines, and each operation i is associated with a collection point CP_ℓ. Similarly to the calculation of the machine capacity, it is assumed that one evacuation operation coressponds to the evacuation of people by a filled bus. Thus, the set of evacuation operations is deduced from the number of people to evacuate and the capacity of a bus. A fleet of K identical buses is given and used to evacuate people. We suppose that the capacity of the reconfigured network is sufficient after the disaster to enable a fluent evacuation. This hypothesis enables to approximate the bus routing by considering travel times only. An additional assumption is made on the time horizon since, on the original problem, events that modify the transportation network may occur: we assume that k such events happen at a known time d_l and can change the value of processing time of operation i on machine j:

$$p_{i,j,t} = \begin{cases} a^0_{i,j} & \text{if } t_{ij} \in]0, d_1] \\ a^1_{i,j} & \text{if } t_{ij} \in]d_1, d_2] \\ \vdots & \\ a^{k-1}_{i,j} & \text{if } t_{ij} \in]d_{k-1}, d_k] \end{cases}$$

where $a^x_{i,j}$ is the travel time if the evacuees are transported from a collection point i to gathering point j in the x^{th} time interval, i.e. its starting time $t_{ij} \in]d_{x-1}, d_x]$. The number of finite intervals $[d_{l-1}, d_l]$ is determined by a preliminary forecasting of the evolution of the transportation network.

To model BEP we proposed a time-indexed mathematical formulation. Usually, time-indexed formulations on scheduling problems yield to simple and efficient models despite the presence of a pseudo-polynomial number of variables (see (Berghman *et. al.* 2013) among others). Let us turn to the model for our evacuation scheduling problem in which T is the time horizon. The decision variables are:

$$\forall i = \{1, ..., N\}, \forall j = \{1, ..., M\}, \forall t = \{0, ..., T-1\};$$

$$x_{i,j,t} = \begin{cases} 1 \text{ , if a bus starts the evacuation operation } i \text{ towards } j \text{ at } [t, t+1[\\ 0 \text{ , otherwise.} \end{cases}$$

and C_{max}, the duration of the schedule.

The proposed (IP) formulation is as follows :

$$\min C_{max}. \tag{1}$$

Subject to:

$$C_{max} \geq (t + p_{i,j,t})x_{i,j,t} \qquad \forall i = \{1, ..., N\}, \forall j = \{1, ..., M\}, \forall t = \{0, ..., T-1\} \quad (2)$$

$$\sum_{t=0}^{T-1} \sum_{i=1}^{N} x_{i,j,t} \leq cap_j \qquad \forall j = \{1, ..., M\}] \quad (3)$$

$$\sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{\substack{t' \in [0,t] \\ p_{i,j,t'} + t' > t}} x_{i,j,t'} \leq K \quad \forall t = \{0, ..., T-1\} \quad (4)$$

$$\sum_{t=0}^{T-1} \sum_{j=1}^{M} x_{i,j,t} = 1 \qquad \forall i = \{1, ..., N\} \quad (5)$$

$$x_{i,j,t} \in \{0, 1\} \quad (6)$$

Constraints (2) guarantee that C_{max} defines the value of the criterion which is then minimized by the objective function (1). Constraints (3) are the gathering point capacity constraints: we cannot exceed the capacities of gathering points. Constraints (4) are the bus capacity constraints: we cannot exceed the number of buses we have. Constraints (5) ensure that each operation is processed once and only once. Constraints (6) are the logical binary restriction on the $x_{i,j,t}$ variables.

3 Preprocessing the time-indexed formulation

To make the solution of the time-indexed formulation faster, a preprocessing method is proposed in this section. The goal of this method is to reduce the size of the instances to be solved. This technique has shown good results for some $\mathcal{NP}$-Hard scheduling problems, ((Baptise *et. al.* 2010) and (Tang *et. al.* 2013)).

The preprocessing technique is based on the relationship between (IP) and its continuous relaxation (LP). Let z^*_{LP} and z^*_{IP} be respectively the optimal solutions (i.e. values of the criterion) of the LP and the IP and let B^* be the basis associated to the optimal solution of the (LP). The preprocessing consists in deducing the value of some boolean variables $x_{i,j,t}$ in an optimal solution of the (IP). We distinguish between the basic variables and the non-basic variables of the optimal solution to (LP).

It is well known in mathematical programming that the following relationship exists between the optimal solution of the (IP) and the optimal solution of the (LP), (applied to our model):

$$z^*_{IP} = z^*_{LP} + \sum_{x_{i,j,t} \notin B^*} c_{i,j,t} x_{i,j,t} \quad (7)$$

with $c_{i,j,t}$ be the reduced cost associated to variable $x_{i,j,t}$. Let UB an upper bound to z^*_{IP} (calculated by an heuristic algorithm). Then, we can be deduced that :

$$UB \geq z^*_{LP} + \sum_{x_{i,j,t} \notin B^*} c_{i,j,t} x_{i,j,t} \Leftrightarrow UB - z^*_{LP} \geq \sum_{x_{i,j,t} \notin B^*} c_{i,j,t} x_{i,j,t} \quad (8)$$

From inequality (8) we can deduce the following fixing rule: $\forall x_{i,j,t} \notin B^*$, if $c_{i,j,t} > UB - z^*_{LP}$ then in any optimal solution of (IP), $x_{i,j,t} = 0$. By reasonning on the slack variables associated to variables $x_{i,j,t}$, we can deduce when $x_{i,j,t} = 1$.

Concerning basic variables $x_{i,j,t} \in B^*$, we use pseudo-costs $l_{i,j,t}$ and $u_{i,j,t}$ associated to variables $x_{i,j,t}$. These two values can be calculated, for instance, by means of Driebeeck's penalties. These penalties correspond to lower estimates on the increase of z^*_{LP} if $x_{i,j,t}$ is set to 0 or 1: they are, typically calculated based on reduced costs of non basic variables. We kindly refer the reader to the paper of (Driebeeck 1966) for a detailed explanation on how these penalities are calculated. We have:

1. $\forall x_{i,j,t} \in B^*$, if $(\ell_{i,j,k} x_{i,j,t}) \geq (UB - z^*_{LP})$ then in any optimal solution of (IP) we have $x_{i,j,t} = 1$;
2. $\forall x_{i,j,t} \in B^*$, if $(u_{i,j,k}(1 - x_{i,j,t})) \geq (UB - z^*_{LP})$ then in any optimal solution of (IP) we have $x_{i,j,t} = 0$.

This variable fixing technique can be used at any node of a search tree, in a Branch & Cut algorithm for instance, as long as the (LP) relaxation and the corresponding node has been solved. The preprocessing algorithm works as follows: (1) Calculate an upper bound for BEP, (2) Solve the (LP) relaxation, (3) Fix some variables $x_{i,j,t}$, finally, the (IP) is solved with the set of fixed variables included.

At the conference, we will present some valid inequalities to restrict the solution space of the problem and to see the efficiency of the preprocessing algorithm. We will also present a computational experiment of these algorithms, and report the size of the instances solved to optimality.

Acknowledgements

This research has been supported by ANR-11-SECU-002-01, project DSS_EVAC_LOGISTIQUE (CSOSG 2011).

References

Baptiste P., F. Della Croce, A. Grosso and V. T'kindt, 2010, "Sequencing a single machine with due dates and deadlines: An ILP-based approach", *Journal of Scheduling*, Vol. 13(1), pp. 39-47.

Berghman L., R. Leus and F. Spieksma, 2013, "Optimal solutions for a dock assignment problem with trailer transportation". *Annals of Operations Research*, Springer US.

Bish D R, H D. Sherali, A G. Hobeika, 2013, "Optimal evacuation planning using staging and routing", *Journal of the Operational Research Society*. Doi:10.1057/jors.2013.3.

Bish D R., 2011, "Planning for a bus-based evacuation". *OR Spectrum*, Springer-Verlag, Vol. 33, pp. 629-654

Driebeeck N J., 1966, "An algorithm for the solution of mixed integer programming problems". *Management Science* , Vol. 12, pp. 576-587.

Hamacher H W. and S A. Tjandra, 2001, "Mathematical Modeling of Evacuation Problems: A State of The Art". *In Pedestrian and Evacuation Dynamics*, Vol. 1964, pp. 227-266.

Kwon E. and S. Pitt, 2005, "Evaluation of emergency evacuation strategies for downtown event traffic using a dynamic network model", *Journal of the Transportation Research Board*, Vol. 1922, pp. 149-155.

Mahmassani H S., 2001, "Dynamic network traffic assignment and simulation methodology in advanced system management applications", *Networks and Spatial Economics*, Vol. 1, pp. 267-292.

Sbayti H and H S. Mahmassani, 2006, "Optimal Scheduling of Evacuation Operations". *Journal of the Transportation Reasearch Board*, Vol. 1964, pp. 238-246.

Sbayti H., 2008, "Optimal Scheduling of Evacuation Operations with Contraflow". PHD Dissertation. Faculty of the Graduate School of the University of Maryland, USA.

Tang X., A. Soukhal and V. T'kindt, 2013, "Preprocessing for a map sectorization problem by means of mathematical programming", *Annals of Operations Research*, DOI 10.1007/s10479-013-1447-8.

A review on Decisions Support Systems for Manufacturing Scheduling

Dios, M.[1] and Framinan, Jose M.[2]

[1] University of Seville, Spain mdios@us.es
[2] University of Seville, Spain framinan@us.es

Keywords: Decision Support System, DSS, production, scheduling.

1 Introduction

There is a widely recognised gap between research and practice in the scheduling field (MacCarthy and Liu 1993). Among the different causes cited, the lack of an integrated view of scheduling has been frequently mentioned (Herrmann 2004), with most research focusing exclusively on technical –i.e. optimisation– aspects of scheduling. In this regard, Decision Support Systems (DSSs) for scheduling have been acknowledged as a key to integrate human and technical perspectives, thus providing a direction to advance in bridging the aforementioned gap. Therefore, within the scheduling field there is a growing interest in DSS, which has produced a number of case-studies and descriptions of implementation of DSSs for manufacturing scheduling. The analysis of these case studies and contributions may serve to identify a number of relevant issues still not properly addressed and thus provide future research lines to close the gap between theory and practice in manufacturing scheduling. In addition, such analysis may provide a retrospective study on which techniques and approaches to model scheduling problem are been succesfully implemented in practice.

The goal of this paper is to review and classify these contributions. To do so, we first carry out a systematic review to identify the relevant papers in the literature. In total, 86 contributions have been regarded as relevant. In order to provide a coherent taxonomy for the analysis of these papers, we develop a classification based in the works by (Monfared and Yang 2007, Framinan and Ruiz 2010, Framinan and Ruiz 2012). More specifically, we focus on the **structure** –or functionalities– of the DSSs reviewed (i.e. what the systems do), and on their **methodology** (i.e. how the systems achieve their functionality). The first aspect is oriented towards the identification of issues not adequately covered up-to-now, and the second aspect is related to analysing the degree of success of the diferent techniques and methods available. Due to space problems, the complete classification will be presented in the conference (the full tables with the classification are available in `http://taylor.us.es/componentes/mdr/PMS/Review_PMS_2014.pdf`), and here we simply briefly discuss the classification criteria and comment some conclusions.

2 Structure of DSS

As mentioned before, the structure of the DSSs refers to their functionalities which are classified here according to the architecture of manufacturing scheduling systems by (Framinan and Ruiz 2010):

- **Scope of the System**, i.e the extent of the decisions supported by the system. Although this paper focuses on manufacturing scheduling (S), some DSSs also address related decisions, most typically Planning (P) and Control(C).

- **Problem Modeling**. This functionality relates to the ability of the system to capture the different constraints and features of the shop floor, which can be facilitated by the so-called Model Detection (MD), i.e. DSS' capabilities to determine the most suitable model for solving a specific scheduling scenario. Another feature is Constraints Abstraction (CA), indicating whether the system can reduce the complexity of the models by means of e.g. aggregating constraints.
- **Problem Solving**. This functionality relates to the ability of the system to solve the models. The following specific features can be identified (Framinan and Ruiz 2010):
  - Algorithms for Rescheduling (AR), which refers to the capability of the DSS for reacting to disturbances by applying the corresponding algorithms.
  - Multi Algorithm (MA) scheduling, a feature allowing the decision maker to compare the different solutions and choose the one fitting his/her objectives better.
  - Evaluation of Algorithms (EA) can be seen as a refinement of the previous feature, as the system suggests the planner which one is the best algorithm available.
  - Generation of New Algorithms (GNA), meaning the capability of the system to embed new algorithms.
  - Incorporation of Human Expertise (HE), indicating that the DSS allows the decision maker to incorporate his/her expertise in some manner.
- **Solution Evaluation**. This functionality refers to the ability of the system to present the solutions from different points of view so the decision maker can analyse them. Different features can be considered:
  - Different Objectives (DO). Note that this feature does not refer to considering different objectives in the solution procedure, but on evaluating the resulting schedules with respect to different objectives.
  - Analysis of Scenarios (AS) offers the decision maker the possibility of comparing different solutions obtained from the DSS. By means of this feature the decision maker can modify the input data of the DSS to see what happens if, for example, there are more customer orders or if the duration of a task in a specific machine is increased.
- **User Interface**. In this functionality the DSSs show the resulting schedules to the user. Different charts and graphs can be used, including Gantt Charts (GC), Job Screens (JS), Machine Loading Boards (MLB), Textual Information (T) or other kinds of charts or diagrams (OC).

3 Methodology of DSS

With respect to the methodology adopted in the different DSSs in order to provide the functionalities described in Section 2, we use the classification by (Monfared and Yang 2007). In their work, three different levels of methodologies are described, i.e.: supporting discipline, major approaches, and techniques. In the first level – disciplines–, they consider Computer Science (CS), Operations Research (OR), and Control Theory (CT). For each supporting discipline, one or more of the different major approaches can be adopted: Optimization Techniques (OT), Artificial Intelligence (IT), Simulation (S) and Neural Networks (NN). Finally, within each approach, different techniques can be applied:

- Regarding Optimization Techniques, we distinguish between Mixed Integer Linear Programming (MILP), and approximate techniques such as metaheuristics, i.e.: Tabu Search (TS), Genetic Algorithms (GA), Simulated Annealing (SA) or Specific Heuristics (SH) developed for the problem.
- Regarding Simulation, we idenfiy some contributions using Discrete Event Simulation (DES) and Queuing Theory (QT).

- Regarding Artifical Intelligence, three different techniques are considered: Expert Systems (ES), Constraint Programming (CP) and Multi-Agents Systems (MAS).
- Regarding Neural Networks, we distinguish between Feed Forward Neural Network (FF) and Multi-Layered Perceptron (MLP).

4 Conclusions

A number of general conclusions that can be drawn are summarised next:

- Regarding the integration of scheduling and related decisions, 28 DSSs address production scheduling and control, in four cases planning and scheduling are simultaneously solved, and in an additional case, scheduling and transport decisions are integrated. We also find two cases where planning, scheduling and control are faced together.For the rest of the cases, manufacturing scheduling is addressed in isolation. This fact speaks for the relatively autonomy of manufacturing scheduling decisions, which certainly eases the development of DSSs.
- When analysing the functional features of the reviewed DSSs, there is a wide diversity in the number and type of features. Problem Modelling is present only in about 27% of the systems, while Problem Solving in more than half of them. Finally, Solution Evaluation and User Interface features are described in around 60% of the systems. A conclusion is that there are few described DSSs addressing the whole process, from modeling to solution representation (only 20%). Most DSSs focus on modelling and solving the models, and do not include information on data management or user interfaces. This makes difficult to transfer the knowledge generated by the authors of these contributions to the real industry.

With respect to the structure of the DSS, the following specific conclusions can be presented:

- Regarding Problem Modelling, only 26 out of the 86 DSSs include some feature related to this aspect. Moreover, only seven references describe a system with capabilities of Model Detection. If this finding is aligned with the fact that most solution procedures in the literature are model-specific, then it is clear that this functionality is clearly an area in urgent need of research to close the gap between theory and practice.
- With respect to Problem Solving, the importance of incorporating human expertise in production scheduling is acknowledged in most DSSs (52%). Algorithms for rescheduling are present in more than 25%. In constrast, the rest of the related features seldomly appear in the DSSs reviewed, all of them referring to algorithms creation, maintenance and evaluation. Since these aspects greatly influence the capability of adapting a DSS to different scheduling scenarios, this factor is probably limiting the expansion of generic manufacturing scheduling DSSs.
- Regarding Solution Evaluation, there is a lack of DSSs dealing with stochasticity, as no single contribution was found in this respect. Additionally, a half of the DSSs give the user the possibility to analyse different scenarios to get insights about how to enhance his schedules, and around one quarter allows the user for selecting different objectives to generate their schedules.
- User Interface. It is particularly difficult to infer information regarding this feature obtain as most of the works do not include screenshots of the DSSs nor descriptions about how these present the information to the decision maker. Based on the available information we obtained that a 33.7% of the results used text to show the schedules while almost the half of them show their results through Gantt Charts and around 10% using Job Screens or Machine Loading Boards. There were some works where the information was offered through different methods.

The following conclusions can be extracted regarding the methodology of the DSS:

- Regarding the supporting discipline, most DSSs in practice do not adopt a single supporting discipline, but rely on several supporting disciplines depending on the part of the DSS. When addressing issues related to database management components or dialogue management components, the most used discipline is Computer Science, whereas for the model management components the most employed discipline is Operations Research. Finally, Control Theory is predominant for those systems including reactive scheduling. This speaks for the need of an integrated approach and for interdisciplinary teams when trying to comprehensively address the design and implementation of DSSs for manufacturing scheduling.
- Regarding approaches for modeling the scheduling problem, there is a strong correlation between the supporting discipline and the approach adopted, although this is not strictly required according to the work by (Monfared and Yang 2007). Perhaps not surprisingly, most DSSs using Optimization Techniques involve minimisation problems, while approaches based on Artificial Intelligence and Neural Networks are oriented towards the obtention of feasible schedules. This reveals an apparent lack of interest from the Operations Research approaches to focus on obtaining feasible (but not neccesarily optimal) schedules, as well as the difficulty for approaches derived from Computer Science to efficiently handle optimisation approaches.

Finally, with respect to the specific techniques employed, several remarks can be done:

- Deterministic techniques are preferred in front of techniques explicitly addressing the stochastic nature of most scheduling problems. While this does not neccessarily mean that such stochastic nature is ignored in most DSSs, it leads to the need of investigating the degree of variability that deterministic techniques can cope with, i.e. how different sources of variability affect the quality of the schedules provided by deterministic techniques.
- The majority (39) of the DSSs use Specific Heuristics for the models, which are obviously difficult to be generalized or applied for different scenarios. This may point out to the need of moving towards at least two directions: 1) the generalisation of *ad hoc* techniques so they can be applied to a broader range of situations, and 2) the development of systematic approaches efficiently building and testing specific heuristics so the –usually high– effort to develop and evaluate heuristics for new scheduling problems can be shortened.

References

MacCarthy, B. L. and Liu, J., 1993, "Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling", *International Journal of Production Research*, Vol. 31(1), pp. 59-79.

Hermann J.W., 2004, "Information flow and decision-making in production scheduling", *IIE Annual Conference and Exhibition 2004*, pp. 1811-1816.

Monfared M.A.S., J.B. Yang, 2002, "Design of integrated manufacturing planning, scheduling and control systems: a new framework for automation", *The international Journal of Advanced Manufacturing Technology*, Vol. 33, pp. 545-559.

Framinan J.M., R. Ruiz, 2010, "Architecture of manufacturing scheduling systems: Literature review and an integrated proposal ", *European Journal of Operational Research*, Vol. 205, pp. 237-246.

Framinan J.M., R. Ruiz, 2012, "Guidelines for the deployment and implementation of manufacturing scheduling systems ", *International Journal of Production Research*, Vol. 50, pp. 1799-1812.

A Multi-Criteria Approach for Ranking Schedules for Multi-Mode Projects

Utkan Eryılmaz[1], Öncü Hazır[1], and Klaus Werner Schimdt[2]

[1]Department of Business Administration, TED University, Ankara
e-mail: {utkan.eryilmaz;oncu.hazir}@tedu.edu.tr

[2] Department of Mechatronics Engineering, Çankaya University, Ankara
e-mail: schmidt@cankaya.edu.tr

Keywords: Scheduling, Robustness, Multi-Criteria Decision Analysis, Data Envelopment Analysis

1. Introduction

In this study, we investigate a multi-mode project scheduling problem that considers a single non-renewable resource (money), namely, the *discrete time/cost trade-off problem* (*DTCTP*). The deadline version of the problem assigns modes to each activity so that the total cost is minimized while respecting the project deadline; whereas the budget version minimizes the project duration given a constraint on total spending. Both versions have application areas in practice as they model the time/cost relationship in processing activities. Small and medium size problems could be solved by exact methods (Demeulemeester et.al. 1996, 1998); whereas approximate solution methods are needed for large size problems (See Hazır et al. 2010a, for a comprehensive literature review and the problem formulation).

Combining the two versions of the DTCTP, we propose to construct non-dominated time/cost solutions and rank them regarding robustness. We define robust solutions as the schedules which are insensitive regarding deviations in activity durations. To evaluate robustness, we propose a new measure in section 2.2. Even if cost and completion time are the most widely investigated criteria in academic literature; in real life, developing robust schedules becomes critical to hedge against uncertain events (Hazır et al. 2010b).

To rank the solutions we use *super-efficiency models* (Andersen and Petersen 1993) of *data envelopment analysis (DEA)* [Charnes, Cooper and Rhodes (CCR), 1978]. DEA models do not require defining explicit weights for criteria, which are difficult to obtain in real life and may not be uniform in the feasibility region. In the literature, there are multi-objective resource constrained project scheduling studies. Viena and Sousa (2000) applied metaheuristics to optimize project completion time, mean weighted lateness and resource availability. Balestin and Blanco (2011) obtained the set of non-dominated solutions based on completion time and tardiness. They also proposed measures to express quality of these sets and compare the techniques based on these measures. Differently, we focus on ranking. To the best of our knowledge, this study is the first multi-criteria optimization research that integrates robustness in multi-mode project scheduling.

2. Methodology

We develop a two-stage algorithm. First, we obtain a set of approximately efficient solutions and then order them based on total cost, completion time and robustness.

2.1. Approximately Non-Dominated Solutions

Hapke et al. (1998) used the term *approximately non-dominated solutions* to describe the set obtained through their heuristic search. We use that term for schedules in ϵ_b and ϵ_c neighborhood regarding budget and project completion time (B, C_{max}); where ϵ_b and ϵ_c represent tolerance levels. According to our definition, for an approximately efficient solution (B, C_{max}) there exist no feasible solutions with $\{(B, C_{max}'): C_{max}' < C_{max} - \epsilon_c\}$ and $\{(B', C_{max}): B' < B - \epsilon_b\}$. To generate solutions to the *DTCTP*, we use direct CPLEX modeling and Benders Decomposition for larger instances. Setting the optimality gap around 2% and truncating the branch & bound, solving the problems iteratively produces high quality solutions quickly (Hazir 2010a).

2.2. Robustness Measure

To measure robustness, we use *total slack* (*TS*), the amount of time by which the activity completion can be delayed without delaying the project completion time, based measures. For DTCTP, several slack based functions have been already formulated and tested using simulation (Hazir et al. 2010b). We formulate a new robustness measure (RM), an exponential utility function that uses the ratio of slack (TS_i) to activity duration (p_i), and represents the average value over n activities. Figure 1 shows that when TS is equal to the activity duration (SDR=1), the contribution of additional slack becomes minimal. Hence, to maximize RM, considering the durations, slacks should be distributed evenly among activities.

$$RM = \frac{\sum_{i=1}^{n} 1 - e^{-2SDR_i}}{n}$$

$$SDR_i = \frac{TS_i}{p_i}$$

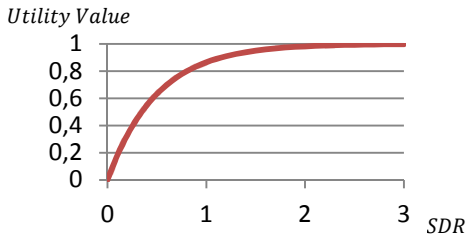


Figure 1. Utility Function

2.3. DEA as a Multi Criteria Ranking Tool for Alternatives

DEA is used for ranking decision making units that include multiple inputs and/or outputs. Every unit maximizes its own ranking based on most favorable weights for inputs and outputs for itself rather than fixed values (CCR, 1978). DEA has been applied to evaluate the performance of different types of organizations in different sectors such as education, health care, banking and services. Efficiency of a unit is calculated by dividing the aggregated value of all outputs (y_{kj} : j[th] output of the k[th] unit) to the aggregated value of inputs (x_{ks} : s[th] input of the k[th] unit). None of the units will have efficiency value greater than 1. The DEA CCR model is formulated below:

$$\max h_k = \frac{\sum_{j=1}^{J} v_j y_{kj}}{\sum_{s=1}^{S} u_s x_{ks}} \quad (1)$$

subject to:

$$\frac{\sum_{j=1}^{J} v_j y_{ij}}{\sum_{s=1}^{S} u_s x_{is}} \leq 1; \;\; i = 1, \dots, I \quad (2)$$

$$v_j,\ u_s \geq 0; j = 1,\dots,J,\;\; s = 1,\dots,S \quad (3)$$

For each unit $k = 1,\dots,I$, this model is solved maximizing the efficiency (h_k). Input (x_{ks}) and output values (y_{kj}) are assumed to be known. Decision variables of the model are the weights for S inputs (u_s) and J outputs (v_j). Note that the model can easily be converted to a linear program (LP) by equating the denominator of the objective function to unity.

The interactive approach of Belton and Vickers (1999) was one of the first studies using DEA as an MCDM tool. Stewart (1996) analyzed the correspondence between the efficiency definition and distance to the Pareto frontier. Joro et al. (1998) showed the connection between DEA and LP by formulating DEA as a reference point model. Eryilmaz and Karasakal (2006) combined DEA with outranking methods and ranked MBA programs using published data.

We use the *super efficiency approach* to obtain distinct scores for efficient units. In the original DEA all efficient units are scored 1 and cannot be differentiated. However, super efficiency score is calculated using the radial distance of the current unit to the efficiency frontier excluding itself; the constraint is valid for $i \neq k$ *(Eq.4)*; the unit under evaluation can get a score greater than 1.

$$\frac{\sum_{j=1}^{J} v_j y_{ij}}{\sum_{s=1}^{S} u_s x_{is}} \leq 1; \;\; i \neq k, i = 1, \dots, I \quad (4)$$

DEA serves as a practical approach for selection among schedules: First, there is a non-linear and complex relation among duration, cost and robustness which is difficult to formulate and solve analytically. Moreover, marginal improvement in robustness (utility) is a decreasing function of

slacks, which depend on the schedule (mode selection). Secondly, there is positive correlation between input and output variables: given the budget (completion time), robustness could be increased by increasing the completion time (budget). In both cases, some of the activities would include more slacks hence flexibility. However inputs and outputs cannot be easily measured by a common concrete unit (such as money) and therefore cannot be aggregated.

2.4. Algorithm for Ranking the Non-Dominated Schedules

Stage 1: Generating the set of approximately non-dominated schedules

We use the data set of Akkan et al. (2005). Given the budget, solutions to the DTCTP (budget version) instances are solved. In each iteration, for a given budget, 10 feasible solutions within an optimality gap [0.25-2.5 %] and the optimal one are recorded. This process is replicated for budget values above [10%, 20%] the minimal total cost, with discrete steps of 0.5 %). Therefore, 231 (21*11) schedules with associated duration and cost (B^k, $C_{max}{}^k$) are recorded.

Stage 2: Ranking the Schedules using DEA

In this stage, the *RM* is calculated and integrated as output score for each schedule generated in stage 1. Cost and project completion time are taken as input values and robustness as output (for the k^{th} schedule, $x_{k1} = B^k$ and $x_{k2} = C_{max}{}^k$, and $y_k = RM^k$). The super-efficiency method is then suitable to present the decision maker schedules that have distinct properties compared to other solutions. We use the tool EMS tool [Scheel, 2000] for calculations.

The first 10 ranked solutions for a problem instance with 136 activities (complexity index, CI=14, coefficient of network complexity, CNC=8, concave cost function and between 2 and 10 modes per activity) are given in Table 1. The efficient solutions have diverse characteristics: the 4th schedule has a favorable budget value, whereas the completion time for 1st. 3th places are balanced in both criteria. Even the 2nd schedule is worse than the 1st schedule regarding the completion time, it is significantly better in budget and slightly in robustness. Figure 2a illustrates the time/cost trade-off and the generated schedules. The efficient ones are highlighted as red in Figure 2b and 2c. Note that as the budget increases, we can generate left shifted schedules, which results in smaller slacks regarding project completion time and hence the RM decreases.

Table 1. Ranked schedules using DEA

Rank	B	Cmax	RM	Efficiency
1	29817	523	0.657	104.35
2	23022	590	0.686	101.41
3	17386	704	0.727	100.86
4	15928	741	0.730	100.69
5	19568	654	0.710	100.47
6	19498	651	0.704	99.53
7	15902	739	0.724	99.39
8	20333	639	0.697	99.04
9	16441	723	0.718	98.94
10	17392	702	0.717	98.92

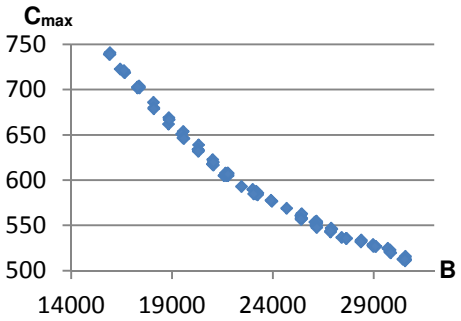


Figure 2a. Budget vs Completion Time

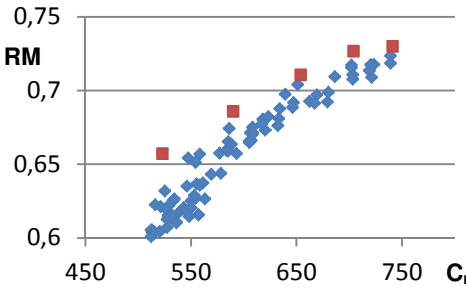


Figure 2b. Robustness Measure vs. Completion Time

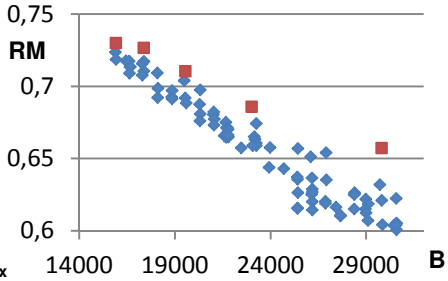


Figure 2c. Robustness Measure vs. Budget

3. Conclusion

In this research, time/cost relations in project scheduling are investigated and robust solutions are sought. An algorithmic basis for a decision support system (DSS) that support project managers is under development. Using the algorithm, a set of high quality project schedules are determined and ranked using DEA analysis. Therefore project managers can concentrate on only a few alternative schedules that will be the basis in planning to complete projects within time and cost targets. To validate the effectiveness and efficiency of our approach, extensive computational experiments and statistical analysis will be performed in future work.

Acknowledgements
This study was supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under grant SOBAG 113K245.

References

Akkan C., A. Drexl and A. Kimms, 2005, "Network Decomposition-Based Benchmark Results for the Discrete Time–Cost Trade-off Problem", European Journal of Operational Research 165, pp. 339-358.

Andersen P., N. C. Petersen, 1993, "A procedure for ranking efficient units in data envelopment analysis", Management Science, 39.10, pp. 1261-1264.

Ballestín F., R. Blanco, 2011, "Theoretical and practical fundamentals for multi-objective optimisation in resource-constrained project scheduling problems", Computers & Operations Research, 38.1, pp. 51-62.

Belton V., S. P. Vickers, 1993, "Demystifying DEA-a visual interactive approach based on multiple criteria analysis", Journal of the Operational research Society, pp. 883-896.

Charnes A., W. W. Cooper, and E. Rhodes, 1978, "Measuring the efficiency of decision making units", European journal of operational research, 2.6, pp. 429-444.

Cooper W.W., L.M. Seiford, and J. Zhu (eds.), 2011, Handbook on data envelopment analysis, Springer Science+ Business Media.

Demeulemeester E. L., W. S. Herroelen, and S. E. Elmaghraby, 1996, "Optimal procedures for the discrete time/cost trade-off problem in project networks", European Journal of Operational Research, 88.1, pp. 50-68.

Demeulemeester E.L., B. De Reyck, B. Foubert, W. Herroelen, and A.M. Vanhoucke, 1998, "New computational results on the discrete time/cost trade-off problem in project networks", Journal of the Operational Research Society, pp. 1153-1163.

Eryılmaz U., E. Karasakal, 2006, "A Hybrid Ranking Method Based on DEA and Outranking Methods", 18th International Conference on Multiple Criteria Decision Making", pp. 81.

Hapke M., J. Andrzej, and R. Słowiński, 1998, "Interactive analysis of multiple-criteria project scheduling problem", European Journal of Operational Research, 107:2, pp. 315-324.

Hazır Ö., M. Haouari, and E. Erel, 2010a, "Discrete time/cost trade-off problem: A decomposition-based solution algorithm for the budget version. "Computers & Operations Research, 37:4, pp. 649-655.

Hazır Ö., M. Haouari, and E. Erel, 2010b, "Robust scheduling and robustness measures for the discrete time/cost trade-off problem", European Journal of Operational Research, 207:2, pp. 633-643.

Joro T., P. Korhonen, and J. Wallenius, 1998, "Structural comparison of data envelopment analysis and multiple objective linear programming", Management Science, 44.7, pp. 962-970.

Scheel H., 2000, "EMS (Efficiency Measurement System) A Data Envelopment Analysis (DEA) Software", Available [online] http://www.holger-scheel.de/ems/

Stewart T. J., 1996, "Relationships between data envelopment analysis and multicriteria decision analysis", Journal of the Operational Research Society, pp. 654-665.

Viana A., J. P. de Sousa, 2000, "Using metaheuristics in multiobjective resource constrained project scheduling", European Journal of Operational Research, 120.2, pp. 359-374

A new fast heuristic to minimize flowtime in permutation flowshops

Victor Fernandez-Viagas[1]and Jose M. Framinan[2]

[1] Industrial Management, School of Engineering, Spain
vfernandezviagas@us.es
[2] University of Seville, Spain
framinan@us.es

Keywords: Scheduling, Flowshop, Heuristics, Flowtime.

1 Introduction

The flowshop scheduling problem involves determining the sequence in which n jobs are processed on m machines in the same order. It is usually assumed that the job sequences will be the same on every machine (permutation flowshops), along with other hypotheses such as e.g. the simultaneous availability of all jobs and of all machines, deterministic processing times, etc (RA Dudek and OF Teuton 1964). Among the objectives that can be considered, the minimisation of the sum of the completion times of the jobs –or $F|prmu|\sum C_j$, according to the notation by Graham, R. L. *et. al.* (1979)– has been consistenly pointed out both as relevant and meaningful for today's dynamic production environment (J Liu and CR Reeves 2001). This problem is known to be NP-hard, therefore most of the research on this topic is devoted to developing approaches yielding good (but not necessarily optimal) solutions in reasonable computation time. Recently, Pan, Q.-K. and Ruiz, R. (2013) exhaustively evaluate the different heuristics in the literature taking into account the quality of the solutions (measured as the average relative percentage deviation over the best known solution) and the CPU time (in seconds). Using these two indicators as in a bicriteria decision problem, they derived a set of 14 non-dominated (i.e. approximation of a Pareto set) heuristics, which can be considered state-of-the-art for the problem. A detailed analysis of this Pareto set reveals that 12 out of the 14 heuristics employ a mechanism for constructing the solutions based in the heuristic by J Liu and CR Reeves (2001). In this paper, we propose a heuristic with complexity $O(n^2m)$ that outperforms the heuristic by J Liu and CR Reeves (2001) both in terms of quality of the solutions and in CPU time. By embedding this new heuristic in several composite heuristics in the Pareto set, we obtain a completely new efficient set.

2 Problem Statement

The problem under consideration can be stated as follows: n jobs have to be scheduled in a flowshop consisting of m machines. On each machine i, each job j has a processing time denoted as p_{ij}. The completion time of job j on machine i is denoted as C_{ij}, whereas $C_{i[j]}$ indicates the completion time on machine i of job scheduled in position j. C_{mj} represents the completion time of job j.

As mentioned before, a great number of heuristics have been proposed for the problem. Pan, Q.-K. and Ruiz, R. (2013)exhaustively evaluate all these heuristics in terms of both the quality of the solutions and computational requirements, and depict a Pareto set to place the efficient heuristics for the problem in view of their performance on the well-known Taillard's testbed. This Pareto set is formed by the following heuristics: Raj, $LR(1)$, RZ, $LR-NEH(5)$, $LR-NEH(10)$, $LR-NEH(15)$, $LR-FPE$, $PR4(5)$, $PR2(5)$, $PR3(5)$,

$PR4(10)$, $PR4(15)$, $PR2(15)$ and $PR1(15)$. From the analysis of the Pareto set, at least two conclusions can be derived: 1) Each heuristic in the Pareto set has at least a complexity of $n^3 \cdot m$; and 2) all the efficient heuristics consist on variation/adaptations of the following five main (or primary) procedures: NEH, $LR(x)$, FPE, iRZ and VNS. More specifically, the $LR(x)$ heuristic is present in 12 of the 14 heuristics in the Pareto set.

From these conclusions, it can be seen that LR is a key heuristic of complexity $O(n^3 \cdot m)$, playing a role similar to that of the NEH for makespan minimisation. For this latter problem, Taillard, E. (1990) showed that the complexity of the NEH can be reduced from $O(n^3 m)$ to $O(n^2 m)$ by using an acceleration mechanism, but unfortunately, such mechanism cannot be used to minimize flowtime. The only acceleration proposed is due to Li *et. al.* (2009), who reported savings in the CPU time around 30-50%. Nevertheless, the complexity of the NEH remains the same and thus a way to reduce the complexity of efficient approximate algorithms for flowtime to $O(n^2 m)$ has remained elusive.

3 The proposed heuristic

The proposed constructive heuristic –denoted as $FF(x)$– iteratively selects one job after another at the end of the sequence. When introducing a new job at the end of the sequence, there are three elements to be considered:

- *Idle time induced by the newly inserted job*, which influences the next jobs to be inserted. Clearly, this influence decreases with each step (being 0 in the last step). Its calculation has a complexity $O(m)$.
- *Completion time in machine m of the newly inserted job*, which affects the total flowtime since the completion time of each job in machine m is included in the objective function. This data can be calculated within $O(m)$ using the completion time on each machine of the preceding job.
- *Completion time in machine m of the artificial job.* It is thus convenient to ensure that the unscheduled jobs will not have a very large completion time in machine m. However, the calculation of this completion time has a complexity of $n \cdot m$.

Since choosing the adequate position of a job is critical, the main issue lies in weighting the influence of the aforementioned elements. To do so, we use two parameters (a and b), to balance the first two elements, i.e. idle time and completion time of the newly inserted job. In contrast, we leave aside the third element (completion time of the artificial job), since its influence in the objective function is not as direct as the other two elements, and its consideration would increase the complexity of the algorithm to $n^3 \cdot m$.

More specifically, the proposed heuristic is as follows:

1. Sort the jobs according to a non descending order of indicator ξ'_{j0} (eq. 1), breaking ties in favor of jobs with higher $IT'_{j,0}$ (eq. 2). Let us denote by I the so-obtained vector
2. Obtain x partial sequences π^i $(i = 1, \ldots, x)$ of length 1, where the first (and only) job of sequence π^i is the job in position i in I. Store in U^i the jobs not scheduled in π^i.
3. For $k = 2$ to n:
   (a) For each partial sequence π^i, remove from U^i the job for which the minimum value of $\xi'_{j,k}$ (see equation 1) is found and place it in the last position of π^i.
4. Return the (final) sequence π^i yielding the lowest completion time.

Therefore, the proposed procedure begins with x sequences (π^i with $i \in [1, x]$) with only one job. The first job of each sequence π^i is the job in position i of a vector sorted in non descending order of indicator ξ'_{j0} (equation 1) breaking ties in favor of jobs with

higher $IT'_{j,0}$ (equation 2) and each final sequence π^i is obtained adding one by one jobs to the last position of the vector.

Let us denote by k the size of the vector in each step. To insert a new job j ($j \in U^i$) in each sequence π^i, one of the unscheduled jobs of each sequence, U^i, is removed according to an ascending index of a complexity m, $\xi'_{j,k}$. This index is based on $IT'_{j,k}$ the weighted idle time between the job in position k and the new job j to be inserted, and on the makespan of the sequence when inserting job j, $C_{m,j}$. For each job $j \in U^i$, $\xi'_{j,k}$ is calculated as follows:

$$\xi'_{j,k} = \frac{(n-k-2)}{a} \cdot IT'_{j,k} + AT'_{j,k} \tag{1}$$

where $AT'_{j,k}$ and $IT'_{j,k}$ are defined as follow:

$$IT'_{j,k} = \sum_{i=2}^{m} \frac{m \cdot max\{C_{i-1,j} - C_{i,[k]}, 0\}}{i - b + k \cdot (m - i + b)/(n-2)} \tag{2}$$

$$AT'_{j,k} = C_{m,j} \tag{3}$$

Note that by avoiding the calculation of the completion time of the artificial job p (C_{mp}), the complexity of the algorithm decreases from $n^3 \cdot m$ to $n^2 \cdot m$, a complexity n times lower than the fastest heuristics in the efficient set by Pan, Q.-K. and Ruiz, R. (2013).

In order determine the best values for parameters a and b, a multi-factor Analysis of Variance (ANOVA) was performed with four factors (n, m, a and b) where $a \in \{1, 2, 3, 4\}$, $b \in \{0, 0.5, 1\}$. Factors n and m are those in the testbed by Taillard, E. (1993), which is employed to perform the analysis. More specifically, $n \in \{20, 50, 100, 200, 500\}$ and $m \in \{5, 10, 20\}$. The best combination of parameters is found using $a = 4$ and $b = 1$, so these values are used for the new set of heuristics.

4 Computational experience and evaluation

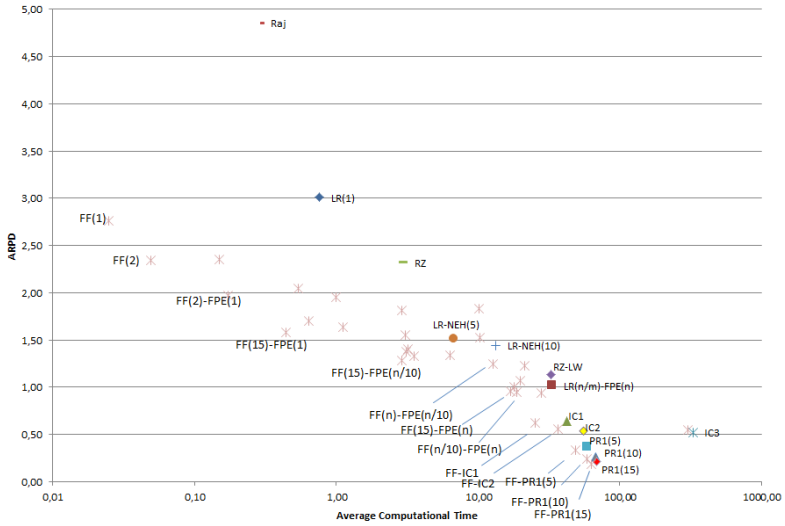


Fig. 1. New Pareto Set of Heuristics. X-axis is shown in logarithmic scale.

In order to compare the performance of our proposal, the efficient heuristics described by Pan, Q.-K. and Ruiz, R. (2013) are implemented and their results on the benchmark set of Taillard, E. (1993) are collected. Since Li *et. al.* (2009) showed that the CPU time of heuristics with insertion and pair-wise exchanges can be reduced by 30-50%, this acceleration has been implemented. The overall results –summarised in Table 1– show the efficiency of the proposed heuristic. For instance, the average CPU time of $FF(1)$ is $0.02s$ while that for $LR(1)$ is $0.76s$. Not only the complexity of the algorithm has been reduced from $n^3 \cdot m$ to $n^2 \cdot m$, but the $ARPD$ of $FF(1)$ is also lower as compared to that of $LR(1)$. By replacing LR by our proposal in the up-to-now efficient set, a new set of 13 efficient heuristics (all of them using FF) can be identified (see Fig. 1).

Table 1. Comparisons between composite heuristics which include LR and FF heuristics

Heuristic	*ARPD*	**Avg. Time**		**Heuristic**	*ARPD*	**Avg. Time**
LR(1)	3.01	0.76	→	FF(1)	2.76	0.02
LR(n/m)-FPE(n)	1.02	33.07	→	FF(n/m)-FPE(n)	1.01	18.05
IC1	0.64	41.93	→	FF-IC1	0.62	25.33
IC2	0.54	55.33	→	FF-IC2	0.56	36.47
IC3	0.53	330.92	→	FF-IC3	0.55	300.93
LR-NEH(5)	1.52	6.69	→	FF-NEH(5)	1.40	3.18
LR-NEH(10)	1.44	13.37	→	FF-NEH(10)	1.34	6.33
Raj	4.86	0.29		—	—	—
RZ	2.32	2.97		—	—	—
RZ-LW	1.13	32.69		—	—	—
PR1(5)	0.37	58.87	→	FF-PR1(5)	0.34	48.60
PR1(10)	0.26	67.38	→	FF-PR1(10)	0.24	58.48
PR1(15)	0.21	68.54	→	FF-PR1(15)	0.19	63.03

References

Li, Xiaoping and Wang, Qian and Wu, Cheng, 2009, "Efficient composite heuristics for total flowtime minimization in permutation flow shops", *Omega*, Vol. 37, pp. 155-164.

Taillard, E., 1993, "Benchmarks for basic scheduling problems", *European Journal of Operational Research*, Vol. 64, pp. 278-285.

Taillard, E., 1990, "Some efficient heuristic methods for the flow shop sequencing problem", *European Journal of Operational Research*, Vol. 47, pp. 65-74.

Pan, Q.-K. and Ruiz, R., 2013, "A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime", *Computers and Operations Research*, Vol. 40, pp. 117-128.

A.J. Vakharia, U. Wemmerlov, 1990, "Designing a cellular manufacturing system: a materials flow approach based on operation sequences", *IIE Transactions*, Vol. 22, pp. 84-97.

L.J. Krajewski, B.E. King, L.P. Ritzman and D.S. Wong, 1987, "Kanban, MRP, and shaping the manufacturing environment", *Management Science*, Vol. 33, pp. 39-57.

R.H. Storer, S.D. Wu and R. Vaccari, 1992, "New search spaces for sequencing problems with application to job shop scheduling", *Management Science*, Vol. 38, pp. 1495-1509.

RA Dudek and OF Teuton, 1964, "Development of m stage decision rule for scheduling n jobs through m machines", *Operations Research*, Vol. 12, pp. 471.

J Liu and CR Reeves, 2001, "Constructive and composite heuristic solutions to the $P||\sum C_i$ scheduling problem", *European Journal of Operational Research*, Vol. 132, pp. 439-452.

Graham, R. L. and Lawler, E. L. and Lenstra, J. K. and Rinnooy Kan,A. H. G., 1979, "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey", *Annals of Discrete Mathematics*, Vol. 5, pp. 287-326.

Exploring heuristic solutions for the stochastic flowshop problem

Jose M Framinan[1] and Paz Perez-Gonzalez[1]

University of Seville, Spain
framinan@us.es; pazperez@us.es

Keywords: Scheduling, Stochastic, Makespan Objective.

1 Introduction

In the flowshop scheduling problem with makespan objective (denoted as $Fm|prmu|C_{max}$), a classical assumption is that the processing times of each job in each machine are known in advance. In contrast, in our paper this assumption is relaxed so these processing times are not deterministic, but follow some known distribution. The objective considered is that of minimizing the expected makespan. we will denote our problem as $Fm|prmu|E[C_{max}]$. This problem has been much less studied, and it is clearly much more complex. In fact, apart from a dominance rule obtained by Makino (1965) for $n = 2$, no exact solution is available without assumptions on the distribution of the processing times. For $m = 2$ and exponential distribution of the processing times, Talwar (1967) conjectured an exact solution known as Talwar's rule. For the two-machine case, three approximate solutions have been proposed by Baker and Trietsch (2011), all of them with similar (near optimal) performance. For the general m machine case, Baker and Altheimer (2012) suggest three heuristics based on adaptations of the CDS (Campbell et al. 1970) and NEH (Nawaz et al. 1983) heuristics with similar (near optimal) performance. In view of these results it might seem that $Fm|prmu|E[C_{max}]$ is already solved, some issues have to be discussed:

- The evaluation of sequences in an stochastic flowshop is far from being a trivial task, as $E[C_{max}]$ has to be estimated by running N simulations using the sequence as a solution, and obtaining a sample mean $\bar{C}_{max}$ of size N. However, there is no standardised procedure to determine N, e.g. Baker and Altheimer (2012) uses a fixed value of $N = 100,000$, while Gourgand et al. (2003) employs $N = 200,000$. In addition, there is no way to ensure the statistical significance of the so-obtained $\bar{C}_{max}$ and, consequently, to establish the significance of the differences in the performance of the heuristics. We propose a procedure to address this issue in Section 2

- The need of specific stochastic heuristics has not been established yet, as the stochastic performance of sequences obtained from applying (deterministic) heuristics to a deterministic flowshop with processing times equal to the means of the processing times has to be tested. Even if it is expected that these heuristics are outperformed, it would be interesting to quantify the degree of variability for which using them is still acceptable. This evaluation is addressed in Section 3.

2 A procedure for the estimation of the expected makespan

Our proposal to estimate the expected makespan of a sequence is based on the maximum percentual error accepted for the estimation of $E[C_{max}]$. More specifically, the half-width of a confidence interval for $E[C_{max}]$ of $1-\alpha$ confidence level is given by $t_{\alpha/2,N-1}\frac{s}{\sqrt{N}}$, where

s is the sample standard deviation of the makespan, and $\alpha/2$ is the area of a Student's t-distribution with $N-1$ degrees of freedom left in the interval $(-\infty, t_{\alpha/2,N-1}]$. We intend that $t_{N-1,\alpha/2}\frac{s}{\sqrt{N}} \leq \bar{C}_{max} \cdot p$, where p is a (small) percentage. By doing so, $E[C_{max}]$ is confined in the interval $[\bar{C}_{max}(1-p), \bar{C}_{max}(1+p)]$ with a $1-\alpha$ confidence level. If we set α to a very low value (in our experiments, $\alpha = 0.001$), we can be almost sure (statistically speaking) that p represents the percentual error of the estimation of $E[C_{max}]$ and use it to check the significance of different results obtained by several heuristics. Note that the normality assumption of the sum of the sample values of makespan required to use this confidence interval seems a loose restriction given the high number of samples that would be run in practice, and the fact that the result of each run is independent from the others. In order to check the number of simulation runs required by this procedure for different degrees of variability and percentage error p, we obtain the estimations of the expected makespan of a random sequence for each instance in the testbed. The results are shown in Table 1. From these results, it can be seen that the number of runs varies greatly depending on the percentage error accepted. Allowing a 5% error means that results can be obtained with less than 10,000 runs, but a 0.5% error requires more than 400,000 runs even for instances with small variability, and around 1,000,000 for $c = 0.5$. The results show that it is difficult to be confident in the results obtained for the number of simulation runs employed in the literature.

Table 1. Estimation of the expected makespan for the testbed: number of simulation runs required

		$p = 0.005$, several c values				$p = 0.01$, several c values				$p = 0.05$, several c values			
n	m	0.01	0.1	0.2	0.5	0.01	0.1	0.2	0.5	0.01	0.1	0.2	0.5
5	2	431975	445822	496753	2023060	107999	111461	124159	500291	4327	4465	4981	17707
5	5	432423	440565	469612	1357063	108111	110146	117412	324274	4332	4413	4707	12420
5	10	432690	437698	455579	1013767	108178	109431	113899	247341	4334	4385	4559	9438
5	20	432856	435632	445717	787848	108220	108914	111445	197201	4336	4364	4464	8418
10	2	432495	440274	469076	1387634	108129	110069	117269	333995	4332	4411	4701	12204
10	5	432679	437646	455673	1001419	108175	109416	113923	244879	4334	4384	4561	9358
10	10	432818	436128	447910	804226	108210	109040	111982	201065	4336	4369	4487	7671
10	20	432911	434916	442260	691900	108233	108734	110568	173496	4336	4357	4430	6724
15	2	432686	437973	457365	1052151	108177	109495	114367	263858	4334	4387	4579	9512
15	5	432793	436369	449396	870760	108204	109097	112343	215040	4335	4371	4501	7987
15	10	432870	435250	444120	725289	108223	108816	111031	181546	4336	4360	4447	7498
15	20	432949	434544	440394	641138	108243	108642	110100	158538	4337	4353	4411	6360
20	2	432788	436753	451444	949730	108202	109194	112862	239675	4335	4375	4521	8842
20	5	432856	435628	445883	776576	108219	108912	111467	194368	4336	4364	4466	7177
20	10	432914	434841	442099	689996	108234	108716	110532	172757	4337	4356	4430	6858
20	20	432968	434284	439143	613362	108248	108577	109790	154052	4337	4350	4399	6195
Avg.		432729	437145	453276	961620	108188	109291	113322	237649	4335	4379	4540	9023

3 Comparison of heuristics

Next, we carry out a computational study to establish the performance of different heuristics for the problem according to the procedure for the estimation of the expected makespan presented in the previous section. The heuristics tested are the following: *SNEH* the stochastic version of the NEH heuristic as described in Baker and Altheimer (2012), *SCDS/Talwar* the stochastic version of the CDS/Talwar heuristic as

described in Baker and Altheimer (2012), NEH the deterministic NEH heuristic applied using as data the mean processing times of the instances, $CDS/Talwar$ the deterministic CDS/Talwar heuristic applied using as data the mean processing times of the instances, and $NEH-Talwar$ the deterministic NEH heuristic applied using as data the mean processing times of the instances, but using as initial order that given by the deterministic CDS/Talwar heuristic.

For all of these heuristics requiring the estimation of $E[C_{max}]$, the procedure in Section 2 has been employed. Tables 2 to 3 show the results obtained for different values of c and for different problem sizes. Apart from the average values obtained by the estimated makespan for each one of the heuristic (labelled as Avg. in the tables), the average percentage increase of the makespan of each heuristic with respect to that of SNEH is presented (labelled as Δ in the tables).

Several comments can be done on the results obtained. First, regarding the heuristics specifically designed for the stochastic problem, there are significant differences in performance. This result contradicts those obtained by Baker and Altheimer (2012). However,it has to be noted that their way to estimate $E[C_{max}]$ is based on a fixed number of simulation runs and that the value employed (100,000) has been shown to be unsufficient to establish consistent results for medium/large coefficient of variations. In addition, our testbed is of bigger size, a fact that may also explain some differences. Differences in performance between SNEH and SCDS/Talwar decrease with the variability of the testbed, but still are substantial for relatively large coefficient of variations. The explanation may lie in the fact that Talwar's rule is optimal for the exponential distribution, whose coefficient of variation is 1, and therefore its performance improves for instances where c is closer to that value.

With respect to the deterministic heuristics tested, the best performance corresponds to the NEH. It is interesting to note that CDS/Talwar is supposed to incorporate some stochastic considerations in their ordering, however these do not pay off, either as a simple heuristic, or as a starting order for the NEH heuristic. When comparing SNEH and NEH, that differences are very small, in some cases below the error accepted for p (1%). The conclusion is that, for realistic ranges of variability, using NEH with the mean processing times gives an extremely good estimation of the performance of their stochastic counterpart. Since SNEH is highly CPU-intensive, it has to be questioned whether such effort pays off.

References

Baker, K.R. and Altheimer, D., 2012, "Heuristic solution methods for the stochastic flow shop problem", *European Journal of Operational Research*, Vol. 216(1), pp. 172-177.

Baker, K.R. and Trietsch, D., 2011, "Three heuristic procedures for the stochastic, two-machine flow shop problem", *Journal of Scheduling*, Vol. 14(5), pp. 445-454.

Gourgand, M., Grangeon, N. and Norre, S., 2003, "A contribution to the stochastic flow shop scheduling problem", *European Journal of Operational Research*, Vol. 151(2), pp. 415-433.

Campbell, HG, Dudek, RA and Smith, ML, 1970, "Heuristic algorithm for the n job, m machine sequencing problem", *Management Science*, Vol. 16(10), pp. 630-637.

Makino, T., 1965, "On a scheduling problem", *Journal of the Operations Research Society of Japan*, Vol. 8, pp. 32-44.

Nawaz, M., Enscore Jr., E.E. and Ham, I., 1983, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem", *Omega*, Vol. 11(1), pp. 91-95.

Talwar, P.P., 1967, "A note on sequencing problems with uncertain job times", *Journal of Operations Research Society of Japan*, Vol. 9, pp. 93-97.

Table 2. Comparative results of the different heuristics for $c = 0.01$.

		SNEH	SCDS/Talwar		NEH		CDS/Talwar		NEH-Talwar	
n	m	Avg.	Avg.	Δ	Avg.	Δ	Avg.	Δ	Avg.	Δ
5	2	309.848	348.360	13.056	309.978	0.041	348.367	13.059	310.171	0.103
5	5	477.448	550.888	15.905	477.552	0.024	550.878	15.903	477.373	-0.086
5	10	795.812	891.632	12.574	795.833	0.004	891.644	12.576	798.258	0.301
5	20	1372.339	1482.957	8.063	1374.269	0.148	1482.933	8.061	1374.786	0.167
10	2	571.929	632.638	11.423	572.181	0.046	632.649	11.426	572.615	0.136
10	5	720.947	866.235	20.268	721.875	0.119	866.224	20.266	716.499	-0.575
10	10	1049.066	1246.054	18.884	1053.062	0.384	1246.057	18.885	1052.709	0.348
10	20	1673.640	1902.705	13.662	1676.495	0.169	1902.713	13.662	1681.827	0.490
15	2	801.950	857.986	6.943	802.028	0.010	857.982	6.943	805.056	0.406
15	5	1008.644	1206.445	19.737	1008.840	0.017	1206.441	19.736	1013.747	0.536
15	10	1309.681	1574.132	20.336	1312.544	0.223	1574.124	20.335	1324.418	1.134
15	20	1965.180	2261.390	15.063	1981.237	0.833	2261.389	15.063	1972.786	0.375
20	2	1085.429	1155.937	6.582	1085.508	0.008	1155.929	6.582	1088.635	0.309
20	5	1224.466	1486.417	21.419	1228.891	0.368	1486.406	21.418	1240.408	1.305
20	10	1601.085	1939.413	21.186	1609.659	0.548	1939.435	21.187	1615.548	0.926
20	20	2258.262	2658.262	17.759	2269.505	0.501	2658.259	17.759	2274.953	0.746
				15.179		0.215		15.179		0.414

Table 3. Comparative results of the different heuristics for $c = 0.5$.

		SNEH	SCDS/Talwar		NEH		CDS/Talwar		NEH-Talwar	
n	m	Avg.	Avg.	Δ	Avg.	Δ	Avg.	Δ	Avg.	Δ
5	2	744.176	776.915	4.380	748.874	0.657	776.746	4.365	752.611	1.146
5	5	1529.894	1595.708	4.422	1537.664	0.520	1595.128	4.392	1533.510	0.224
5	10	2726.438	2842.900	4.478	2741.956	0.596	2848.047	4.689	2739.926	0.513
5	20	4807.027	4982.471	3.657	4836.312	0.613	4986.449	3.744	4834.410	0.561
10	2	1411.348	1496.158	6.206	1446.847	2.448	1496.660	6.243	1446.062	2.461
10	5	2579.011	2764.617	7.170	2610.984	1.251	2763.582	7.132	2613.878	1.326
10	10	4384.309	4678.766	6.734	4433.741	1.153	4684.903	6.870	4433.034	1.118
10	20	7312.968	7656.029	4.647	7357.016	0.616	7659.549	4.700	7373.893	0.843
15	2	1979.146	2095.786	5.837	2021.776	2.165	2095.632	5.814	2042.227	3.156
15	5	3683.235	3989.336	8.347	3737.412	1.467	3990.114	8.363	3756.730	2.026
15	10	5793.048	6186.216	6.805	5861.646	1.184	6189.823	6.863	5878.701	1.479
15	20	9561.606	10073.656	5.356	9661.007	1.039	10078.809	5.412	9646.360	0.881
20	2	2647.731	2821.413	6.605	2711.862	2.447	2819.596	6.535	2743.518	3.665
20	5	4527.154	4910.429	8.443	4590.192	1.394	4914.857	8.541	4632.402	2.332
20	10	7330.081	7883.050	7.532	7445.274	1.580	7882.278	7.528	7446.577	1.593
20	20	11624.512	12411.446	6.818	11730.277	0.911	12403.771	6.757	11745.989	1.044
Avg.				6.090		1.252		6.122		1.523

Combination between metaheuristics and simulation model for a routing problem

Fabrice Gayraud[1], Laurent Deroussi[1], Nathalie Grangeon[1] and Sylvie Norre[1]

LIMOS UMR CNRS 6158 - Antenne de l'IUT d'Allier, France
{fagayraud, deroussi, grangeon, norre}@moniut.univ-bpclermont.fr

Keywords: routing problem, metaheuristics, simulation, home care.

1 Introduction

This paper deals with a routing problem within the context of home health care. The cares are delivered by nurses, doctors, assistant nurses, physiotherapists, for instance. We explain the problem proposed by a nurse coordinator. This nurse organizes the care workers' routes. Each patient is characterized by an address and a dependency level. This level impacts activity duration for the patients with a high dependency level. The cares have a duration, a time windows when they can be carried out, the required number of resources, the resources with the capacity to do it. Each patient can give a preference for a resource. The resources have a type like nurse, doctor or assistant nurse and its have an experience related to each care. We have many constraints: we must respect the resource's and patient 's availability, each activity requires resources with a good number and compatible type, the routes begin and end at the depot and we must take into account the travel time between two activities. Our goal is to define the planning of each resource to provide cares at patient's home. We consider three criterion: to minimize the transporting time, to balance the dependency level of the activities carried out by each resource and to maximize the patients' preference. We have found several models for a home health care problem in the literature. We have identified different characteristics concerning resources or patients. In our problem, we have integrated the dependency level, the compatibility degree and three criteria described above. In the first part, we will propose a mathematical model. In the second part, we will explain approached methods and the combination between metaheuristics and simulation model. We conclude with fiew results.

2 Mathematical modeling

The proposed model is an extension of the m-TSPTW (multiple Traveling Salesman Problem with Time Windows) model. A set of cares for a patient is called an activity. An activity is carried out by one or two resources at patient's home. The several specific constraints are: synchronization of the activities, resources' number for each activity, compatibility degree between resources and activities. The type of the resources is modeling by compatibility degree. The degree is a real number between zero and one. When the compatibility between an activity and a resource is equal to zero, the resource can't carry out the activity. This degree may model another aspect, patients' preference or resources' experience. In the first case, a value close to zero means a low preference. In the second case, the closer to one the degree is, the more experienced the resource is to carry out the activity. We introduce two fictitious activities that represent the beginning and ending of the routes at the depot. We will describe the notations used in the model:

n_R	Number of resources
$R = \{1, ..., n_R\}$	Set of resources
n_A	Number of activities
$A = \{1, ..., n_A\} \cup \{0, n_{A+1}\}$	Set of planned activities and two fictitious activities
nb_i	Number of resources required to carry out activity $i \in A$
$\left[t_i^{min}, t_i^{max}\right]$	Time window of the duration of activity $i \in A$
τ_{ik}	Duration of activity $i \in A$ carried out by resource k
$[b_i, e_i]$	Time window to carry out activity $i \in A$
$\eta_i \in \{1, 2, 3, 4\}$	Level of dependence for the person receiving activity $i \in A$
D_{ij}	Travelling time between activity $i \in A$ and activity $j \in A$
$C_{jk} \in [0, 1]$	Compatibility degree between activity $j \in A$ and resource $k \in R$
M	High value

The duration and the dependency level of the fictitious activities equal to zero. The compatibility degree between each resource and fictitious activities equals to one. We introduce two variables:

$x_{ijk} = 1$ if resource $k \in R$ carries out activity $i \in \{0, ..., n_A\}$ before activity $j \in \{1, ..., , n_{A+1}\}$; 0 otherwise

z_i Starting date for activity $i \in A$

The objective function is a linear combination of the three criteria:

$$\sum_{i=0}^{n_A} \sum_{j=1}^{n_{A+1}} D_{ij} \sum_{k=1}^{n_R} x_{ijk} \ ; \ \max_{k \in R} \sum_{i=1}^{n_A} \sum_{j=1}^{n_{A+1}} x_{ijk}.\eta_i \ ; \ \sum_{j=1}^{n_A} \sum_{k=1}^{n_R} (1 - C_{jk}) \sum_{i=1}^{n_{A+1}} x_{ijk} \tag{1}$$

The constraints are:

$$\sum_{j=1}^{n_{A+1}} \sum_{k=1}^{n_R} x_{ijk} = nb_i \ \ \forall i \in \{1, ..., n_A\} \tag{2}$$

$$\sum_{i=0}^{n_A} \sum_{k=1}^{n_R} x_{ijk} = nb_j \ \ \forall j \in \{1, ..., n_A\} \tag{3}$$

$$\sum_{j=1}^{n_A} x_{Ojk} = 1 \ \ \forall k \in R \tag{4}$$

$$\sum_{i=1}^{n_A} x_{i(n_{A+1})k} = 1 \ \ \forall k \in R \tag{5}$$

$$\sum_{i=0}^{n_A} x_{ilk} = \sum_{j=1}^{n_{A+1}} x_{ljk} \ \ \forall l \in \{1, ..., n_A\}, \forall k \in R \tag{6}$$

$$z_j \leq z_i + \tau_{ik} + D_{ij} + (x_{ijk} - 1).M \ \ \forall i \in \{0, ..., n_A\}, \forall j \in \{1, ..., n_{A+1}\}, \forall k \in R \tag{7}$$

$$z_i \geq b_i \ \ \forall i \in \{1, ..., n_A\} \tag{8}$$

$$z_i \leq e_i - \tau_i \ \ \forall i \in \{1, ..., n_A\} \tag{9}$$

$$\sum_{i=0}^{n_A} x_{ijk} \leq C_{jk}.M \ \ \forall j \in \{1, ..., n_A\}, \forall k \in R \tag{10}$$

Constraints 2 make sure that the number of resources is correct for each activity. Constraints 3 assure that all resources will exit from patients' home. Constraints 4 and 5 force that the route begins and ends by fictitious activity. Constraints 6 ensure the conservation of the flow. Constraints 7 formulate the travel time between two activities. All activities are completed during the time window according to the constraints 8 and 9. Constraints 10 assure that the resources are compatible with the resources.

3 Proposed coupling

We propose a method based on a combination between metaheuristic (iterated local search) and a simulation model summarized in the figure 1. W. Abo-Hamad *et al.* (2010) propose a recent review of this kind of approach. The metaheuristic consists in ordering the activities carried out by each resources. The simulation model checks that the solution is feasible and computes the starting date of the activities. In this part, we are going to describe the proposed metaheuristics and the simulation model.

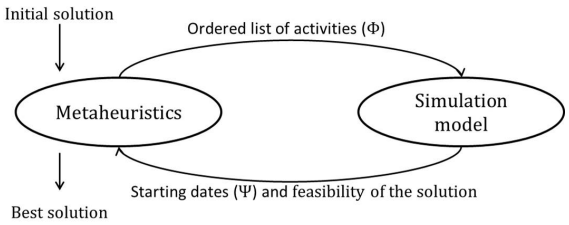


Fig. 1. Proposed method

3.1 Metaheuristics

We propose to use an iterated local search proposed by Lourenço H. *et al.* (2002). We will begin with the description of the solution encoding. The solution is a matrix $\phi = (\phi_{ik})_{A \times R}$. It gives the ordered list of the activities for each resource. We can represent the solution by a graph. The figure 2 on the left represents a feasible solution for ten activities

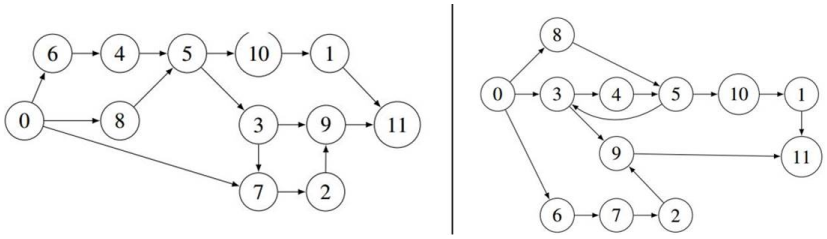


Fig. 2. Example for three resources and ten activities

and three resources. The activity 5, 7 and 9 need two resources. The two neighborhood systems used are based on swap and insertion moves. In the two cases, we ensure that the neighborhood gives a feasible solution according to the constraints of compatibility. The principle algorithms of the neighborhood system are given in algorithms 1 and 2.

The cardinal of the swap neighborhood is $\frac{a.(a-1)}{2}$ and the cardinal of the insertion neighborhood is $a.(a-1)$. To ensure that each neighborhood has the same probability to be chosen, we fix a probability equals to $\frac{1}{3}$ for swap neighborhood and $\frac{2}{3}$ for insertion neighborhood. Metaheuristics must be initialized by a feasible solution build with a greedy heuristics. The solutions are evaluated with a weighted sum of the criterion described in

Algorithm 1: Neighborhood Swap

Choose randomly and uniformly a resource i;
Choose randomly and uniformly an activity a carried out by the resource i;
Choose randomly and uniformly a resource j, compatible with activity a;
Choose randomly and uniformly an activity b compatible with the resource i carried out by the resource j;
Swap the activities a and b;

Algorithm 2: Neighborhood Insertion

Choose randomly and uniformly a resource i;
Choose randomly and uniformly an activity a carried out by the resource i;
Choose randomly and uniformly a resource j compatible with activity a;
Choose randomly and uniformly a position b in the ordered list of the resource j;
Insert the activity a in position b in the ordered list of the resource j;
Remove the inserted activity in the ordered list of the resource i;

section 1. We have encountered a difficulty relating that some activities are synchronized. When we modify the solution with the neighborhood, the graph representing the solution can contain a cycle. In this case, the solution is not feasible. The figure 2 on the right represents this case, activity 3 and 5 have caused a blocking position. We have developed a simulation model to compute the starting date. This model also permits to detect the cycle. In the following section, we will present this model.

3.2 Simulation model

We have developed a simulation model by using a programming language (Java). In the first step, we are allocating the first activity at each resource according to the list given by the metaheuristics. If an activity has enough resources to be carried out, we compute the starting date and the resources are released. In the second step, we are allocating the free resources at the next activity taken in the ordered list. We iterate the process while all activities have not a starting date. If no activity cannot be carried out, the required number of resource for the activities is not reached, the solution is not feasible. Otherwise we have computed the starting dates and the criterion.

4 Results

We obtain results with Cplex on a dataset randomly generated. It is composed by 30 activities and 4 resources. Seven activites need 2 resources and 23 activities need one resource. Two resources have the type nurse and two resources have the type doctors. We give priority to each criterion and we limit the computational time to 60 minutes. The small dataset permits to validate our method. We used our method in biggest datasets and a real dataset.

References

W. Abo-Hamad , A. Arisha, "Simulation-Optimisation Methods in Supply Chain Applications: A review", *Irish Journal of Management*, tome 1, pages 95-124

H. Lourenço, O. Martin, T. Stutzle, "Iterated local search", *Handbook of Metaheuristics*, Ed. F. Glover and G. Kochenberger, ISORMS 57, P. 321-353, Klumer

Using Integer Programming Methodologies for complex Single Machine Scheduling

Paul Göpfert, Stefan Bock

University of Wuppertal, Germany
pgoepfert, sbock@winfor.de

Keywords: Single machine scheduling, Branch & Price, Branch & Cut.

1 The scheduling problem

We consider the single machine scheduling problem defined in the commonly used three field notation as

$$1|seq-dep, inv, chains, d_j|\sum w_j C_j. \tag{1}$$

The problem is motivated by a real-world application at a supplier company in an automotive supply chain. The company operates a highly automated production line consisting of several stations for the assembly of car components. Due to mass customization, a considerable number of different component types is produced by this machine, in what follows, denoted as product versions. Changing from one product version to another induces setup activities on all stations that are affected by a change of the material build into the components. This leads to sequence dependent setup times ($seq-dep$) between jobs of different product versions. As for every job the needed raw materials have to be available at its start time in the schedule, inventory constraints (inv) occur. The actual release date of a job in the schedule is therefore determined by the maximum of the makespan for the scheduled predecessors and the earliest point in time that ensures material availability after processing all preceding jobs. All jobs of a single product version have to be produced in fixed order (earliest due date sequencing) establishing a special set of precedence relations ($chains$) among the jobs. As the supplier has to deliver the components on time, due dates (d_j), either given by the end manufacturers or derived from the subsequent production stages, have to be fulfilled. The objective of the considered scheduling problem is the finding of a feasible production schedule minimizing the total weighted completion time of all jobs.

2 Solution methodology

Exact solution approaches for NP-hard scheduling problems usually employ enumeration techniques like Branch and Bound or Dynamic Programming or a combination of both. However, due to the significant complexity, only instances with a moderate number of jobs can be solved to optimality in reasonable time.

In their literature review on scheduling with sequence dependent setup times, Zhu and Wilhelm (2006) propose the application of integer programming techniques as a promising future research path. While optimally solving an IP model with all constraints in one step is not promising, we present a decomposition of the problem. By defining a master problem, we construct by Branch and Price a schedule for the original problem out of the solutions of the arising subproblems. The solution process for the subproblems itself is based on Branch and Cut.

2.1 Decomposition

In order to derive a decomposition we exploit the predetermined precedences among the jobs and consider the following set cover:

Definition 1 *A **chain partition** $\mathcal{C}$ for the set of Jobs J is defined by the following properties:*

1. $\bigcup_{C \in \mathcal{C}} C = J$.
2. $C_1 \cap C_2 = \emptyset \, \forall \, C_1, C_2 \in \mathcal{C}, C_1 \neq C_2$.
3. *The jobs contained in every set $C \in \mathcal{C}$ (chain) are in a precedence chain.*

In our case a default chain partition is given by the different product versions of the jobs. A chain partition can also be dynamically determined by an analysis of the actual set of precedences. In the following, let $c(j)$ denote for every job j the unique chain C for which holds $j \in C$.

Definition 2 *A **chain schedule** for a chain $C \in \mathcal{C}$ is given by a schedule starting from the initial state of the machine. It includes at least all jobs of chain C and ends with the completion of the last job of chain C. A chain schedule is feasible according to all constraints and its objective value is given by the total weighted completion time of all jobs $j \in C$.*

Any chain schedule including all jobs yields a feasible schedule for the original scheduling problem and every feasible schedule for the original problem contains a set of $|\mathcal{C}|$ chain schedules. The schedules' total objective value is given by the sum of the objective values of the included chain schedules.

Instead of minimizing the total weighted completion time of all jobs, the resulting subproblems for every chain $C \in \mathcal{C}$ focus on the minimization of the total weighted completion time for a given subset of all jobs with fixed sequencing.

2.2 Column Generation

In the column generation process for the master problem we have to find a selection of chain schedules with a minimal total objective value while the combination of the selected schedules has to yield a feasible schedule for the single machine at hand. The main part of the Linear Programming Master (LPM) consists of two constraint classes. The convexity constraints (Dantzig and Wolfe 1960) enforce the selection of a single chain schedule for every chain $C \in \mathcal{C}$ (In what follows, S^c denotes the index set for the feasible schedules of chain C, and $\lambda_t^c \in [0,1]$ is the variable indicating the use of the t-th schedule of chain C):

$$\sum_{t \in S^c} \lambda_t^c = 1 \quad \forall \, C \in \mathcal{C} \tag{2}$$

The second class of constraints is based on a set partition. At every possible position $p = 1, \ldots, |J|$ in a feasible schedule exactly one chain may place an own job:

$$\sum_{C \in \mathcal{C}} \sum_{t \in S^c} \epsilon_{pt}^c \lambda_t^c = 1 \quad \forall \, p = 1, \ldots, |J| \tag{3}$$

with $\epsilon_{pt}^c = \begin{cases} 1 \text{ if } j \in C \text{ holds for the job } j \text{ at position } p \text{ in the } t\text{-th schedule of chain } C \\ 0 \text{ otherwise.} \end{cases}$

A further set of valid inequalities for this chain coordination problem is obtained by the following observation: If a chain schedule for chain C' includes a job $j \in C$, the direct predecessor i used by chain C to reach job j should also be used by chain C':

$$\sum_{t \in S^{c(j)}} \gamma_{tij}^{c(j)} \lambda_t^{c(j)} - \sum_{t \in S^{c'}} \gamma_{tij}^{c'} \lambda_t^{c'} \geq 0 \quad \forall\, (i,j) \in E, c' \neq c(j) \tag{4}$$

$$\text{with } \gamma_{tij}^{c} = \begin{cases} 1 \text{ if } i \text{ is a direct predecessor of } j \text{ in the } t\text{-th schedule of chain } C \\ 0 \text{ otherwise.} \end{cases}$$

Constraints of this class are added as cuts during the column generation process. Moreover, there are further classes of valid inequalities that can be applied to this problem.

As an illustrative example for the decomposition and the master problem we consider the job set $J = \{1,2,3,4\}$ with the precedence relations $1 \prec 2$ and $3 \prec 4$. The corresponding chain partition is $\mathcal{C} = \{C_1 = \{1,2\}, C_2 = \{3,4\}\}$. The schedule $(1,2,3,4)$ is feasible for the original problem and also a chain schedule for C_2. The chain schedule for C_1 is then $(1,2)$. The column coefficients of the constraints for some chain schedules are given in Table 1.

Table 1. Column coefficients

Constraint	$(1,2)$	$(1,2,3,4)$	$(1,3,2,4)$	$(1,3,2)$
(2) for C_1	1	0	0	1
(2) for C_2	0	1	1	0
(3) for $p=1$	1	0	0	1
(3) for $p=2$	1	0	1	0
(3) for $p=3$	0	1	0	1
(3) for $p=4$	0	1	1	0
(4) for $(1,3), c'=C_1$	0	0	1	-1

2.3 The Subproblems

The subproblems, each corresponding to a chain $C \in \mathcal{C}$, are used for the finding of chain schedules with negative reduced costs. Their objective is therefore the minimization of the total weighted completion time of the jobs $j \in C$ while the profits for certain decisions induced by the dual prices of the LPMs' constraints (2), (3) and (4) have to be maximized. The following three decision problems have to be adressed simultaneously:

1. Every job $j \in C$ has to be placed at a specific position in the schedule. We may get benefits from the duals of (3) by the choice of certain positions.
2. Job subsets $J_1, \ldots, J_{|C|} \subset J \backslash C, J_k \cap J_l = \emptyset \,\forall\, k \neq l$ have to be determined, that are used to fill the emerging gaps before and between the jobs of chain C.
3. A schedule has to be constructed by sequencing the jobs in every set $J_1, \ldots, J_{|C|}$. The costs of some sequencing decisions are affected by the duals of (4).

A lower bound for the subproblem is derived by the application of a LP-Model including these three aspects. By Branch and Cut promising regions for the finding of a column of negative reduced cost are detected. Besides some classes of valid inequalities that are newly derived for this model, cuts from literature developed for some included problems are applied. Every subproblem is also a selective TSP (cf. e. g. Bérubé *et. al.* (2009)) and

an ATSP with time windows (Ascheuer *et. al.* 2001). The separation method for the tournament constraints (Ascheuer *et. al.* 2000), that eliminate infeasible paths, is also used for the finding of upper bounds on the pricing problems.

As soon as the solution space is reduced to an appropriate size by branching as well as by variable fixation, an adaptation of the dynamic programming approach for the TSPTW by Dumas *et. al.* (1995) is employed to solve the remaining problem. This also ensures the feasibility of the chain schedules with regard to all constraints given in (1).

2.4 Branching Strategies

At every branching step the LP-solutions of the master and the subproblems are analyzed for ambiguities with regard to certain scheduling decisions. The branching decisions applied are based either on enforcing a precedence relation between two jobs (binary branching), including/excluding a single edge or splitting the position window of a single job. Every of these decisions may lead to further precedence relations and further window reductions, so in every tree node some preprocessing routines are employed.

2.5 Preprocessing

The preprocessing aims at the reduction of the time windows for every job and at the finding of further precedences. The upper bound on the maximal possible position of a job is decreased by testing the decision version of a multidimensional and precedence constrained knapsack problem for feasibility. In order to place a job on position p, $p-1$ jobs have to be included before, while the needed processing time may not cause a violation of the due date of the job. Also the implied material usage by the selection of predecessors may not exceed the given inventory levels at the due date of the job. The size of the time windows is decreased by considerations on the inventory levels and the necessary setup activities as well as the methods used in (Dumas *et. al.* 1995) and (Ascheuer *et. al.* 2001).

3 Conclusions

First tests on instances of type $1|chains, seq-dep, proc_j = 0|\sum w_j C_j$ derived from the berlin52 instance from TSPLIB (Reinelt 1991) indicate the strength of the set partitioning formulation (3) of the master problem as well as the need for further cuts as given in (4). An optimality proof in the root node is possible for some instances of this type. First computational results for the problem described in (1) will be presented at the conference.

References

Ascheuer N., M. Fischetti, M. Grötschel, "A Polyhedral Study of the Asymmetric Traveling Salesman Problem with Time Windows", *Networks*, Vol. 36, pp. 69-79.

Ascheuer N., M. Fischetti, M. Grötschel, "Solving the Asymmetric Travelling Salesman Problem with time windows by branch-and-cut", *Math. Program. Ser. A*, Vol. 90, pp. 475-506.

Bérubé J.-F., M. Gendreau, J.-Y. Potvin, "A Branch-And-Cut Algorithm for the Undirected Prize Collecting Traveling Salesman Problem", *Networks*, Vol. 54, pp. 56-67.

Dantzig G. B., P. Wolfe, 1960, "Decomposition Principle for Linear Programs", *Operations Research*, Vol. 8, pp. 101-111.

Dumas Y., J. Desrosiers, E. Galinas, 1995, "An optimal Algorithm for the traveling salesman problem with time windows", *Operations Research*, Vol. 43, pp. 367-371.

Reinelt G., 1991, "TSPLIB - A Traveling Salesman Problem Library", *ORSA Journal on Computing*, Vol. 3, pp. 376-384.

Zhu X., W. E. Wilhelm, 2006, "Scheduling and lot sizing with sequence-dependent setup: A literature review", *IIE Transactions*, Vol. 38, pp. 987-1007.

Scheduling orders with mold setups in an injection plant

Marek Gosławski[1], Joanna Józefowska[1], Marcin Kulus[1] and Jenny Nossack[2]

[1] Poznan University of Technology, Poland
{marek.goslawski, joanna.jozefowska}@put.poznan.pl,
marcin.kulus@doctorate.put.poznan.pl
[2] University of Siegen, Germany
jenny.nossack@uni-siegen.de

Keywords: scheduling, setup, heuristic.

1 Introduction

In this paper we present a scheduling problem observed in an injection molding plant. The problem involves mold and plastics changeovers which are very time consuming operations. Thus the natural objective is to minimize the total setup time. At the same time it is important to observe the order priorities. Additional constraints follow from the limited availability of staff responsible for fixing and adjusting the molds. We propose a 2-stage solution approach. At the first stage orders are assigned to machines and the sequence of orders on each machine is fixed. At the second stage we decide in which sequence molds are fixed and adjusted. We illustrate the approach with two numerical examples of real-life data. The schedules obtained by our approach are compared with schedules generated by a dedicated greedy heuristic and by an experienced dispatcher in the company. The results show that automated approach may lead to significant improvement of the machine utilization. The experiments will be continued. The ultimate goal of this research is to develop a dispatcher decision support system.

2 Problem Description

We consider a scheduling problem observed in an injection plant in a real life company. The plant consists of 35 injection molding machines. The machines differ in the pressure exerted on the mold and the architecture of the machine including, e.g., an additional hydraulic system controlling the core or the nozzle. In addition, some machines are able to produce two-color molders. Taking into account the machine characteristics, ten groupings of machines are distinguished.

Injection molds are very precise and quite complex elements involved in the production of molders. The mold warehouse stores currently about one thousand pieces. Each mold is assigned to a groupings of machines. Usually, a mold can be applied in a number of variants defined by a so-called "insert". Each variant corresponds to a different product (molder).

Plastics are the basic materials and represent the major cost associated with the production of molders. The transport of plastics is fully automated and is done through the so-called "plastics network". The main problem with plastics is related to the planning of changes of the colors. At each change of plastic it is necessary to clean the feeder and mill. Therefore, it is advisable to reduce the number of color changes.

Production orders are generated by an ERP system. Additionally, a dispatcher prioritizes orders according to their due dates.

Several different orders may be related to the same product, so batching is decided at the scheduling level. Each product is assigned to a given mold variant and in consequence to a given machine grouping where the mold may be fixed. There are 3 persons (called fixers)

at each shift responsible for the mold changeover. After fixing, the mold has to be adjusted and tested. These tasks are performed by operators. There are six operators in the plant, each one responsible for a different subset of machines, called a sector. Machines from the same grouping may belong to different sectors and machines in the same sector may belong to different groupings. Concluding, three alternative setups may occur: change of the mold and plastics, change of the mold, change of plastics. Obviously, the alternatives lead to different setup times. Minimization of the total setup time in a given planning horizon is one of the scheduling objectives. Another objective is to maximize the priorities of orders scheduled in the 48 hour timespan. Note that it is possible that not all orders are scheduled in the time frame of 48 hours. If an order stays unscheduled, the order's priority value is incremented and the order is reconsidered in the subsequent planning horizon.

Although we present an existing production plant, the solution procedure is general and does not depend on the specific number of machines, molds or staff. However, in order to verify the approach we have compared schedules obtained by the proposed solution approach with schedules developed by the dispatcher in the plant.

3 Solution Approach

3.1 2-stage heuristic

We propose a 2-stage heuristic for the scheduling problem that decomposes our problem into two subproblems, a machine scheduling and an operator scheduling problem. At the first stage of the algorithm, we solve the machine scheduling problem that decides on the assignment of the orders to the machines and on the sequence of the orders. We leave it to the second stage to assign the fixers and operators to the machines to set up and configure the molds that are required for the production.

3.2 Stage I - Machine scheduling

The first stage of the algorithm is subdivided into a preprocessing and a machine scheduling step which we further describe in detail.

Preprocessing step: Orders may be executed on a subset of machines that belong to the same machine grouping. In other words, orders cannot be carried out on machines that belong to different groupings. Thus, the machine scheduling problem naturally decomposes for the machine groupings and can be solved independently.

Machine scheduling step: For each machine group, we solve a routing problem that assigns orders to machines and decides on the sequence of the orders such that the sum of the priority values is maximized and the planning horizon is respected. This problem relates to the travelling salesman problem with profits (Feillet *et. al.* 2005) (TSP with profits) that generalizes the well-known TSP by assigning profits to the customers. The TSP with profits decides which customers are to be visited such that the total profit is maximized and the total travel cost is minimized. Our problem at hand extends the TSP with profits to multiple vehicles. We can consider each order as a customer, each machine as a vehicle and the priority of an order as the customer's profit. The travel cost between two orders is defined by the setup time plus the preprocessing time of the first order. The setup times are asymmetric and we are thus dealing with an asymmetric routing problem. Notice that the TSP with profits is a bicriteria objective programming problem where "the overall goal is the simultaneous optimization of the collected profit and the travel costs" (Feillet *et. al.* 2005). However, most of the existing literature address the TSP with profits as a single objective problem by either combining both of the objectives linearly or by treating one of the objectives as a constraint. We follow the latter and consider the maximization of

the total sum of priority values as our main objective and the setup times as a constraint. The TSP with profits has mainly been considered in the literature for the single vehicle case. Feillet *et. al.* (2005) give an extensive literature overview on the various exact and heuristic solution approaches. To our knowledge there is only one paper by Chao *et. al.* (1996) which considers the TSP with profits for multiple vehicles. The authors propose a neighboring heuristic based on a record-to-record improvement due to Dueck (1993). Since the machine scheduling problem can be decomposed for the different machine groupings, we obtain smaller problems that can be solved to optimality for some of the machine groupings. We formulate these problems as mixed-integer programming models and solve them by CPLEX.

3.3 Stage II - Operator scheduling

From the first stage of our solution approach, we obtain machine schedules for each of the machine groupings. In the second stage, we synchronize all pre-determined machine schedules by assigning the fixers and operators to the machines. In other words, we decide in which sequence the molds are fixed and adjusted on the machines. Since we have a single operator for each section, only one machine can be adjusted at a given time. Thus, if we neglect the mold setups by the fixers, each section has to solve an operator scheduling problem to decide in which sequence the molds are configured by respecting the pre-determined machine schedules of the first stage. Variants of this operator scheduling problem have been addressed, e.g., by Kravchenko and Werner (1997) and Hall *et. al.* (2000). We, however, have to deal with multiple operators that are synchronized by global operators (fixers). For a preliminary computational study, we simply shift the pre-determined machine schedules such that mold setups and mold configurations do not interfere. We have also started to develop a more sophisticated neighboring heuristic where we control a neighborhood search by a meta-heuristic.

4 Computational experiments

4.1 Greedy heuristic

We have developed the greedy heuristic since the computational times of the 2-stage approach are rather long.

It is assumed that the scheduling horizon is 48 hours. It is practically justified because new plan is delivered every day from the ERP system. In consequence, new orders with high priorities arrive that may influence the schedule.

The greedy heuristic finds a feasible solution by scheduling orders on machines one by one. The algorithm starts with a set of orders sequenced according to their non-increasing priority. An order is assigned to a machine available within the 48 hour horizon with the shortest setup time for that order. Ties are broken by choosing the machine that sooner completes the orders already scheduled. The processing of the order on the selected machine starts at the first moment when all the required resources are available.

4.2 Test instances

The test instances are prepared on the basis of actual production plans delivered from the ERP system. Two real production plans are considered. Usually the plan contains between 300 and 400 orders from which 60-80 are scheduled in the 48 hour horizon.

First experiments showed that computational time of the 2-stage heuristic may run into hours. Thus, it is proposed to select a subset of orders with the highest priority and run the heuristic on the subset. We considered subsets containing 100, 150 and 200 orders.

Table 1. Computational experiment results for 48h time period

Number of orders	Greedy heuristic			2-stage heuristic			Hand planned		
	N	P	S	N	P	S	N	P	S
PLAN 1									
100	29	6.00	2.17	54	4.56	1.98	79	4.11	2.60
150	41	6.07	1.93	69	4.39	1.62			
200	46	5.87	2.15	73	4.48	1.63			
PLAN 2									
100	25	5.88	2.08	55	4.76	1.80	60	3.67	2.82
150	36	5.81	1.94	66	4.27	1.77			
200	42	5.48	1.92	68	4.43	1.78			

4.3 Results

In Table 1 we present the results obtained for plan 1 and 2 respectively. For each number of orders selected for scheduling we present the values of objectives: number of setups (N), average setup time (S) and average priority (P) in a 48 hour horizon.

The computational time of the greedy heuristic is negligible, while for the 2-stage heuristic it runs into hours.

It may be observed that the 2-stage heuristic schedules bigger number of shorter orders while the greedy heuristics schedules long orders more likely. In consequence, average priority as well as average setup time is higher for the greedy heuristics. The schedules constructed by the dispatcher are clearly outperformed by both the heuristics. The experiments are being continued.

5 Conclusions

We have presented two heuristic approaches to solving a scheduling problem occurring in an existing injection mold plant. The problem is to select a subset of orders with possible highest priorities and schedule them in the 48 hour horizon so that the total setup time is minimized. Machine and order dependent setups are considered as well as additional resources (operators and fixers). The results of preliminary experiments show that both heuristics are able to find better solutions than solutions constructed manually by an experienced dispatcher. We plan to develop a decision support system using the proposed heuristics to assist the dispatcher.

References

Chao I.-M., B. L. Golden and E. A. Wasil, 1996, "The team orienteering problem", *European Journal of Operational Research*, Vol. 88, pp. 464-474.

Dueck G., 1993, "New optimization heuristics: The great deluge algorithm and the record-to-record travel", *Journal of Computational Physics*, Vol. 104, pp. 86-92.

Feillet D., P. Dejax and M. Gendreau, 2005, "Traveling salesman problems with profits", *Transportation Science*, Vol. 39, pp. 188-205.

Hall N. G., C. N. Potts and C. Sriskandarajah, 2000, "Parallel machine scheduling with a common server", *Discrete Applied Mathematics*, Vol. 102, pp. 223-243.

Kravchenko S. A., F. Werner, 1997, "Parallel machine scheduling with a single server", *Mathematical and Computer Modelling*, Vol. 26, pp. 1-11.

New Notes On PERT: The Effect of Different Activity Distribution on the Distribution of the Project Duration

Miklós Hajdu[1], Orsolya Bokor[2]

[1]Szent István University, Ybl Miklós Faculty of Architecture and Civil Engineering, Hungary
e-mail: hajdu.miklos@ybl.szie.hu

[2]Szent István University, Ybl Miklós Faculty of Architecture and Civil Engineering, Hungary
e-mail: bokor.orsolya@ybl.szie.hu

Keywords: activity calendar, activity distribution, PERT, sensitivity analysis.

1. Introduction

The original Program Evaluation and Review Technique (PERT) [1] is a probabilistic, activity-on-arrow network with one start and one finish event. In practice, the so-called three point estimation (optimistic (a), most likely (m), and pessimistic (b) duration) is used for describing the distribution of the activities, and to define mean and variance values of the probability density function (pdf).

The main goal of the PERT analysis is to create the distribution of the project duration. The original PERT calculation is based on the central limit theorem of probabilistic theory. According to this, project duration will follow a normal distribution within the domain of practical interests.

PERT has received a great deal of criticism since its "birth". The most important critiques and development of PERT are the following:

- The critiques of the three-point estimation [2], [3], [4], [5],
- Finding distributions that describe real life activity distributions better e.g. double truncated normal distribution [6], log-normal distribution [7], mixed beta and uniform distribution [8], triangular distribution [9], Parkinson distribution [10] etc.
- Original PERT assumes one critical path in the network. In case of more possible critical paths, the project duration determined by the calculations is strongly optimistic. Procedures that give better solutions/assumptions can be divided into three groups according to the classification by Adlakha [11] and Elmaghraby [12], from estimations [13] [14] [15], through Monte Carlo simulation [16] [17] [18], [19] to analytical solutions [20].
- The common characteristic of the above developments are that activity calendars were omitted; they were not considered. Recently Hajdu [21] has shown the dramatic effects of activity calendars on the distribution of the project duration. An unusual result of this research for project duration can be seen on Fig.1. Minor changes in the calendars have caused the differences between the cases, while he used the same one-chain network consisting of ten activities for all the cases.

The application of the beta distribution has been criticized by many researchers. During recent decades, researchers have suggested the usage of many different distributions. Some authors – among them Clark [2] - argue against the introduction of new probability distributions into PERT. According to him: "The author has no information concerning distributions of activity times; in particular, it is not suggested that the beta or any other distribution is appropriate." The main goal of this paper is to bolster the statement of Clark, that is to prove that the usage of different distributions does not result in considerably great differences in the distributions of the project duration, or at least these are smaller than the difference caused by a 10% inaccuracy in the estimation of the most likely values of the activities.

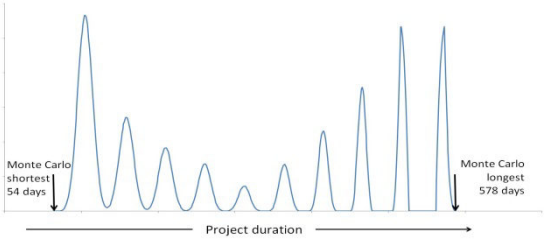


Figure 1. Density function for the project duration

2. The projects analyzed

Three very simple artificial networks (Fig.2, Fig. 3, Fig. 4) are used in this paper for demonstration purposes. The simplest among them is a one-chain PERT network with similar activity attributions. The other two will be used to demonstrate the differences in case of multiple critical paths and the situation of inter-twined critical paths.

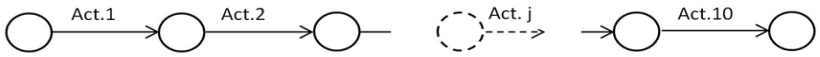


Figure 2. Sample #1, one-chain network

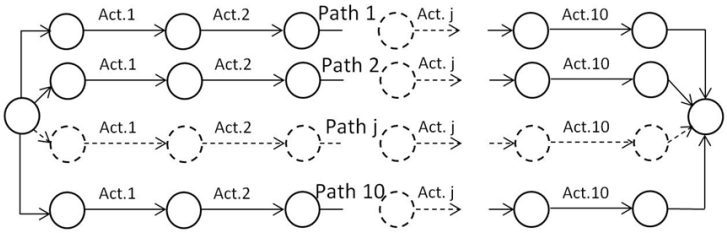


Figure 3. Sample #2

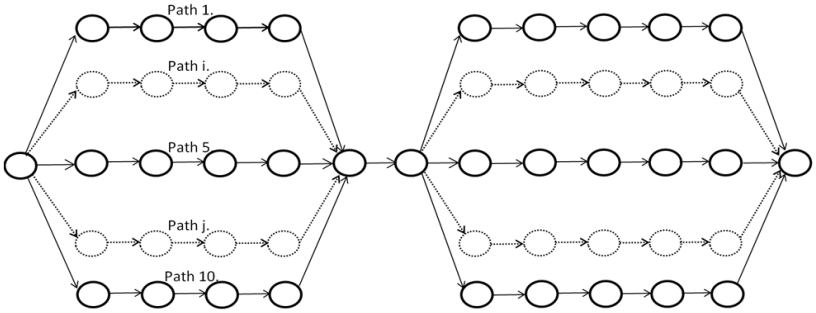


Figure 4. Sample #3, inter-twined critical paths

3. The effects of activity distributions on the project duration

To investigate the effects of different activity distributions on the distribution of the project duration different activity distributions were assigned to the activities (beta (case a), triangular (case d), uniform (case e), lognormal (case f)) in case of all the artificial networks. Then to illustrate the possible inaccuracy of the estimation of the durations, activity durations were decreased (case b) and increased (case c) by 10 percent compared to the original duration (case a). Distributions of the project duration were produced for all the networks in every case (a-f). Table 1 summarizes the durations and distributions applied.

Table 1. Durations and distributions for cases a) – f) of Sample #1 - #3

Cases	Optimistic (a)	Most likely (m)	Pessimistic (b)	Distribution Type	Legend
Case **a)**	60 days	100 days	150 days	PERT Beta	
Case **b)**	54 days	90 days	135 days	PERT Beta	
Case **c)**	66 days	110 days	165 days	PERT Beta	
Case **d)**	60 days	100 days	150 days	Triangular	
Case **e)**	60 days	100 days	150 days	Uniform	
Case **f)**	60 days	100 days	150 days	Lognormal	

Figure 5 shows the results for Sample #1. On the left-hand side of Figure 6 the distributions of the project duration of Sample #2 can be seen, while the results for Sample #3 are displayed on the right-hand side. It can be seen, that the different activity distributions cause smaller differences in the distribution of the project duration than the difference caused by a +/- 10% percent difference in the three-point estimation.

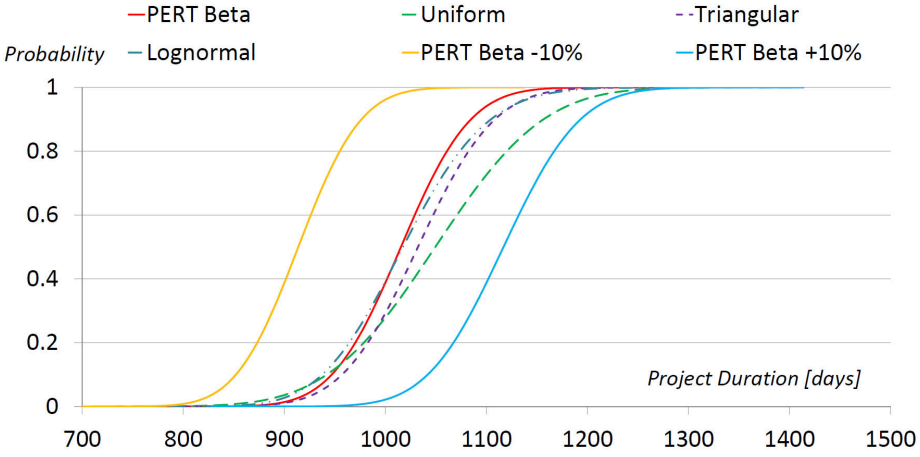


Figure 5. Distributions for Sample #1 cases a) –f)

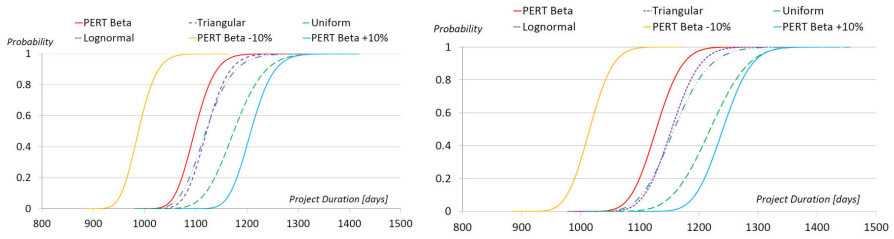


Figure 6. Distributions for Sample #2 and #3 cases a) –f)

4. Conclusions

The results clearly show that the accuracy of the estimation of the activity durations has greater impact on the distribution of the project duration than the effect of the application of various activity distributions. However, justification on real-life projects is still necessary.

References

[1] Malcolm, D.G.; Roseboom, J.H. ; Clark, C.E. ; Fazar W.: "*Application of a technique for a research and development program evaluation*" in: Operations Research, Volume 7, pp. 646–669, 1959

[2] Clark, C.E. "*The PERT model for the distribution of an activity time*" in: Operations Research, Volume 10 pp. 405–406, 1962

[3] Sasieni, M.W. "*A note on PERT times*" in: Management Science, Volume 32 pp. 405–406, 1986

[4] Keefer, D.L. & Bodily, S.E. "*Three-point approximations for continuous random variables*" in: Management Science, Volume 29 (5) pp. 595–609, 1983

[5] Farnum, N.R. & Stanton, L.W. "*Some results concerning the estimation of beta distribution parameters in PERT*" in: Journal of the Operations Research Society, Volume 38 pp. 287–290, 1987
[6] Kotiah, T.C.T. & Wallace, N.D." *Another look at the PERT asssumptions*" in: Management Science, Volume 20 (3-4) pp.44-49, 1973
[7] Mohan, S; Gopalakrishnan, M.; Balasubramanian, H.; Chandrashekar, A.: "*A lognormal approximation of activity duration in PERT using two time estimates*" Journal of the Operational Research Society, Volume 58, pp. 827–831 2007
[8] Hahn, E.D.: "*Mixture densities for project management activity times: A robust approach to PERT*" in: European Journal of Operational Research, Volume 188, pp. 450–459, 2008
[9] Johnson, D. : "*The triangular distribution as a proxy for the beta distribution in risk analysis*." in: Journal of the Royal Statistical Society: Series D (The Statistician), Volume 46, pp. 387–398. doi: 10.1111/1467-9884.00091 (1997)
[10] Trietsch, D., Mazmanyan, L., Gevorgyan, L., Baker, K.R "*Modeling activity times by the Parkinson distribution with a lognormal core: Theory and validation*" European Journal of Operations Research Volume 216 (2) , pp. 386-396, 2012
[11] Adlakha, V.G.:"*A classified bibliography of research on Stochastic PERT networks: 1966-1987*" , INFOR Volume 27 Issue 3 pp. 272-296, 1989
[12] Elmaghraby, S.E.:"*The estimation of some network parameters in PERT model of activity networks: Review and critique*" in: R. Slowinski; J. Weglarz (Eds.), Advances in Project Scheduling, Elsevier, Amsterdam, pp. 371-432, 1989
[13] Dodin, B.M.: „*Bounding the project completion time distribution in PERT networks*" Operations Research, Volume 33 pp. 862–881, 1985
[14] Kamburowski, J. : „*Bounding the distribution of project duration in PERT networks*", Operations Research Letters Volume 12, Issue 1, pp 17-22, July 1992.
[15] Yao, M.; Chu,W.: *„A new approximation algorithm for obtaining the probability distribution function for project completion time"* in: Computers and Mathematics with Applications, Volume: 54, pp. 282-295, 2007
[16] Dodin, B.M.: „*Approximating the distribution functions in stochastic networks*" Computers & Operations Research, Volume 12 Issue 3., pp. 251-264, 1985
[17] Van Slyke, R.M.: "*Monte Carlo methods and the PERT problem*" in: Operational Research, Volume 11, pp. 839-861, 1963
[18] Burt, J. M., Garman, M.B.: "*Conditional Monte Carlo: A simulation technique for stochastic network analysis*", Management Science Volume 18, Issue 3, 1971
[19] Sigal, C.E., Pritsker, A.B., Solberg, J.J.: „*The use of cut sets In Monte Carlo analysis of stochastic networks*", Mathematics and Computers in Simulation, Volume 21, pp 376-384, 1979
[20] Fischer, D.L.; Saisi, D.; Goldstein, W.M.: "*Stochastic PERT networks: OP diagrams critical path and project completion time*" in: Computers & Operations Research Volume 12 Issue 5, pp. 471-482, 1985
[21] Hajdu, M.: "*Effects of the application of activity calendars on the distribution of project duratiom in PERT networks*. " in: Automation in Construction, Volume 35, 2013. pp. 397-404

Decentralized Subcontractor Scheduling with Divisible Jobs

Behzad Hezarkhani[1], Wiesław Kubiak[2]

[1] Department of Industrial Engineering, Eindhoven Technical University, Netherlands
b.hezarkhani@tue.nl
[2] Faculty of Business Administration, Memorial University, NL, Canada
wkubiak@mun.ca

Keywords: Scheduling, Subcontracting, Coordination, Mechanism Design.

1 Introduction

We study the coordination of a decentralized system comprised of a subcontractor with a single machine and several manufacturing agents having divisible jobs. By reducing the finish-time, an agent achieves some monetary saving. The subcontracting problem considered in this paper was previously studied by Vairaktarakis and Aydinliyim (2007). The model borrows from the concept of divisible jobs introduced in the context of job-shops by Anderson (1981), and in the context of distributed computer systems scheduling by Bharadwaj (1996). Given the characterization of efficient (optimal) centralized allocations for our problem, we address the following question: how could efficient allocations be obtained in decentralized settings, i.e. without a centralized authority to schedule jobs optimally?

For a subcontractor who knows the true processing times of the agents, we introduce necessary and sufficient conditions for the existence of coordinating pricing schemes that result in a situation where each agent's optimal choice of subcontracting intervals coincides with its allocation in the efficient schedule. In general such pricing schemes do not necessarily exist (see Wellman et al. (2001)). We introduce a family of such coordinating pricing schemes and investigate the subcontractor's revenue under such pricing schemes and obtain a lower bound for it.

Next, we allow each job's processing time to be privately known by the agent owning that job. We obtain a closed-form formula for the payments based on the *pivotal mechanism* and prove that with this mechanism truth-telling is the *unique* optimal strategy for all agents which is exceptional since the uniqueness of truth-telling strategies is not necessarily implied by the pivotal mechanism. We prove that the total subcontractor's revenue equals the lower bound of the total payments with a coordinating pricing scheme.

2 The Problem and Efficient (Centralized) Solutions

Consider a set of agents $N = \{1, ..., n\}$. At time $t = 0$, every agent $i \in N$ has a divisible job with processing time p_i. Let $p = (p_1, ..., p_n)$ be the vector of processing times. A subcontractor is available that can process any portion of any job on its machine. By subcontracting, an agent's job can be processed in parallel on both its private machine and subcontractor's machine which results in reduction of completion time of its job. Let $T_i = [t_i, t_i + \bar{t}_i)$ be the uninterrupted time allocated to agent i, $i \in N$, on subcontractor's machine, where $t_i \geq 0$ and $\bar{t}_i > 0$ are the start-time and the duration of the interval respectively. Given allocation T_i, the saving from subcontracting to agent i is obtained in the following manner. Assume that initially, agent i uses its private machine only in the interval $[0, p_i]$. If $t_i < p_i$, then a portion of the remainder of job i to be done after t_i, i.e. $p_i - t_i$, can be transferred to the subcontractor. The most efficient way to do this is for

agent i to split the remainder equally between its private and subcontractor's machines, unless the duration of the interval $\bar{t}_i$ is too short. Therefore, the agent i can reduce the finish time of its job by an amount equal to

$$\upsilon_i(T_i) = \max\left\{\min\left\{(p_i - t_i)/2, \bar{t}_i\right\}, 0\right\}.$$

We assume that the saving in time is equivalent to saving in money with an equal rate for all agents. Therefore, $\upsilon_i(T_i)$ represents the monetary saving obtained by agent i under allocation T_i. An allocation $T = (T_1, ..., T_n)$ is *feasible* if and only if for any i, and j, the T_i and T_j do not overlap. Let $\mathfrak{T}$ denote the set of all feasible allocations T for the agents in N. We assume equal processing speeds for all machines. The *total saving* of a feasible allocation T is the sum of savings of all agents, that is $\upsilon(T) = \sum_{i \in N} \upsilon_i(T_i)$.

An *efficient allocation* $\mathcal{T} = (\mathcal{T}_1, ..., \mathcal{T}_n)$ is a feasible allocation that maximizes the total saving, that is $\mathcal{T} \in \arg\max_{T \in \mathfrak{T}} \upsilon(T)$. Let $\mathcal{T}_i = [\mathfrak{t}_i, \mathfrak{t}_i + \bar{\mathfrak{t}}_i)$ be the agent i's efficient allocation in an efficient allocation $\mathcal{T}$. Every efficient (optimal) allocation ensures that (a) each job finishes simultaneously on its own private and subcontractor's machines, i.e. $p_i - \bar{\mathfrak{t}}_i = \mathfrak{t}_i - \bar{\mathfrak{t}}_i$, and (b) the agents' allocations on subcontractor's machine are ordered based on non-decreasing sequence of their processing times. We assume hereafter until Section 4 that N is arranged in the non-decreasing order of processing times with ties broken according to the first-come-first-serve rule. An efficient allocation can be fully determined by a vector of efficient durations $\bar{\mathfrak{t}} = (\bar{\mathfrak{t}}_1, ..., \bar{\mathfrak{t}}_n)$. Although multiple efficient allocation exists as long as there are jobs with equal processing times, the allocations to jobs in the same positions are the same in different efficient allocations. Let $\mathcal{E}$ be the set of all efficient allocations.

3 Coordinating Pricing Schemes

A *pricing scheme* q, defined on $t \geq 0$, determines a price $q(t) \geq 0$ for acquiring time t on the subcontractor's machine by any agent. The subcontractor announces its pricing scheme, and subsequently agent i buys the time interval T_i in a first-come-first-serve manner. An agent makes its decision so as to maximize its utility, which for a given q, and assuming quasilinear utilities, equals $u_i^q(T_i) = \upsilon_i(T_i) - \pi^q(T_i)$, where $\pi^q(T_i) = \int_{t \in T_i} q(t)dt$ is the agent i's payment for T_i made to the subcontractor. A coordinating pricing scheme results in agents choosing the intervals that the efficient allocation prescribes.

Definition 1. *The pricing scheme q is* weakly *coordinating if for any $\mathcal{T} \in \mathcal{E}$, and each $T \notin \mathcal{E}$, $\pi^q(\mathcal{T}_i) - \pi^q(T_i) \leq \upsilon_i(\mathcal{T}_i) - \upsilon_i(T_i)$ for each $i \in N$. The pricing scheme q is* strongly *coordinating if the latter always holds strictly.*

A weakly coordinating pricing scheme may not necessarily result in the efficiency of the system since an agent upon arrival may chooses an interval which is not a part of any efficient allocation. However, the strong coordination requires all agents to exclusively choose efficient allocations.

A natural class of pricing schemes to consider for scheduling consists of those with non-increasing q where agents have to pay a higher price for buying earlier intervals. Let $\pi^q(T_i) = \pi^q(t_i, t_i + \bar{t}_i)$ for $T_i = [t_i, t_i + \bar{t}_i)$. Our main results in this section are as follows:

Theorem 1. *A non-increasing pricing scheme q is weakly coordinating if $q(0) < 1$ and for an efficient allocation $\mathcal{T}$:*

C1. $\pi^q(\mathfrak{t}_i - 2\epsilon, \mathfrak{t}_i) - \pi^q(\mathfrak{t}_{i+1} - \epsilon, \mathfrak{t}_{i+1}) \geq \epsilon$ *for* $i = 2, ..., n$, *and for* $0 < \epsilon \leq \bar{\mathfrak{t}}_{i-1}/2$;
C2. $\pi^q(\mathfrak{t}_i, \mathfrak{t}_i + 2\epsilon) - \pi^q(\mathfrak{t}_{i+1}, \mathfrak{t}_{i+1} + \epsilon) \leq \epsilon$ *for* $i = 1, ..., n-1$ *and for* $0 < \epsilon \leq \bar{\mathfrak{t}}_i/2$, *and for* $i = n$ *and* $0 < \epsilon \leq \bar{\mathfrak{t}}_n$.

We are now ready to give a family of non-increasing pricing schemes that are strongly coordinating.

Theorem 2. *Consider the coefficient set* $\kappa = \{\kappa_1, ..., \kappa_n\}$ *such that for* $i = 1, .., n-1$, $0 < \kappa_i < \kappa_{i+1}/2$ *and* $0 < \kappa_n \leq 2^{n-1}\delta/3$ *where* $0 < \delta \leq 3/2^{n+1}$. *The following pricing scheme is strongly coordinating:*

$$q_O(t) = \begin{cases} 1 - 2^{i-1}\delta + \kappa_i \left[1 - 2(t - \mathfrak{t}_i)/\bar{\mathfrak{t}}_i\right] & \text{if } \mathfrak{t}_i \leq t < \mathfrak{t}_i + \bar{\mathfrak{t}}_i, i \in N \\ 1 - 2^{n-1}\delta - \kappa_n & \text{if } t \geq \mathfrak{t}_n + \bar{\mathfrak{t}}_n \end{cases}$$

We now provide a necessary condition for any coordinating pricing scheme.

Theorem 3. *Any coordinating pricing scheme* q *meets the following condition: for every* $i = 2, ..., n$ *it holds that* $2q\left(t^-\right) - q\left(\mathfrak{t}_{i+1}^-\right) \geq 1$ *for* $0 < t < \mathfrak{t}_i$ *where* $t^- = \lim_{\epsilon \to 0^+}(t - \epsilon)$.

Let $\Pi^q = \sum_{i \in N} \pi^q(T_i)$ be the total payment from agents to the subcontractor under the pricing scheme q.

Theorem 4. *For any coordinating pricing scheme* q, *we have* $\Pi^q \geq \sum_{i \in N} \bar{\mathfrak{t}}_i \left(1 - 1/2^{n-i}\right)$. *For strongly coordinating pricing schemes this inequality holds strictly.*

A closed form formula for the subcontractor's revenue is $\Pi^{q_O} = \upsilon(\mathcal{T}) - \delta \sum_{i \in N} 2^{i-1}$. Thus, the coordinating pricing schemes from Theorem 2 can extract (almost) all of the total saving $\upsilon(\mathcal{T})$ from the agents by selecting δ small enough.

4 Private Processing Times

We allow each agent to report a processing time r_i belonging to its individual type space $P_i \subset \mathfrak{R}_+$. Let $P = P_1 \times ... \times P_n$ be the *type space* of all agents. We draw upon mechanism design theory Nisan (2007) and introduce a *payment scheme* that motivates agents to report true processing times knowing the subcontractor's intention of maximizing the total savings. A mechanism M is defined by a $(n+1)$-tuple $(f^M, \pi_1^M, ..., \pi_n^M)$ consisting of an allocation function $f^M : P \to \mathfrak{T}$, and a payment scheme $\pi^M : P \to \mathfrak{R}^n$. Given a vector of reported processing times r and the sequence in which the agents report them (to break ties in the allocation), the payment scheme of a mechanism determines the monetary amount $\pi_i^M(r)$ that $i \in N$ pays in return for receiving the allocation prescribed for i by f^M. The subcontractor is seeking to allocate the subcontracting intervals efficiently (optimally), thus we focus on mechanisms where $f^M = \mathcal{T}$. We call such mechanisms efficient. With quasilinear utilities, for $i \in N$ an efficient mechanism M would result in the utility $u_i^M(\mathcal{T}(r)) = \upsilon_i(\mathcal{T}(r)) - \pi_i^M(r)$.

Definition 2. *An efficient mechanism* M *is* incentive compatible *if for every reported processing time vector* $r = (r_1, ..., r_n)$ *and for every agent* $i \in N$, $u_i^M(\mathcal{T}(p_i, r^{-i})) \geq u_i^M(\mathcal{T}(r))$ *where* r^{-i} *is the vector* r *without its* i*th coordinate.*

Green and Laffont (1977) show that with quasilinear utilities and general valuation functions the Vickery-Clarke-Groves (VCG) mechanisms uniquely characterize the efficient and weakly incentive compatible mechanisms. The payment scheme for VCG mechanisms is defined as $\pi_i^{VCG}(r) = h_i(r^{-i}) - \sum_{j \in N, j \neq i} \upsilon_j(\mathcal{T}(r))$ where h_i is a function independent of agent i's announced processing time r_i. Among the class of VCG mechanisms, the *pivotal mechanism* with $h_i^{PM}(r^{-i}) = \sum_{j \neq i} \upsilon_j(\mathcal{T}(r^{-i}))$ guarantees that all payments from the agents to the mechanism are positive. Let $[i]$, for $i \in N$, be the position of i in $\mathcal{T}(r)$. A closed-form formula for payments obtained from the pivotal mechanism is given below.

Lemma 1. *For all* $i \in N$, *we have* $\pi_i^{PM}(r) = \bar{\mathfrak{t}}_i(r)\left(1 - \frac{1}{2^{n-[i]}}\right)$.

Our main result in this section is the following.

Theorem 5. *Let $r = (r_1, ..., r_n)$ be reported processing times and let p_i be true processing time of agent i. Then $u_i^{PM}(\mathcal{T}(p_i, r^{-i})) > u_i^{PM}(\mathcal{T}(r))$, provided that $p_i \neq r_i$ and i is not in the last position of $\mathcal{T}(r)$.*

The last theorem implies that truth telling is the *only* strategy of each agent which is not dominated by any other strategy.

Let us define $\Pi^{PM}(p) = \sum_{i \in N} \pi_i^{PM}(p)$ as the total payment from agents to the subcontractor in pivotal mechanism under true announcement of processing times. Instances can be made such that $\Pi^{PM}(p)/v(\mathcal{T}(p))$ be arbitrary close to zero. Based on the results obtain by Moulin (1986), in the decentralized subcontracting problem, among all efficient, incentive compatible, and individually rational mechanisms, pivotal mechanism generates the highest total payment. Thus we have the following.

Theorem 6. *Subcontractor's revenue with any efficient mechanism is always dominated by that with a coordinating pricing scheme.*

Therefore, *any* coordinating pricing scheme provides the subcontractor with the revenue higher than $\Pi^{PM}(p)$.

5 Concluding Remarks

An important extension of this model is the asymmetric valuation case where the agents obtain different amounts of savings for a unit time reduction in completion times of their jobs. Although the efficient centralized allocations in this case are more difficult to obtain, we show that it is impossible to devise a coordinating pricing scheme in general for this case. An open problem is the complexity of finding the efficient allocations in this case.

Acknowledgements

This research has been supported by the Natural Sciences and Engineering Research Council of Canada Grant OPG0105675.

References

Anderson E. J., 1981, "A New Continuous Model for Job-Shop Scheduling", *International Journal of System Science*, Vol. 12(12), pp 1469-1475.

Bharadwaj V., Ghose D., Mani V., and Robertazzi T. G., 1996, "Scheduling Divisible Loads in Parallel and Distributed Systems", *IEEE Computer Society Press.*

Green J. and Laffont J. J., 1977, "Characterization of satisfactory mechanisms for the revelation of preferences for public goods", *Econometrica*, Vol. 45(2), pp 427-438.

Moulin H., 1986, "Characterizations of the pivotal mechanism", *Journal of Public Economics*, Vol. 31(1), pp 53-78.

Nisan. N., 2007, "Introduction to mechanism design (for computer scientists)" in *Algorithmic Game Theory*, Cambridge University Press, pp 209-242.

Vairaktarakis G. L. and Aydinliyim T., 2007, "Centralization vs. competition in subcontracting operations", *Technical Memorandum Number 819*, Case Western Reserve University.

Wellman M. P., Walsh W. E., Wurman P. R., and MacKie-Mason J. K., 2001, "Auction protocols for decentralized scheduling", *Games and Economic Behavior*, Vol. 35(1-2), pp 271-303.

A heap of pieces model for the cyclic job shop problem

Laurent Houssin[1]

CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France
laurent.houssin@laas.fr

Keywords: cyclic scheduling, max-plus algebra.

1 Introduction

In classical scheduling, a set of tasks is executed once while the determined schedule optimizes objective functions such as the makespan or earliness-tardiness. In contrast, cyclic scheduling means performing a set of generic tasks infinitely often while minimizing the time between two occurrences of the same task. Cyclic scheduling has several applications, e.g. in robotic industry (see Kats and Levner [1997]), in manufacturing systems (see Pinedo [2005] and Hillion and Proth [1989]) or multiprocessor computing (see Hanen and Munier [1995b]). It has been studied from multiple perspectives, since there exist several possible representations of the problem such as graph theory, mixed integer linear programming, Petri nets or $(max, +)$ algebra. An overview about cyclic scheduling problems and the different approaches can be found in Hanen and Munier [1995a].

However, a less known tool is the *heaps of pieces* approach. A first paper considering this modelling for cyclic scheduling is Gaubert and Mairesse [1999]. The authors introduce a method to represent safe Petri nets as particular automata which compute the height of heaps of pieces. This approach enables to compute efficiently the throughput of a cyclic schedule. In this paper, we use the heaps of pieces approach for solving cyclic scheduling problems with resource constraints.

2 Cyclic scheduling problems

The basic cyclic scheduling problem (BCS) involves generic tasks and precedence constraints between tasks but no resource constraints are considered. In this problem, a set of n *generic operations* are processed in parallel by an unbounded number of machines and there is a set of q precedence constraints. We denote this set $\mathcal{A} = \{a_1, \ldots, a_q\}$ where a_l corresponds to a constraint represented by a triple (i, j, h). More precisely, the *uniform constraint* (i, j, h) means that

$$s_i(k) + p_i \leq s_j(k + h), \quad \forall k \geq 1$$

where s_i denotes the beginning of the operation i and p_i is the processing time of i. In this framework, the asymptotic cycle time $\alpha(S)$ is usually minimized (with S a feasible schedule). Equivalently we can aim at maximizing the throughput $r(S) = \frac{1}{\alpha(S)}$.

We can distinguish two categories of cyclic schedules: the 1-periodic schedules and the K-periodic schedules. The first category is characterized by a period α such that :

$$s_i(k + 1) = \alpha + s_i(k), \quad \forall i \in \mathcal{T}, \quad \forall k \geq 1,$$

where $\mathcal{T}$ is the set of tasks to perform. Whereas K-periodic schedules are also defined by period which corresponds to a fixed interval time between any K consecutive occurences of i :

$$s_i(k + K) = \alpha_K + s_i(k), \quad \forall i \in \mathcal{T}, \quad \forall k \geq 0.$$

For our concern, we are interested in the cyclic job shop problem. In this case, tasks are *a priori* mapped onto machines and the number of machines is smaller than the number of tasks to perform .More precisely, a cyclic job shop is defined by :

- a set $\mathcal{T}$ of elementary tasks,
- a set $\mathcal{R}$ of machines,
- for each task $t \in \mathcal{T}$, a processing time p_t and a machine $m_t \in \mathcal{R}$ on which the task has to be performed,
- a set $\mathcal{A}$ of uniform constraints (as defined above),
- a set of jobs $\mathcal{J}$ corresponding to a production sequence of elementary tasks. More precisely, a job J_1 defines a sequence $J_1 = t_{11} \dots t_{1k}$ to be executed in this order.

Previous studies of this problem have shown that K-periodic schedules are dominant (Hanen and Munier [1995a]) and the problem is NP-hard (Hanen [1994]) for throughput maximization.

One can find the description of two classical methods of performance evaluation of cyclic job shop in Hanen and Munier [1995a]. The first one is based on graph theory. In this framework, the throughput computation problem of a cyclic job shop schedule becomes the search of maximum circuit ratio in a graph. The second approach considers the job shop as a $(max, +)$-linear system. Then, the spectrum of the evolution matrix of the system gives the cycle time. In both approaches, the complexity of the evaluation of the cycle time is the same (polynomial).

3 The heap of pieces model

We first recall some algebraic tools concerning $(max, +)$ algebra. The $(max, +)$ semiring is the set $\mathbb{R} \cup \{-\infty\}$ endowed with the max operator, written $a \oplus b = max(a, b)$, and the usual sum written $a \otimes b = a + b$. The sum (resp. product) admits a neutral element denoted $\varepsilon = -\infty$ (resp. $e = 0$), it leads to $a \oplus \varepsilon = a$ and $a \otimes e = a$. For matrices, additions and products give $(A \oplus B)_{ij} = A_{ij} \oplus B_{ij}$ and $(A \otimes B)_{ij} = \bigoplus_{k=1}^{n} A_{ik} \otimes B_{kj}$.

An exhaustive presentation of $(max, +)$ algebra can be found in Baccelli et al. [1992]. We now present the heap model structure that was introduced in Gaubert and Mairesse [1999].

Definition 1 (Heap model). *A heap model is composed by :*

- $\mathcal{P}$ *a finite set of pieces,*
- $\mathcal{S}$ *a finite set of slots,*
- R *gives the subset of slots occupied by a piece,*
- $l : \mathcal{P} \times \mathcal{S} \to \mathbb{R} \cup \{-\infty\}$ *gives the lower contour of a piece,*
- $u : \mathcal{P} \times \mathcal{S} \to \mathbb{R} \cup \{-\infty\}$ *gives the upper contour of a piece.*

For each piece p (possibly non connected) of a heap model, we define the matrix $\mathcal{M}(p)$ of dimension $|\mathcal{S}| \times |\mathcal{S}|$ by

$$\mathcal{M}(p)_{sr} = \begin{cases} 0 & \text{if } s = r,\ r \notin R(p), \\ u(p)_r - l(p)_r & \text{if} r \in R(p),\ s \in R(p), \\ \varepsilon & \text{otherwise.} \end{cases}$$

Calculus in the heap model are based on $(max, +)$-algebra. More precisely, if a piece p_1 is piled up in an empty heap, the upper contour of the heap is given by $x(p_1) = I \otimes \mathcal{M}(p_1)$ where I is a $1 \times |\mathcal{S}|$ matrix defined by $I_j = e$, $\forall j \in \{1, \dots, |\mathcal{S}|\}$. In the same manner, the pile of two pieces p_1 and p_2 give the following upper contour of the heap : $x(p_1 p_2) = I \otimes \mathcal{M}(p_1) \otimes \mathcal{M}(p_2)$.

4 Application to cyclic job shop

In this section, we show how to model the cyclic job shop problem as a heap of pieces. We take the assumption that constraints $a_k \in \mathcal{A}$ are such that $h = 0$ (see §2). As shown in Gaubert and Mairesse [1999], the job shop problem admits a heap realization described under

$$\begin{aligned} \mathcal{P} &= \mathcal{T} \\ \mathcal{S} &= \mathcal{R} \cup \mathcal{J} \\ R(t) &= m(t) \cup \{\mathcal{J}_i | t \in \mathcal{J}_i\} \\ l(t,r) &= 0 \text{ if } m(t) = r, \qquad l(t,r) = \varepsilon \text{ otherwise} \\ u(t,r) &= p(t) \text{ if } m(t) = r, \quad l(t,r) = \varepsilon \text{ otherwise.} \end{aligned}$$

In this framework, each elementary task t is represented by a matrix $\mathcal{M}(t)$. Considering the objective of the cyclic job shop problem, the problem is to find the sequence of pieces piled up in the heap that maximize the throughput. The following theorem indicates how to compute the cycle time

Theorem 1. *The throughput of a job J_i for the cyclic sequence $p_1 \dots p_n$ in a heap model representing a job shop system is given by*

$$\lambda_{J_i} = |p_1 \dots p_n|_{J_i} \otimes (\rho(\mathcal{M}(p_1) \otimes \dots \otimes \mathcal{M}(p_n)))^{-1}. \tag{1}$$

where $|p_1 \dots p_n|_{J_i}$ denotes the number of jobs J_i completed under the sequence $p_1 \dots p_n$ and $\rho(X)$ is the unique eigenvalue of the irreducible matrix X (see [Baccelli et al., 1992, chap.2]).

Proof. Considering the heap, the cyclic sequence $p_1 \dots p_n$ corresponds to a cyclic piled-up of these pieces. Let $v = p_1 \otimes \dots \otimes p_n$. The matrix $\mathcal{M}(v)$ can be seen as the matrix representing an unique piece $p_1 \dots p_n$. The evolution of the heap is now given by the pile up of v. As seen in section 3, the upper contour of the heap is given by $x(v \dots v) = I \otimes \mathcal{M}(v) \otimes \dots \otimes \mathcal{M}(v)$. Since the job shop is connected (and we assume it is), the matrix $\mathcal{M}(v)$ is irreducible. After a certain number of iterations (that is called the transient time), a cyclic behaviour appears for an irreducible matrix $X : X^{k+c} = \rho(X)^c \otimes X^k$ (see Cohen et al. [1983]). This property hold for $\mathcal{M}(v)$ and after a certain number of iterations, the growth rate of the heap become $\rho(\mathcal{M}(v))$ and the throughput is given by (1).

Considering the heap of pieces model, the cyclic job shop problem is equivalent to find a bounded sequence of pieces with the maximum throughput. Compared to classical approaches, this model takes benefit from K-cyclic schedules. Indeed, graph based approach have to consider $K \times n$ number of nodes. Regarding mixed integer linear programming, the number of variables grows substantially. To deal with this situation, the heap of pieces is the best model since the number of slots is independant of K. Nonetheless the drawback of this method is that we can only consider a work in progress of one and it can not represent a "nested" sechedule since it can only represent patern that can be piled up (no intertwine). We propose a branch and bound procedure to solve this problem. The branching rule chose the piece to pile up in the heap and the evaluation fonction is the cycle time computation of the pattern in the heap. We extend this model to consider several work in process inside the pattern (but not nested pattern) through an extension of the number of slots.

Bibliography

F. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat. *Synchronization and Linearity*. Wiley, 1992.

J. Cochet-Terrasson, G. Cohen, S. Gaubert, M. Mc Gettrick, and J. P. Quadrat. Numerical computation of spectral elements in max-plus algebra. In *Proceedings of the IFAC Conference on System Structure and Control*, Nantes, 1998.

G. Cohen, D. Dubois, J. P. Quadrat, and M. Viot. Analyse du comportement périodique des systèmes de production par la théorie des dioïdes. Rapport de recherche 191, INRIA, Le Chesnay, France, 1983.

Ali Dasdan and Rajesh K. Gupta. Faster maximum and minimum mean cycle algorithms for system-performance analysis. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 17(10), 1998.

S. Gaubert and J. Mairesse. Modeling and analysis of timed petri nets using heap of pieces. *IEEE Trans. on Automatic Control*, 44(4), Apr. 1999.

C. Hanen. Study of a np-hard cyclic scheduling problem: The recurrent job-shop. *European Journal of Operational Research*, 72(1):82 – 101, 1994.

C. Hanen and A. Munier. *Scheduling Theory and Its Applications*, chapter Cyclic Scheduling on Parallel Processors: An Overview. Wiley, 1995a.

C. Hanen and A. Munier. A study of the cyclic scheduling problem on parallel processors. *Discrete Applied Mathematics*, 57:167–192, 1995b.

H. P. Hillion and J. M. Proth. Performance evaluation of job-shop systems using timed event graphs. *IEEE Transaction on Automatic Control*, 34(1):3–9, Jan. 1989.

R. M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Math*, 23, 1978.

V. Kats and E. Levner. A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling. *Operations Research Letters*, 21:171–179, 1997.

M. Pinedo. *Planning and Scheduling in Manufacturing and Services*. Springer, 2005.

Multi-Mode Resource-Constrained Project Scheduling with Sequence Dependent Transfer Times

Roubila Lilia Kadri and Fayez F. Boctor
Centre Interuniversitaire de Recherche sur le Réseau d'entreprises, la Logistique et le Transport
Faculté des sciences de l'administration, Université Laval, Québec, Canada.

Keywords: Project scheduling, Multi-mode activities, Resource transfer times.

1. Introduction

This paper deals with the *Multi-Mode Resource-Constrained Project Scheduling Problem with sequence dependent Transfer Times* (MMRCPSPTT). A sequence dependent transfer time is the time needed to transfer the required resource from the site of the preceding activity to the site of the succeeding one. Thus transfer time of a resource depends on the locations of the activities to execute as well as on the characteristics of the resource to transfer.

In the studied problem we assume that preemption is not allowed, the used resources are either renewable or non-renewable, each activity has a discrete number of execution modes and precedence relations are zero-lag finish-to-start relations. Also we assume that activity durations and transfer times are known and deterministic. The objective is to determine start time and execution mode for each activity of the project such that the project duration is minimized. Recently Krüger and Scholl (2009 and 2010) considered the single-mode resource-constrained project scheduling problem with transfer times (SMRCPSPTT). They proposed priority rule based solution procedures and presented numerical results to assess their performance.

To the best of our knowledge, the MMRCPSPTT has never been addressed before. Section 2 describes the considered MMRCPSPTT and section 3 presents a mathematical formulation of the problem. Section 4 presents a quick description of the heuristic we designed to solve large scale instances and section 5 describes the experiment conducted to assess the performance of the proposed heuristic.

2. Problem description

In the MMRCPSPSTT we have a set J of activities, a set P of finish-to-start precedence constraint, a set K of renewable resource and a set W of non-renewable resources. The limits on renewable resources are denoted Q_k; $k \in K$ and the availability over the entire project of the non-renewable resources is given by N_w; $w \in W$. Each activity i has a set of immediate predecessors $P_i \subseteq J$, a set of immediate and indirect predecessors, denoted π_i, a set of immediate successors $Z_i \subseteq J$ and a set of immediate and indirect successors, denoted σ_i.

In addition, every activity i has a set of possible execution modes M_i. A mode $m \in M_i$ is characterized by q_{imk}, the required number of units of the renewable resource k, n_{imw}, the required number of units of the non-renewable resource w, and the corresponding duration d_{im}. We also define I_i as the time window between the earliest start time and the latest start time of activity i. These times are calculated using the critical path method based on a feasible project duration T.

We assume that activities are to be performed in a number of different locations (sites). The transfer of a unit of renewable resource k from the execution location of activity i to the execution location of activity j requires a transfer time denoted Δ_{ijk}. The sum of transfer times of non-renewable resources to the project activities is constant and consequently is not considered in the problem.

Without loss of generality we assume that project activities include a dummy start activity, activity 1, and a dummy finish activity, activity *F*. These two activities have only one execution mode of duration zero, require no non-renewable resources and requires the entire available amount Q_k of the renewable resource *k*; $\forall k \in K$ (i.e., $q_{11k}=q_{F1k}=Q_k$). This is because the dummy start activity should provide renewable resources to the other activities while the dummy finish activity should collect them back. Transfer times from the dummy start activity and to the dummy finish activity are nil (i.e., $\Delta_{1ik}=\Delta_{iFk}=0; \forall i \in J, \forall k \in K$).

The objective in MMRCPSPTT is to assign a mode to each activity and to construct a schedule that minimizes the project duration while satisfying precedence, resource and transfer constraints.

3. Mathematical formulation

Decision variables

$f_{imjm'k}$ number of units of resource *k* directly transferred from activity *i* executed using mode *m*, to activity *j* executed using mode m'

$y_{imjm'k}$ a binary that takes the value 1 if and only if a unit or more of renewable resource *k* are transferred from activity *i* executed using mode *m* to activity *j* executed using mode m'

x_{imt} a binary that takes the value 1 if and only if activity *i* is executed using mode *m* and starts at the beginning of period *t*

Mathematical model

$$Minimize: \sum_{t\in I_F} t.x_{F1t} \quad (1)$$

Subject to:

$$\sum_{t\in I_i}\sum_{m\in M_i} x_{imt} = 1, \qquad \forall i \in J \quad (2)$$

$$\sum_{t\in I_j}\sum_{m'\in M_j} t x_{jm't} - \sum_{t\in I_i}\sum_{m\in M_i}(t+d_{im})x_{imt} \geq 0 \quad \forall (i,j) \in P \quad (3)$$

$$\sum_{m'\in M_j}\sum_{t\in I_j} t x_{jm't} - \sum_{m\in M_i}\sum_{t\in I_i}(t+d_{im})x_{imt} - T\sum_{m\in M_i}\sum_{m'\in M_j} y_{imjm'k} \geq -T + \Delta_{ijk}, \ \forall k \in K, i \in J-\{F\}, \ j \in J-\pi_i \quad (4)$$

$$y_{imjm'k} \leq \sum_{t\in I_i} x_{imt}, \quad k \in K, i \in J-\{F\}, \ j \in J-\pi_i, m \in M_i, \ m' \in M_j \quad (5)$$

$$y_{imjm'k} \leq \sum_{t\in I_j} x_{jm't}, \quad k \in K, i \in J-\{F\}, \ j \in J-\pi_i, m \in M_i, \ m' \in M_j \quad (6)$$

$$f_{imjm'k} \leq \min(q_{imk}, q_{jm'k})\, y_{imjm'k}, \quad k \in K, i \in J-\{F\}, j \in J-\pi_i, m \in M_i, m' \in M_j \quad (7)$$

$$\sum_{m\in M_i}\sum_{j\in J-\pi_i}\sum_{m'\in M_j} f_{imjm'k} = \sum_{m\in M_i}\sum_{t\in I_i} q_{imk}\, x_{imt}, k \in K, i \in J-\{F\} \quad (8)$$

$$\sum_{m'\in M_j}\sum_{i\in J-\sigma_i}\sum_{m\in M_i} f_{imjm'k} = \sum_{m'\in M_j}\sum_{t\in I_j} q_{jm'k}\, x_{jm't}, k \in K, i \in J-\{1\} \quad (9)$$

$$\sum_{i\in J}\sum_{m\in M_i}\sum_{t\in I_i} n_{imw} x_{imt} \leq N_w, \forall w \in W \quad (10)$$

$$f_{imjm'k} \geq 0, \forall m \in M_i, \forall m' \in M_j, \forall k \in K, \forall i \in J-\{F\}, \forall j \in J-\pi_i \quad (11)$$

$$y_{imjm'k} \in \{0,1\}, \forall m \in M_i, \forall m' \in M_j, \forall k \in K, \forall i \in J-\{F\}, \forall j \in J-\pi_i \quad (12)$$

$$x_{imt} \in \{0,1\}, \forall i \in J, \forall m \in M_i, \forall t \in I_i \quad (13)$$

The objective function (1) minimizes the project duration. Constraints (2) ensure that each activity is performed in only one mode and is started within its time window. Constraints (3) are the precedence constraints. Constraints (4) and (5) ensure that the transfer time of the renewable resource *k* is taken into consideration when transferred from activity *i* performed using mode *m* to activity *j* performed using mode *m'*. Constraints (6) imply that if $y_{imjm'k} = 0$ then $f_{imjm'k} = 0$. Constraints (7) and (8) represent the flow conservation of renewable resources. Constraints (9) ensure that the used quantities of non-renewable resources do not exceed the imposed limits.

4. The proposed heuristic

To solve the MMRCPSPTT we propose a three-phase heuristic. In the **first** phase, called the mode selection phase, we select an execution mode for each activity. We only require that the selected modes be feasible with respect to the non-renewable resource limits. This is done by solving an integer linear model. Two such models were tested. The first one (*M*1) is:

Find: $x_{im} \in \{0,1\}$; $\forall i \in J$, $\forall m \in M_i$ which

$$Minimize: \quad \sum_w (N_w - \sum_{i \in J} \sum_{m \in M_i} n_{imw} x_{im}) \tag{14}$$

$$Subject\ to: \quad \sum_{m \in M_i} x_{im} = 1 \tag{15}$$

$$\sum_{i \in J} \sum_{m \in M_i} n_{imw} x_{im} \leq N_w; \forall w \in W. \tag{16}$$

Where: x_{im} takes the value 1 if the mode *m* is assigned to the activity *i*.

The second model (*M*2) has the same constraints but the object function is replaced by:

$$Minimize: \quad \sum_{i \in J} \sum_{m \in M_i} \sum_{w \in W} d_{im} x_{im} \tag{17}$$

In the **second** phase we use the execution modes obtained in phase I and solve the resulting Single-Mode Resource Constrained Project Scheduling Problem with transfer times (SMRCPSPTT). This is done by using a combination of selection rules. The activity selection rule is the MinSlk (activity with minimum slack) rule and the resource transfer rule is the MinTT (minimum transfer time) rule. These two rules are used within a parallel scheduling scheme. Kruger et al (2009) tested several combinations of priority rules and indicated that this is the most efficient way to solve the SMRCPSPTT.

The **third** phase is an improvement phase. Two improvement procedures were tested. The first procedure (*P*1) is a simple hill climbing approach where we search the unit neighborhood where we modify the execution mode of only one activity, provided that this modification does not lead to requiring an amount of any non-renewable resource that exceeds its availability. Once a local optimal is reached, we use again the hill climbing procedure to search a pairwise neighborhood where we modify the execution mode of two activities simultaneously. The procedure stops when we reach a local optimum of this second neighborhood. The second improvement procedure (*P*2) is a multi-neighborhood improvement approached (see Boctor, 1993). The same two neighborhoods presented above are used within this approach. The multi-neighborhood approach is an iterative approach where each iteration consists of searching the first neighborhood until no more improvement can be achieved. Next we search the second neighborhood until a better solution is found. Then we go back to search the first neighborhood of the obtained solution. The procedure stops if no improvement can be achieved during a complete iteration.

5. Computational results

Test instances and optimal solutions

We conducted a computational study to assess the performance of the proposed heuristic. To do this study we extended three ProGen instance sets: the sets J10, J20 and J30 of the MMRCPSP (Kolisch et al., 1996). Transfer times are randomly sampled from a uniform distribution between 0 and 10 periods. In all, we used 536 instances with 10 non dummy activities, 554 instances with 20 non dummy activities and 552 instances with 30 non dummy activities. All coding was done in Matlab 2012 environment and all experiments were run on a computer with i7 core processor, 2 GHz and 6 GB of internal memory. GUROBI 5.5 was used to solve the proposed mathematical formulation with a run time limit of 3600 CPU-seconds per instance. Table 1 shows that the optimal solution was found for only 40% of the test-instances and that problems with RF=1(Resource Factor) are more difficult to solve than those with RF=0.5. Notice that

although we have not obtained the optimal solution for the 60 % other instances, we got at least a feasible one for each of them within the imposed run time limit.

Table 1: Number of times the optimal solution was found and average percentage deviation

Set	RF	RS	Total number of instances	Number of instances where the optimum is found	Percentage of instances where the optimum is found	Average percentage deviation from the optimum (calculated only over instances where the optimum is found)				Average percentage deviation from the best found solution (calculated over all instances)			
						*M*1*P*1	*M*1*P*2	*M*2*P*1	*M*2*P*2	*M*1*P*1	*M*1*P*2	*M*2*P*1	*M*2*P*2
All J10, J20 & J30	0,5	0.25	190	48	25,26%	7,99	6,46	5,40	6,33	5,56	3,84	4,76	3,69
		0.5	210	113	53,81%	9,67	7,77	9,52	8,64	8,24	6,33	6,85	6,20
		0.75	210	190	90,48%	8,21	6,84	6,88	6,39	8,31	6,98	6,82	6,29
		1	190	189	99,47%	5,78	4,74	4,80	4,48	5,82	4,78	4,85	4,53
	1	0.25	209	0	0,00%	-	-	-	-	2,45	1,78	2,15	1,44
		0.5	210	7	3,33%	7,81	6,39	6,86	6,86	3,31	2,38	2,14	1,57
		0.75	212	31	14,62%	11,80	11,58	9,94	9,71	4,81	4,08	3,31	2,86
		1	211	73	34,60%	9,96	9,79	8,60	8,17	5,30	5,19	4,48	4,18
	All RF-RS		1642	651	39,65%	8,10	6,92	6,96	6,58	5,47	4,43	4,41	3,84

Results obtained by four versions of the proposed heuristic

As shown in Table 1, four versions of the heuristic were tested. These versions are the result of combining *M*1 or *M*2 in the first phase with *P*1 or *P*2 in the third one. The table gives the percentage deviation from the optimum (when the optimum is obtained) and the percentage deviation from the best found solution (for all instances) for each version. Table 2 gives the number of times the optimal and the best solutions are found. Many other results are in Kadri et Boctor (2014).

Table 2: Number of times the optimal solution and the best solution are found by each version of the heuristic

Set	RF	RS	Total number of instances	Number of instances where the optimum is found	Number of times the optimum is found				Number of times the best solution is found			
					*M*1*P*1	*M*1*P*2	*M*2*P*1	*M*2*P*2	*M*1*P*1	*M*1*P*2	*M*2*P*1	*M*2*P*2
J10, J20 & J30	0,5	0.25-0.5	400	161	39	49	46	49	189	263	217	260
		0.75-1.0	400	379	154	172	154	166	255	299	281	305
	1	0.25-0.5	419	7	1	1	1	1	214	263	256	297
		0.75-1.0	423	104	18	17	16	16	252	270	287	306
All			1642	651	212	239	217	232	910	1095	1041	1168

Acknowledgement

This research was partially supported by the Canadian Natural Sciences and Engineering Research Council Grant OPG0036509. This support is gratefully acknowledged.

Reference

Boctor F.F. (1993). Discrete optimization and multi-neighborhood local improvement heuristics. Document de travail 93-35, FSA, Université Laval.

Kolisch R. and A. Sprecher (1996). PSPLIB – A project scheduling problem library. *European Journal of Operational Research*, 96: 205–216.

Krüger, D. and A.Scholl (2009). A heuristic solution framework for the resource constrained (multi-)project scheduling problem with sequence-dependent transfer times. *European Journal of Operational Research*, 197: 492-508.

Krüger, D. and A.Scholl (2010). Managing and modelling general resource transfers in (multi-)project scheduling. *OR SPECTRUM*, 32: 369-394.

Kadri R L and Boctor F F (2014) Multi-mode, resource-constrained project scheduling with sequence-dependent transfer times, document de travail, FSA Université Laval.

Enhanced Precedence Theorems for $1||\sum T_j$

J.J. Kanet[1], R. Andris[2], C. Gahm[2], and Axel Tuma[2]

[1]Department of MIS OM & DS, University of Dayton, Dayton, OH, USA
kanet@udayton.edu

[2]Department of Production & SCM, University of Augsburg, Bavaria, Germany
{ralph.andris, christian.gahm, axel.tuma}@wiwi.uni-augsburg.de

Keywords: scheduling, tardiness, one-machine, precedence.

1. Introduction

Scheduling jobs onto a single machine to minimize the total tardiness ($1||\sum T_j$) is a well-known problem of practical importance in virtually all industrial operations. Moreover, it is known to be strongly NP-hard (Lawler, 1977). As the problem is NP-hard and therefore requires enumeration procedures like dynamic programming or branch & bound for its solution, precedence (dominance) theorems can help in determining pair-wise relations between jobs, e.g., that a job *j* precedes a job *k* in an optimum solution. We shall use the notation "$j \prec k$" to mean "j precedes k in some optimum sequence." Each time we discover such a relation, the search space is reduced by as much as one half. Precedence relations are transitive, e.g., $i \prec j$ and $j \prec k \Rightarrow i \prec k$, so the effect of their discovery can accumulate. Such theorems were first introduced by Emmons (1969) and later extended by Rinnooy Kan, Lageweg and Lenstra (1975) and Kanet (2007). Schedules adhering to such precedences are called *dominant* schedules.

The proof tactics employed by Emmons to test if job $j \prec k$ is to start with an assumed optimal sequence S where k precedes j. Now a schedule S' is constructed in which j precedes k with minimal disturbance to the other jobs in the schedule. The two tactics are:

- Tactic 1: Swapping positions of jobs k and j
- Tactic 2: Inserting job k immediately after job j

If it can be shown that the maneuver provides tardiness for S' not higher than that for S, then it is possible to say that schedules in which k precedes j are not uniquely optimum, thus $j \prec k$. The new theorems follow exactly those tactics but consider one or more additional jobs in the maneuver. This report describes the current progress of this research effort. In the section to follow, we first introduce notation (2.1) and then describe in more detail two of the seven new theorems (2.2). Section 2.3 summarizes the seven new theorems. Section 3 provides initial findings of the computational results illustrating the marginal advantage (beyond Emmons's theorems) that the two theorems provide. Section 4 projects the remaining research underway.

2. The New Theorems

2.1. Notation

We consider the sequencing of a set N of n jobs available for processing at time t = 0 by a continuously available machine. Each job *i* is characterized by a processing time $p_i > 0$ and a due date $d_i > 0$. The goal is to determine the scheduled completion time for job *i*, $C_i \; \forall$ i, such that total tardiness is minimum. We let B_i represent the set of jobs already known to precede i in some optimum sequence; A_i represents the set of jobs known to follow i in some optimum sequence. We use the notations $B_i' = N\text{-}B_i$, and $A_i' = N\text{-}A_i$. We use the notation P(X) to mean the total processing time of jobs in

set X. In determining if job $j \prec k$, the new theorems make use of $P(B_k)$, $P(A_j)$, as well as the information about one or more other jobs $w \in B_k$ or $z \in A_j$. In proving the new theorems we use the concept of the tardiness improvement, $TI_i(d_i)$, tardiness decrement, $TD_i(d_i)$, for a job i that results from a maneuver (be it swapping or inserting after), where TI, TD are functions of the job's due date. Finally, the notation UB(.), LB(.) denotes upper bound, lower bound, respectively.

2.2. Using the swap and insert-after tactic with job $w \in B_k$

Theorem SW1: Given jobs j, k, $w \in B_k$ and $p_j \leq p_w$,

if $d_j \leq max\{d_w, P(B_w)+p_w\}+min\{max\{0, P(B_k)+p_k-d_k\}, p_w-p_j\}$ *then* $j \prec k$.

The proof involves beginning with a dominate sequence in which w and k precede j, then swapping w with j and observing if $UB(TD_w) \leq LB(TI_j)+LB(TI_k)$. Figure 1 illustrates.

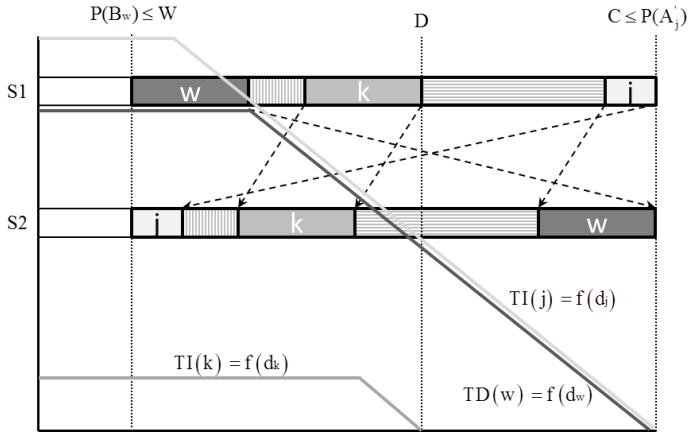


Figure 1. New swap theorem considering three jobs

Theorem IA1: Given jobs j, k, $w \in B_k$,

if $max\{d_w, P(B_w)+p_w\} \geq max\{d_j, P(A_j')-p_w\}-min\{max\{0, P(B_k)+p_k-d_k\}, p_w\}$ *then* $j \prec k$.

As above, the proof involves beginning with a dominate sequence in which w and k precede j, then inserting w after j and observing if $UB(TD_w) \leq LB(TI_j)+LB(TI_k)$. Figure 2 illustrates.

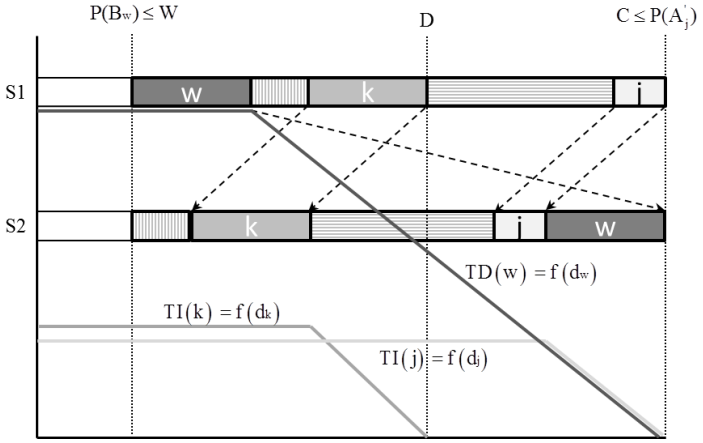


Figure 2. New insert-after-theorem considering three jobs

2.3. The set of seven possible theorems

We currently envision seven new theorems which are under investigation. Three of the possible new theorems follow the swap tactic while four new theorems follow the insert-after tactic. Figure 3 summarizes those concepts. SW1, SW3, IA1, and IA 3 consider the case when one or more jobs $w \in B_k$. Theorems SW2, IA2, and IA4 have one or more jobs $z \in A_j$. There is no theorem SW4 because its condition would be too restrictive to discover further precedences ($p_j + \sum z_{wi} <= p_k$).

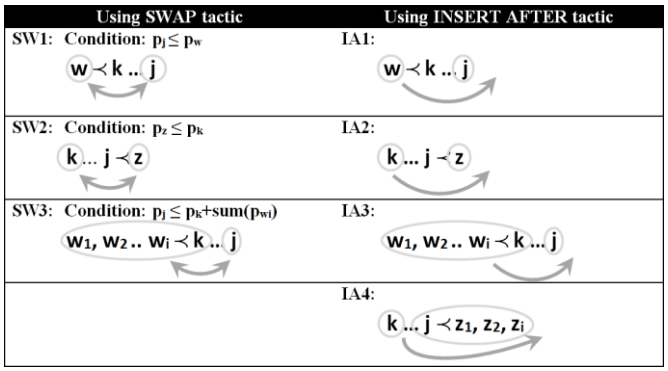


Figure 3. Current and possible new theorems

3. Computational Results

To gain an insight into the benefit of the new theorems a computational study was conducted to observe the number of precedence discoveries by the original theorems of Emmons (1969) and the additional discoveries of the new theorems over a specific set of known problem instances. For this analysis, we use a set of twelve 20-job problem instances from Baker and Trietsch (2009) and 40-job and 50-job problems from Beasley (1990) with 125 instances each. The study was performed in several steps. At first, we calculated the optimum objective function value for each instance with the CPLEX solver to create a validation reference. We then started to activate the theorems one after the other and let the program calculate until no further precedences were discovered. In this state, we captured the number of "hits" produced by the activated theorems.

Table 1. Emmons 1,2,3 & new theorems applied on twelve Baker-/Trietsch' 20-job instances

Experiment(s)	E123	E123+SW1			E123+IA1			E123+SW1+IA1		
	HITS	TOTAL HITS	INCR. [%]	+ HITS (SOLOS)	TOTAL HITS	INCR. [%]	+ HITS (SOLOS)	TOTAL HITS	INCR. [%]	+ HITS (SOLOS)
Baker 1	148	149	0,67	1 (0)	148	0,00	0 (0)	149	0,67	1 (0)
Baker 2	195	195	0,00	0 (0)	195	0,00	0 (0)	195	0,00	0 (0)
Baker 3	234	234	0,00	0 (0)	235	0,43	1 (0)	235	0,43	1 (0)
Baker 4	182	182	0,00	0 (0)	183	0,55	1 (0)	183	0,55	1 (0)
Baker 5	128	128	0,00	0 (0)	128	0,00	0 (0)	128	0,00	0 (0)
Baker 6	179	182	1,65	3 (0)	181	1,10	2 (0)	184	2,72	5 (0)
Baker 7	122	122	0,00	0 (0)	123	0,81	1 (0)	123	0,81	1 (0)
Baker 8	204	204	0,00	0 (0)	205	0,49	1 (0)	205	0,49	1 (0)
Baker 9	189	194	2,58	5 (0)	190	0,53	1 (0)	195	3,08	6 (0)
Baker 10	216	216	0,00	0 (0)	216	0,00	0 (0)	216	0,00	0 (0)
Baker 11	115	115	0,00	0 (0)	115	0,00	0 (0)	115	0,00	0 (0)
Baker 12	176	176	0,00	0 (0)	177	0,56	1 (0)	177	0,56	1 (0)

We compared the number of total theorem hits when applying the new theorems to the number of hits we received with Emmons' theorems solely. The result is provided in the "Increase"-column. Furthermore, the number of hits provided by SW1, IA1, or both is provided in the last column of each test. The number in brackets specifies the quantity of solo hits, which means that only SW1 or only IA1 discovered a precedence relation.

The results for the small Baker instances (cf. Table 1) reveal first hits of the new theorems, though providing no new information (no solo hits). This can be explained due to the fact that the new theorems are more likely to hit if there are many already known precedence relations ("Given jobs (…), $w \in B_k$").

In contrast to this, table 2 depicts results for the bigger Beasley instances showing a lot of solo hits, especially by SW1. As there are 125 instances in each of the Beasley sets, the results are condensed. They provide the average number of hits throughout all the instances. Furthermore, the average and maximum increase of discovered precedences is stated. The last column in each test shows the amount of additional (solo) hits provided by the new theorem(s).

Table 2. Emmons 1,2,3 and new theorems applied on 125 Beasley's 40- and 50-job instances

Experiment(s)	E123	E123+SW1				E123+IA1				E123+SW1+IA1			
	Ø HITS	Ø TOTAL HITS	Ø INCR. [%]	MAX INCR. [%]	+ HITS (SOLOS)	Ø TOTAL HITS	Ø INCR. [%]	MAX INCR. [%]	+ HITS (SOLOS)	Ø TOTAL HITS	Ø INCR. [%]	MAX INCR. [%]	+ HITS (SOLOS)
Beasley's 125 40-job instances	575	616	**10,11**	**31,70**	**5781 (596)**	581	**1,05**	**5,41**	**714 (8)**	620	**11,05**	**32,45**	**6415 (599)**
Beasley's 125 50-job instances	862	919	**9,49**	**30,83**	**7802 (672)**	870	**1,01**	**5,17**	**958 (2)**	925	**10,42**	**31,39**	**8692 (674)**

4. Outlook

The results and first insights into the benefit of all the new theorems necessitate further research. At first, the full set of new theorems should be specified in detail. In regard of computational studies we expect a performance boost for the runtime of the solver which has not been analysed in detail yet. Those effects will arise especially when solving larger instances. After proving a benefit from those new theorems virtually all existing approaches for solving $1||\sum T_j$ and its variants might be well served by application of those.

References

Arkin, E.M., and Roundy, R.O., 1991, "Weighted-Tardiness Scheduling on Parallel Machines with Proportional Weights", Operations Research, Vol. 39, pp. 64-81

Baker, K.R., and Trietsch, D., 2009, "Principles of Sequencing and Scheduling", Wiley

Beasley, J.E., 1990, "OR-Library: distributing test problems by electronic mail", Journal of the Operational Research Society, pp. 1069-1072.

Emmons, H., 1969, "One-machine sequencing to minimize certain functions of job tardiness", Operations Research, Vol. 17, pp. 701-715.

Kanet, J.J., 2007, "New Precedence Theorems for One-Machine Weighted Tardiness", Mathematics of Operations Research, Vol. 32, pp. 579-588.

Kanet, J.J., and Birkemeier, C., 2013, „Weighted Tardiness for the single machine scheduling problem: An examination of precedence theorem productivity", Computers & Operations Research, Vol. 40, pp. 91-97.

Lawler, E.L., 1977, "A 'Pseudopolynomial' Algorithm for Sequencing Jobs to Minimize Total Tardiness", Annals of Discrete Mathematics, Vol. 01, pp. 331-342.

Rinnooy Kan, A.H.G., Lageweg, B.J., and Lenstra J.K., 1975, "Minimizing total costs in one-machine scheduling", Operations Research, Vol. 23, pp. 908-927.

Szwarc, W., and Liu, J.J., 1993, "Weighted Tardiness Single Machine Scheduling with Proportional Weights", Management Science, Vol. 39, pp. 626-632.

Vepsalainen, A.P.J., and Morton, T.E., 1987, "Priority rules for job shops with weighted tardiness costs", Management Science, Vol. 33(8), pp. 1035-1047.

A Polynomial Time Algorithm for an Integrated Scheduling and Location Problem

Corinna Kaufmann

Technical University of Kaiserslautern, Germany
kaufmann@mathematik.uni-kl.de

Keywords: Scheduling, Location Planning, ScheLoc

1 Introduction

Scheduling and Location Problems are two important areas of operations research. While each of them individually has a large number of applications (e.g., manufacturing, logistics) for many problems (e.g., supply chain management) both location and scheduling decisions have to be made. To overcome weaknesses of a sequential approach, Hennes and Hamacher (2007) introduced an integrated Scheduling-Location (ScheLoc) Problem. The ScheLoc Problem is the problem of simultaneously finding machine locations and a schedule, where release dates of jobs are given by the distance from job locations to the location of the machine on which they are processed such that some scheduling objective function is optimized. A large number of applications of this problem have been summarized in Kalsch (2009). One of them can be found in the service sector. Some services (e.g., mammography screenings) are not offered in one fixed location, but the service location can change on a regular basis. For certain periods demand of customers and their locations are known or can be estimated. Considering this demand already when locating the service station can significantly reduce service times.
The Single Machine Network (SMN) ScheLoc Problem was further investigated by Hennes (2005) and polynomial solvability of special cases was shown. Polynomial time algorithms for the Single Machine Planar (SMP) Makespan ScheLoc Problem were given by Elvikis *et. al.* (2008) and Kalsch and Drezner (2010). Kalsch (2009) introduced the Universal ScheLoc Problem similar to the Ordered Weber Problems in location theory and applied it to SMP ScheLoc Problems. Other Universal Combinatorial Problems were studied in Turner (2013).
Related integrated scheduling problems like the Integrated Production and Outbound Distribution Scheduling (IPODS) Problem (see e.g., the survey by Chen (2010)) or the Scheduling with Interplant Transportation (SIT) Problem (see e.g., Hurink and Knust (2001), Lee and Chen (2001), Lee and Strusevich (2005)) consider simultaneous scheduling and transportation decisions. The goal of the IPODS Problem is to simultaneously find a production schedule and plan the delivery of finished products to customers, while SIT Problems consider shop environments with transportation times between the machine. IPODS Problems with individual and immediate delivery, i.e., where each finished job is directly taken to the customer, are equivalent to the pure scheduling problem in many cases. The structure is similar to the one of Network ScheLoc Problems. However, since the transportation is done after the processing, such that it does not influence the scheduling data of the jobs, and there are no location decisions made, solution approaches are not applicable to the ScheLoc Problem.
The SIT Problem considers more difficult routing decisions and is therefore NP-hard except for a few special cases. It is similar to a more involved ScheLoc Problem, the Shop ScheLoc Problem, but again does not integrate location decisions, making algorithms unapplicable to ScheLoc Problems.

In this study we will consider a special case of the Single Machine Universal Network (SMUN) ScheLoc Problem, that can be solved in polynomial time. We will formally introduce the SMUN ScheLoc Problem in Section 2, introduce a polynomial time algorithm for a Universal Scheduling Problem with preemption in Section 3 and use this algorithm in a polynomial time algorithm for the SMUN ScheLoc Problem with preemption in Section 4.

2 Problem Definition

We consider the SMUN ScheLoc Problem. This is the problem of simultaneously finding an optimal machine location in the network and a schedule that respects all given scheduling constraints and the release dates induced by travel times of jobs to the machine location minimizing the universal ScheLoc objective function. Formally, we consider the following problem:
Let an undirected graph $G = (V, E)$ with edge lengths l_e for all $e \in E$ be given and let $A(G)$ denote the set of locations in the network, i.e., the nodes and arcs of the network. A set of jobs $J = \{J_1, \ldots, J_n\}$ is given that has to be processed on a single machine X. To each job J_i a storage location $a_i \in V$, a storage arrival time $\sigma_i \geq 0$, and a travel speed $\nu_i > 0$ is associated. For processing jobs have to be taken from their storage location to the machine location. Job J_i becomes available at time σ_i and will immediately move towards the machine along the shortest path at its speed ν_i. Shortest paths $dist(x, y)$ between $x, y \in A(G)$ in the network are calculated using the lengths of edges. For $x, y \in V$ the shortest path can be easily calculated by a shortest paths algorithm. For y located on edge $e = (v_k, v_l)$ we can write $y = (e, \alpha)$ for $\alpha \in [0, 1]$ with

$$dist(x, y) = \min\{dist(x, v_k) + \alpha l_e, dist(x, v_l) + (1 - \alpha) l_e\}.$$

The arrival time of job J_i at the machine location X denotes the release date $r_i(X)$ of the job, i.e., the earliest time at which it can start processing. Using travel speed and storage arrival times of the jobs, location dependent release dates can be calculated as

$$r_i(X) = \sigma_i + \tau_i dist(a_i, X),$$

where $\tau_i = 1/\nu_i > 0$.
Denote by $\mathcal{S}^n$ the set of all feasible schedules with n jobs on a single machine that respect the location dependent release dates and the restrictions of the scheduling problem. These restrictions are dependent on the considered problem and may include e.g., preemption (pmtn, processing of jobs can be interrupted and continued at a later point in time), precedence constraints (if $i \to j$, job J_j cannot be processed before the completion of job J_i) or deadlines $\bar{d}_i$ (each job i has to be completed before its deadline).
The objective is to find a location for machine X in $A(G)$ and a schedule $S = (S_1, ..., S_n) \in \mathcal{S}^n$ allocating time intervals for jobs J_i on machine X such that each job is fully processed and the universal ScheLoc objective function $F_{\Delta,\varphi}(C(S, X))$ is minimized. This function is defined as $F_{\Delta,\varphi}(C(S, X)) = \sum_{i=1}^{n} \delta_i f_{\varphi(i)}\left(C_{\varphi(i)}(S, X)\right)$, where $\Delta = (\delta_1, \ldots, \delta_n) \in \mathbb{R}^n$ is a weight vector and $\varphi \in \Pi_n$ a permutation satisfying

$$f_{\varphi(1)}\left(C_{\varphi(1)}(S, X)\right) \leq \ldots \leq f_{\varphi(n)}\left(C_{\varphi(n)}(S, X)\right).$$

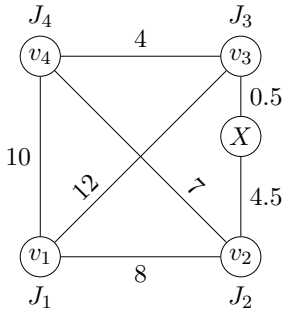
We denote by $C_{\varphi(i)}(S, X)$ the completion time of job $J_{\varphi(i)}$ dependent on the schedule $S \in \mathcal{S}^n$ and the machine location $X \in A(G)$. We assume f_i to be non-decreasing functions in the completion times, i.e., $C_{\varphi(i)}(S, X) \leq C_{\varphi(j)}(S, X)$ implies $f_{\varphi(i)}\left(C_{\varphi(i)}(S, X)\right) \leq f_{\varphi(i)}\left(C_{\varphi(j)}(S, X)\right)$.

The SMUN ScheLoc Problem is formally defined as:

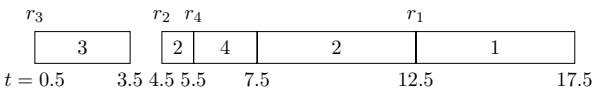
$$\begin{aligned} \min \quad & F_{\Delta,\varphi}\left(C\left(S,X\right)\right) = \sum_{i=1}^{n} \delta_i f_{\varphi(i)}\left(C_{\varphi(i)}\left(S,X\right)\right) \\ \text{s.t.} \quad & f_{\varphi(1)}\left(C_{\varphi(1)}\left(S,X\right)\right) \leq \ldots \leq f_{\varphi(n)}\left(C_{\varphi(n)}\left(S,X\right)\right), \\ & S \in \mathcal{S}^n, \\ & X \in A\left(G\right). \end{aligned} \tag{1}$$

If S and X are clear from the context we will shortly write $F_{\Delta,\varphi}\left(C\right)$ and $f_{\varphi(i)}\left(C_{\varphi(i)}\right)$ instead of $F_{\Delta,\varphi}\left(C\left(S,X\right)\right)$ and $f_{\varphi}\left(i\right)\left(C_{\varphi(i)}\left(S,X\right)\right)$ respectively. Note that the objective function $F_{\Delta,\varphi}$ is a pure scheduling objective, the link between the two problems is in computing a location-dependent schedule.
The big advantage of the universal objective function is its great modelling potential. It contains as special cases some well-known scheduling objective functions, e.g., for $f_{\varphi(i)}\left(C_{\varphi(i)}\right) = C_{\varphi(i)}$ for all $i = 1, \ldots, n$ we get for specific choices of Δ the problem of minimizing the makespan ($\Delta = (0, \ldots, 0, 1)$) or minimizing the total completion time ($\Delta = (1, \ldots, 1)$), but also many new problems like the Total s-Maximum Completion Time ScheLoc Problem ($\Delta = (0, \ldots, 0, 1, \overset{s}{\ldots}, 1)$) with $s \in \{1, \ldots, n\}$ or the Balanced ScheLoc Problem ($\Delta = (-1, 0 \ldots, 0, 1)$). By looking at the unified approach we can solve a large number of problems with the same general algorithm. For more details to Universal ScheLoc Problems we refer the reader to Kalsch (2009). In the following we assume that $\Delta \geq 0$.

Example 1. Consider the graph with 4 nodes and 4 jobs with job locations $J_i = v_i$ as in Figure 1(a). Processing times are given by $p_1 = 5$, $p_2 = 6$, $p_3 = 3$, and $p_4 = 2$. Furthermore, we have $\sigma_i = 0$ for all J_i and $\nu_i = 1$ for all J_i, i.e., all jobs are at their storage locations at time 0 and all move with the same speed $\tau = 1$. We choose $\Delta = (1, \ldots, 1)$ and $f_{\varphi(i)}\left(C_{\varphi(i)}\right) = C_{\varphi(i)}$, i.e., we want to minimize the total completion time. As special scheduling constraints we allow preemption of jobs.
The optimal machine location can be found in $X = (v_2, \alpha = 0.9)$. An optimal schedule for this location is given in Figure 1(b).



(a) Graph $G = (V, A)$ with $l\left(e\right)$ for $e \in A$ and optimal machine location $X \in A\left(G\right)$



(b) Optimal schedule for machine location X

Fig. 1. A ScheLoc Instance

3 The Universal Scheduling Problem

The Universal Scheduling Problem is defined as follows (see Kalsch (2009)):

$$\begin{aligned} \min \quad & F_{\Delta,\varphi}\left(C\left(S\right)\right) = \sum_{i=1}^{n} \delta_i f_{\varphi(i)}\left(C_{\varphi(i)}\left(S\right)\right) \\ \text{s.t.} \quad & f_{\varphi(1)}\left(C_{\varphi(1)}\left(S\right)\right) \leq \ldots \leq f_{\varphi(n)}\left(C_{\varphi(n)}\left(S\right)\right), \\ & S \in \mathcal{S}^n. \end{aligned} \tag{2}$$

It is the problem of finding a feasible schedule, minimizing a universal objective function with weight vector Δ and permutation φ that is dependent on the completion times in the schedule S.
If we fix the machine location X in the Universal ScheLoc Problem, this implies fixed release dates for all jobs. These release dates together with the scheduling constraints of the ScheLoc Problem, yield a restricted set $\mathcal{S}^n$ that is dependent only on scheduling data. Therefore, the Universal ScheLoc Problem with a fixed machine location is equivalent to a pure Universal Scheduling Problem.
We consider the Universal Scheduling Problem $1|r_i, pmtn|F_{\Delta,\varphi}\left(C\right)$ with $\Delta \geq 0$ and f_i non-decreasing for all $i = 1, \ldots, n$. The following result is easy to show:

Theorem 1. *Scheduling Problem $1|r_i, pmtn|F_{\Delta,\varphi}\left(C\right)$ for $\Delta \geq 0$ and f_i non-decreasing for all $i = 1, \ldots, n$ can be solved by the shortest remaining processing time rule (SRPT).*

4 A Polynomial Time Algorithm for the SMUN ScheLoc Problem with Preemption

We look at the SMUN ScheLoc Problem with preemption for $\Delta \geq 0$ and f_i non-decreasing and linear in the completion times for all $i = 1, \ldots, n$, i.e., we look at the ScheLoc problem where jobs may be interrupted during processing and the universal objective function $F_{\Delta,\varphi}\left(C\right)$ is minimized. We propose a polynomial time algorithm for this problem.
In this algorithm we first want to find a finite dominating set (FDS) of locations of polynomial size. In each location of the FDS we can transform the ScheLoc Problem into a pure Scheduling Problem as in Section 3 and solve it by Theorem 1. Choosing the location that yields the best scheduling objective function value gives an optimal solution to the ScheLoc Problem.
Such an FDS can be obtained as follows: We know because of the linearity of f_i that in regions where the completion times are linear, the minimal objective function value for this region is obtained in a borderpoint of this region. Since we consider a network, such regions will be subedges and borderpoints will be their endpoints. Therefore, finding a polynomial number of regions with linear completion times yields a polynomial FDS.
Let

$$BN_{s,t} = \left\{x \in A\left(G\right) \mid r_s\left(x\right) = r_t\left(x\right)\right\} \tag{3}$$

be the points in the network where release dates of jobs s and t coincide, called the bottleneck points of s and t, and

$$\mathcal{BN} = \bigcup_{s,t \in J} BN_{s,t}$$

the set of all bottleneck points. Let $\mathcal{IP}_1$ be the set of points where two jobs have identical remaining processing times, and let $\mathcal{IP}_2$ be the points where $r_{\varphi(i)} = C_{\varphi(i-1)}\left(S, X\right)$. It can be shown that these points together with the node set V of the graph yield regions

with linear completion times and that there are only polynomially many of these points. Therefore, we get the following result:

Theorem 2. *$V' = V \cup \mathcal{BN} \cup \mathcal{IP}_1 \cup \mathcal{IP}_2$ is a polynomial size FDS for the SMUN ScheLoc Problem with preemption for $\Delta \geq 0$ and f_i non-decreasing and linear for all $i = 1, \ldots, n$.*

If we can find these points in polynomial time, we get a polynomial time algorithm for the SMUN ScheLoc Problem. Since the remaining processing times and the completion times are dependent on the schedule, we can calculate $\mathcal{IP}_1$ and $\mathcal{IP}_2$ only for a given schedule. For this reason we consider one edge $e = (v_k, v_l)$ at a time and start in one endnode of the edge, say v_k. In this location we can calculate the optimal schedule by the SRPT rule. Then we consider small changes $\epsilon > 0$ of the location, i.e., we move the location towards v_l. This change of location induces a change of the release dates, the remaining processing times, and the completion times of the jobs. We can show that using the schedule of node v_k, we can find the minimum ϵ yielding a point v_{k_1} in V' in polynomial time. Since by Theorem 2 the completion times are linear on the subedge (v_k, v_{k_1}), this subedge is dominated by one of its endpoints. We calculate the optimal schedule in v_{k_1} and continue this procedure finding points $v_{k_2}, \ldots, v_{k_s}$ until the smallest ϵ yielding a point in V' is larger than $dist\,(v_{k_s}, v_l)$, i.e., there is no further point in V' on edge e. We calculate the optimal schedule in node v_l and continue with the next edge. This procedure yields the following result:

Theorem 3. *The SMUN ScheLoc Problem with preemption for $\Delta \geq 0$ and f_i non-decreasing and linear for all $i = 1, \ldots, n$ can be solved in polynomial time.*

Acknowledgements

Partially supported by the Federal Ministry of Education and Research Germany, grant DSS_Evac_Logistic, FKZ 13N12229.

References

Chen, Z.-L., 2010, "Integrated Production and Outbound Distribution Scheduling: Review and Extensions", *Oper. Res.*, Vol. 58, pp. 130-148.

Elvikis D., H.W. Hamacher and M.T. Kalsch, 2008, "Simultaneous Scheduling and Location (ScheLoc): The Planar ScheLoc Makespan Problem", *J. of Sched.*, Vol. 12, pp. 361-374.

Hennes H., 2005, "Integration of Scheduling and Location Models", *PhD Thesis*, University of Kaiserslautern.

Hennes H., H.W. Hamacher, 2007, "Integrated Scheduling and Location Models: Single Machine Makespan Problems", *Stud. in Locat. Anal.*, Vol. 16, pp. 77-90.

Hurink, J., S.Knust, 2001, "Makespan minimization for flow-shop problems with transportation times and a single robot", *Discrete Appl. Math.*, Vol. 50, pp. 199-216.

Kalsch M.T., 2009, "Scheduling - Location (ScheLoc): Models, Theory and Algorithms", *PhD Thesis*, University of Kaiserslautern.

Kalsch M.T., Z. Drezner, 2010, "Solving scheduling and location problems in the plane simultaneously", *Comput. and Oper. Res.*, Vol. 37, pp. 256-264.

Lee, C.-Y., Z.-L. Chen, 2001, "Machine scheduling with transportation considerations", *J. of Sched.*, Vol. 4, pp. 3-24.

Lee, C.-Y., V. A. Strusevich, 2005, "Two-machine shop scheduling with an uncapacitated interstage transporter", *IIE Trans.*, Vol. 37, pp. 725-736.

Turner L.R., 2013, "Universal Combinatorial Optimization", *PhD Thesis*, University of Kaiserslautern.

First Results on Quality-Oriented Scheduling of Flexible Projects

Carolin Kellenbrink

Leibniz Universität Hannover, Germany
carolin.kellenbrink@prod.uni-hannover.de

Keywords: Project Scheduling, RCPSP-PS-Q, Flexible Projects, Quality.

1 Introduction

The established resource-constrained project scheduling problem (RCPSP) is based on the assumption that the project structure with its activities and its precedence restrictions is predefined. The aim is to determine starting and ending times for all activities in this structure under consideration of scarce capacities. There is no decision necessary whether to implement an activity at all. In contrast to this, in so-called flexible projects the project structure is not known beforehand. In addition to the determination of the starting and ending times of the activities, it is necessary to decide whether to implement particular optional activities and the associated precedence relations at all.

Flexible projects arise for example when complex capital goods are regenerated, e.g., aircraft engines. As there exist different ways to repair the good, the project structure is not given in advance. The decision for one of the alternative project structures determines the costs and influences the project's makespan. Furthermore, the chosen project structure defines the qualitative characteristics of the regenerated good. Thereby, the revenues of the project may vary, because the customer's willingness to pay increases with a higher quality.

In this talk a mathematical model formulation is presented for the consideration of these qualitative aspects when scheduling flexible projects. This work is described in detail in Kellenbrink (2014). It is an extension of the work presented in Kellenbrink and Helber (2013).

2 Problem statement

In order to execute a flexible project successfully, some activities have to be implemented in any case. In contrast to these mandatory activities, other activities have an optional character. The selection of a feasible combination out of these optional activities is based on

1. choices between alternative activities, (eventually)
2. causing the (non-)implementation of further activities, and/or
3. activating further choices.

The decision for one project structure and for one appropriate schedule affects the project's makespan and the costs as well as the qualitative features of the project's outcome. When scheduling such projects, the optimization should take all these – in many cases competing – aspects into account. They can be harmonized by maximizing the profit, cf. Figure 1. We assume that the customer's willingness to pay and thereby the revenues decrease with a lower quality and an increasing makespan.

To describe the qualitative characteristics of a chosen project structure so-called quality levels are introduced. Such a quality level is defined by the associated revenue and by the

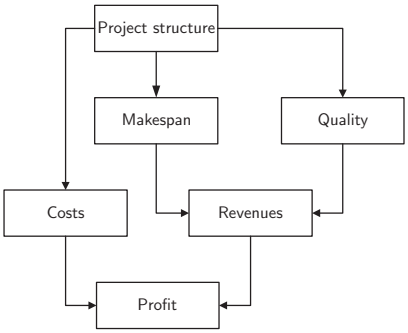


Fig. 1. Influence of a project structure on the profit

minimal requirements which have to be achieved by the implemented activities. Thereby, different attributes, e.g., fuel consumptions and emissions in future operations, can be considered.

3 Related literature

Resource-constrained project scheduling receives much attention in research literature, so that there are many publications available (cf. for a recent literature survey Hartmann and Briskorn (2010)). Hence, only papers which consider qualitative aspects are brought up here.

Vanhoucke (2006) introduces the concept of quality-dependent time slots in which a deviation from an optimal time window results in a lower quality and thereby higher costs. Other approaches which explicitly consider quality characteristics can be found mainly in the context of the multi-mode resource-constrained project scheduling problem (MRCPSP). In this problem setting the achieved quality can vary due to different modes. Icmeli-Tukel and Rom (1997) as well as Erenguc and Icmeli-Tukel (1999) measure the quality as the rework which is necessary if the project is scheduled in specific modes. They focus on a trouble-free realization of the project instead of on the quality of the project's outcome.

Other approaches focus on the quality of the project's outcome as it is intended in this paper. In some papers a minimum quality is requested, e.g., Tiwari *et al.* (2009) as well as Li and Womer (2008). A further maximization of the quality is not considered. Tareghian and Taheri (2006, 2007) consider the quality maximization in the objective function. In these approaches requirements concerning the maximal costs and the maximal makespan have to be met. To the best of our knowledge, there is no approach available in which the makespan, the costs and the quality are optimized simultaneously.

4 Formal description of the RCPSP-PS-Q

The resource-constrained project scheduling problem with a flexible project structure and quality (RCPSP-PS-Q) uses the established binary variable x_{jt} known from the classical RCPSP (Pritsker *et al.* 1969). This variable equals 1 if an activity j is finished in period t and 0 otherwise. In addition, this variable can reflect the decision on the flexible project structure: For a not implemented activity, the variable equals 0 for the whole planning horizon. Due to the usage of this established variable, the RCPSP-PS-Q resembles the RCPSP in many constraints (cf., e.g., Klein (2000), pp.79-80). For modeling the achieved quality, we introduce a binary variable y_{lt}. This variable equals 1 if the project is completed in quality level l in period t and 0 otherwise.

Model RCPSP-PS-Q

$$\max\ Z = \sum_{l\in\mathcal{L}}\sum_{t=1}^{T} u_{lt}\cdot y_{lt} - \sum_{j=2}^{J-1} c_j \cdot \sum_{t=1}^{T} x_{jt} \tag{1}$$

subject to

$$\sum_{t=1}^{T} x_{jt} = 1 \qquad j\in\mathcal{V} \tag{2}$$

$$\sum_{i\in\mathcal{W}_e}\sum_{t=1}^{T} x_{it} = \sum_{t=1}^{T} x_{a(e),t} \qquad e\in\mathcal{E} \tag{3}$$

$$\sum_{t=1}^{T} x_{it} = \sum_{t=1}^{T} x_{jt} \qquad e\in\mathcal{E};\quad j\in\mathcal{W}_e;\quad i\in\mathcal{B}_j \tag{4}$$

$$\sum_{t=1}^{T} t\cdot x_{it} \le \sum_{t=1}^{T}(t-d_j)\cdot x_{jt} + T\cdot\left(1-\sum_{t=1}^{T} x_{jt}\right) \qquad j\in\mathcal{J};\quad i\in\mathcal{P}_j \tag{5}$$

$$\sum_{j=1}^{J} k_{jr}\cdot \sum_{\tau=t}^{t+d_j-1} x_{j\tau} \le K_r \qquad r\in\mathcal{R};\quad t\in\mathcal{T} \tag{6}$$

$$\sum_{j=1}^{J} k_{jr}\cdot \sum_{t=1}^{T} x_{jt} \le K_r \qquad r\in\mathcal{N} \tag{7}$$

$$q_o^{\text{Basis}} + \sum_{j=2}^{J-1} q_{jo}\cdot\sum_{t=1}^{T} x_{jt} \ge w_{ol} - M\cdot\left(1-\sum_{t=1}^{T} y_{lt}\right) \qquad o\in\mathcal{O};\quad l\in\mathcal{L} \tag{8}$$

$$\sum_{l\in\mathcal{L}} y_{lt} = x_{Jt} \qquad t\in\mathcal{T} \tag{9}$$

The objective function (1) defines the profit as the achieved revenue u_{lt} in quality level l and finishing time t minus the sum of the costs c_j of all implemented activities j.

Constraints (2) - (5) model the flexible project structure. Equation (2) addresses the implementation of the mandatory activities $j\in\mathcal{V}$ (including (dummy) activities 1 and J). For each choice e which is activated by the implementation of its triggering activity $a(e)$, Equation (3) chooses exactly one activity $i\in\mathcal{W}_e$ and assures the implementation of this activity. If $a(e)$ is not implemented and thereby the choice e is not triggered, no activity $i\in\mathcal{W}_e$ is implemented. Equation (4) assures that all activities $i\in\mathcal{B}_j$ which are caused by an activity $j\in\mathcal{W}_e$ are implemented if and only if the activity j is implemented too.

Constraint (5) reflects the precedence constraints between a preceding activity $i\in\mathcal{P}_j$ and its succeeding activity j . Thereby, the second summand on the right-hand side assures that this constraint only limits the solution space if both activities i and j are implemented.

Constraint (6) defines the limitation of the renewable resources $r\in\mathcal{R}$. The available capacity K_r may not be exceeded by the capacity consumption k_{jr} of all activities which are executed in period t. The capacity constraints of the non-renewable resources $r\in\mathcal{N}$ are reflected by Constraint (7).

Inequality (8) enforces the variable y_{lt} to 0 for all quality levels l which are not met. This is the case if for at least one attribute o the sum of the basic quality q_o^{Basis} and the enhancements q_{jo} due to the implemented optional activities j does not equal or exceed the minimal requirement w_{ol}. Constraint (9) selects exactly one out of the feasible quality

levels l. In interaction with the objective function always the quality level with the highest revenue is chosen. In addition, this equation assures that the binary variable y_{lt} only equals 1 in that specific period in which the whole project – reflected by dummy job J – is finished.

5 Outlook

In practical applications, we often have the possibility to use overtime in order to accelerate the project. In this case, it can be economically efficient to decrease the makespan and therefore increase the revenues by accepting overtime costs. This modification of the problem will necessitate substantial changes especially of possible solution procedures, because known approaches concentrated either on makespan minimization or on resource leveling, but not on both simultaneously.

Acknowledgements

The author thanks the German Research Foundation (DFG) for the financial support of this research project in the CRC 871 'Regeneration of complex durable goods'.

References

Erenguc, S.S. and O. Icmeli-Tukel, 1999, "Integrating quality as a measure of performance in resource-constrained project scheduling problems". In: J. Węglarz, "Project Scheduling: Recent Models, Algorithms and Applications", pp. 433-450, Kluwer Academic Publishers, Boston.

Hartmann, S. and D. Briskorn, 2010, "A survey of variants and extensions of the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 207, pp. 1-14.

Icmeli-Tukel, O. and W.O. Rom, 1997, "Ensuring quality in resource constrained project scheduling", *European Journal of Operational Research*, Vol. 103, pp. 483–496.

Kellenbrink, C., 2014, "Ressourcenbeschränkte Projektplanung für flexible Projekte", Springer Gabler, Wiesbaden.

Kellenbrink, C. and S. Helber, 2013, "Scheduling resource-constrained projects with a flexible project structure", *Diskussionspapier der wirtschaftwissenschaftlichen Fakultät der Leibniz Universität Hannover Nr. 511.*

Klein, R., 2000, "Scheduling of Resource-Constrained Projects", Kluwer Academic Pubslishers, Boston.

Li, H. and K. Womer, 2008, "Modeling the supply chain configuration problem with resource constraints", *International Journal of Project Management*, Vol. 26, pp. 646-654.

Pritsker, A.A.B., L.J. Watters and P.M. Wolfe, 1969, "Multiproject scheduling with limited resources: A zero-one programming approach", *Management Science*, Vol. 16, pp. 93-108.

Tareghian, H.R. and S.H. Taheri, 2006, "On the discrete time, cost and quality trade-off problem", *Applied Mathematics and Computation*, Vol. 181, pp. 1305-1312.

Tareghian, H.R. and S.H. Taheri, 2007, "A solution procedure for the discrete time, cost and quality tradeoff problem using electromagnetic scatter search", *Applied Mathematics and Computation*, Vol. 190, pp. 1136-1145.

Tiwari, V., J.H. Patterson and V.A. Mabert, 2009, "Scheduling projects with heterogeneous resources to meet time and quality objectives", *European Journal of Operational Research*, Vol. 193, pp. 780-790.

Vanhoucke, M., 2006, "Scheduling an R&D Project with Quality-Dependent Time Slots". In: M. Gavrilova, O. Gervasi, V. Kumar, C.J.K. Tan, D. Taniar, A. Laganà, Y. Mun and H. Choo, "International Conference on Computational Science and Its Applications", pp. 621-630, Springer, Berlin.

A Best Possible Algorithm for Semi-Online Scheduling

Hans Kellerer[1], Vladimir Kotov[2] and Michaël Gabay[3]

[1] Institut für Statistik und Operations Research, Universität Graz, Universitätsstraße 15, A–8010 Graz, Austria
hans.kellerer@uni-graz.at
[2] Belarusian State University, Faculty of Applied Mathematics and Computer Science, Nezavisimosty ave. 4, 220030 Minsk, Belarus
kotovVM@bsu.by
[3] Grenoble-INP / UJF-Grenoble 1 / CNRS, G-SCOP UMR5272 Grenoble, F-38031, France
michael.gabay@g-scop.grenoble-inp.fr

Keywords: Semi-online scheduling, Competitive analysis, Multiprocessor scheduling

1 Introduction

The well-known classical multiprocessor scheduling problem is a fundamental and well-investigated scheduling problem both in the offline and the online setting. A set of n independent jobs is to be processed on m parallel, identical machines in order to minimize the makespan. In the online scenario each job must be immediately and irrevocably assigned to one of the machines without any knowledge on future jobs. This problem was first investigated by Graham who showed that the list scheduling algorithm has a performance ratio of exactly $2-1/m$ (Graham 1966) and is best possible for $m \leq 3$ (Faigle *et al.* 1989). A long list of improved algorithms has since been published. The best heuristic known for this problem is contained in (Fleischer and Wahl 2000). They designed an algorithm with competitive ratio smaller than 1.9201 when the number of machines tends to infinity. The best lower bound is 1.88 (Rudin 2001).

Recent research has focused on scenarios between offline and online scenarios where the online constraint is relaxed but no full information on the input data is available. For a survey on recent advances we refer to (Albers 2013). In this paper we consider the online multiprocessor scheduling with the additional assumption that the sum of processing times is given in advance. The resulting problem is denoted as the *semi-online scheduling problem*.

A less general semi online problem has been introduced in (Azar and Regev 2001), who labeled it as the *online bin stretching problem*. A sequence of items is given and we know that these items *can* be packed into m bins of unit size. The items are to be assigned online to the bins and the aim is to minimize the *stretching factor* of the bins, i.e., to stretch the sizes of the bins as least as possible such that all items fit in the bins. Thus, the bin stretching problem can be interpreted as a special case of the semi-online scheduling problem where, instead of the total processing time, the value of the optimal makespan is known in advance. In the semi-online scheduling problem a total processing time of m gives a lower bound of one for the optimal makespan, but it is possible that some items have length greater than one which makes the analysis more complicated. For the bin stretching problem, a sophisticated proof for an algorithm with stretching factor 1.625 was given in (Azar and Regev 2001). In (Kellerer and Kotov 2013) an algorithm with stretching factor 11/7 was proposed by using techniques of grouping bins into batches.

In a previous paper (Kellerer *et al.* 1997), an algorithm with performance ratio 4/3 for the semi-online scheduling problem on two machines was given. This bound is best possible for $m = 2$. In (Cheng *et al.* 2005) an algorithm is presented with performance ratio 1.6 for semi-online scheduling for an arbitrary number of machines. Moreover, the authors established a lower bound of 1.5 for $m \geq 6$ machines (Angelelli *et al.* 2004) contains a

deterministic algorithm with performance ratio $(1+\sqrt{6})/2 \approx 1.725$ and a lower bound of 1.565. Recently, an improved lower bound is developed showing that no deterministic semi-online algorithm for semi-online scheduling can attain a competitive ratio smaller than ≈ 1.585 when m tends to infinity (Albers and Hellwig 2012). Moreover, the authors give a simple algorithm with performance ratio 1.75.

In this paper we will present an algorithm with competitive ratio ≈ 1.585, which is equal to the lower bound of Albers and Hellwig. Thus, our algorithm is best possible for large values of m.

2 Problem Definition and Notation

We are given m identical machines and a sequence of jobs $1, \ldots, n$ with processing times $p_1, \ldots, p_n$, which are to be assigned online to one of the machines. We assume that the sum $S = \sum_{j=1}^{n} p_j$ of the jobs processing times is given in advance. W.l.o.g., $S = m$. The objective is to minimize the makespan. We will usually speak of bins $B_1, \ldots, B_m$ instead of machines and of items with weights p_j instead of jobs with processing times p_j.

The *weight* of a bin B is defined as the sum of the weights of all items assigned to B, and is denoted by $w(B)$. When we speak of *time* j, we mean the state of the system just before item j is assigned.

Let α be the positive root of the function $f(x) = 4x^3 + 4x^2 - 2x - 1$. We will show that the competitive ratio of our algorithm is equal to $1 + \alpha \approx 1.58504$.

The items are divided into several classes. Items with weights in $(0; \alpha \approx 0.585]$ are called *small*, items in $(\alpha; \frac{1}{2\alpha} \approx 0.8546]$ are called *medium* and items larger than $\frac{1}{2\alpha}$ are called *large*.

A bin B with $w(B) \in (0; \frac{\alpha}{2} \approx 0.2925]$ is called *tiny*, for $w(B) \in (0; \alpha]$ it is called *small*, for $w(B) \in (\alpha; \frac{1}{2\alpha}]$ it is called *medium*, for $w(B) \in (\frac{1}{2\alpha}; 1]$ it is called *big* and for $w(B) > 1$ it is called *huge*. Big and huge bins are also denoted as *large*. Notice that each tiny bin is small. If B contains a large item, it is called a *large item bin*. The number of empty bins, tiny bins, small bins, medium bins and big bins is abbreviated by eB, tB, sB, mB and bB, respectively. When a bin is *closed*, no more items can be assigned to that bin. Otherwise, it is called *open*. If we want to specify the time j, we write eB^j, $w^j(B)$ instead of eB, $w(B)$, respectively.

We denote by LB a lower bound for the optimal makespan which is recalculated throughout the algorithm. Since $S = m$, we have $LB \geq 1$. Each bin of the heuristic has a *capacity* of $(1+\alpha)LB$ and we say that an item j *fits* in a bin B, if $w(B) + p_j \leq (1+\alpha)LB$.

3 Description of the Algorithm

In this section we will present a best possible algorithm for the semi-online scheduling problem with competitive ratio $1 + \alpha \approx 1.585$. This algorithm is split into two parts. The first part (called Phase 1) is an adaption of Phase 1 for the bin stretching problem as described in (Kellerer and Kotov 2013). It runs until the number of empty bins is around one third of the number of small bins.

Denote by $q_1, \ldots, q_j$ the weights of the items at time j sorted in decreasing order, i.e., $q_1 \geq q_2 \geq \ldots \geq q_j$. Then, an obvious lower bound LB for the optimal makespan is given by

$$LB = \max\{1, q_1, q_m + q_{m+1}\}. \tag{1}$$

A formal description of Phase 1 of the algorithm is depicted in Figure 1. Note that the numbering reflects the priority rules for the assignment of item j, i.e., a packing option is skipped if no bin exists which fulfills the related condition and the next option is checked.

Initially, the lower bound LB is set to one and it is recalculated each time when a new item j arrives. Moreover, all bins are open in the beginning.

Let j be the current item to be assigned. Recalculate LB according to (1).

1. j is small:
   (a) Put j in a large item bin B if j fits in B, i.e., if $w(B) + p_j \leq (1+\alpha)LB$.
   (b) Put j in a small bin B if B remains small, i.e., $w(B) + p_j \leq \alpha$.
   (c) Put j in an empty bin.
2. j is medium:
   (a) Put j in an empty bin.
3. j is large:
   (a) Put j in the small bin with largest weight.
   (b) Put j in an empty bin.

Stopping condition: If

$$sB = 3eB + \lambda, \quad 0 \leq \lambda \leq 3, \tag{2}$$

close all huge bins and goto Phase 2.

Fig. 1. Algorithmic description of Phase 1

It can be shown that after Phase 1, there are no small bins or no big bins, i.e., $sB = 0$ or $bB = 0$. Hence, we distinguish two cases for the algorithm. Consider first $sB = 0$. W.l.o.g., assume that the bins $B_1, \ldots, B_k$ are open after Phase 1 and the bins $B_{k+1}, \ldots, B_m$ are closed. Moreover, the open bins shall be sorted in decreasing order of their weights. During Phase 2a an item is assigned to the bin with largest weight. If it does not fit, it is packed in the bin with smallest weight. Afterwards, the bin with largest weight is closed. Throughout the algorithm the open bin with maximum weight is denoted by $B^{\max}$ and the open bin with minimum weight by $B^{\min}$, respectively. A formal description of Phase 2a of the algorithm is depicted in Figure 2.

Initialization: Set $s = k$ (number of open bins).

Iteration j:

1. Let j be the current item to be assigned. Recalculate LB according to (1).
2. If $s = 1$, assign the remaining items to the last open bin and stop.
3. Put j in bin $B^{\max}$ if j fits in it. If j is put in bin $B^{\max}$ and $w(B^{\max}) + p_j > 1$, close bin $B^{\max}$ and set $s = s - 1$.
4. If j does not fit in $B^{\max}$, put j in bin $B^{\min}$, close bin $B^{\min}$ and set $s = s - 1$.

Fig. 2. Algorithmic description of Phase 2 with no small bins

Phase 2b deals with the case where there are no big bins. By condition (2) there are $4k + \lambda$ empty or small bins after Phase 1 for some integer $k \geq 0$. Among these $4k + \lambda$ bins there are k empty bins and the remaining $3k + \lambda$ bins are small. These bins are now partitioned into k so-called *4-batches* $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$. Each 4-batch $\mathcal{B}$ consists of four bins, where the fourth bin B_4 is an empty bin. If $\lambda > 0$, there is an additional batch $\mathcal{B}_{k+1}$ of at most three non-empty bins. W.l.o.g., a possibly existing tiny bin shall correspond to bin B_1 of 4-batch $\mathcal{B}_1$. Therefore, we may assume that all other non-empty bins are small, but not tiny. Set $k' = k + 1$, if $\lambda > 0$ and $k' = k$, otherwise.

Throughout Phase 2b denote the open medium bin with maximum weight by $M^{\max}$, the open medium bin with second largest weight by M^2 and the open medium bin with

smallest weight by $M^{\min}$, respectively. For the assignment of items to batches, we will always use *First Fit*. The algorithm tries to assign an item j to the bins in the following order: $M^{\max}$, M^2, an open batch with smallest index if $p_j \leq (1+\alpha)/2$, an open batch with largest index if $p_j > (1+\alpha)/2$, $M^{\min}$. A formal description of Phase 2a of the algorithm is depicted in Figure 3.

Initialization: Set $\mu = \ell$ (number of medium bins) and $\tau = k'$ (number of open batches).
Iteration j:

1. Let j be the current item to be assigned. Recalculate LB according to (1).
2. $\mu \geq 1$, j fits in $M^{\max}$: assign j to $M^{\max}$. If $w^j(M^{\max}) + p_j \geq 1$, close $M^{\max}$.
3. $\mu \geq 2$, j fits in M^2: assign j to M^2. Close the bin with largest weight among $M^{\max}$ and M^2 after assigning item j. Set $\mu = \mu - 1$.
4. $\tau \geq 2, p_j \leq (1+\alpha)/2$: let $\mathcal{B}_i$ be the open batch with smallest index. Assign j to $\mathcal{B}_i$ if it fits. Otherwise, close batch $\mathcal{B}_i$ and set $\tau = \tau - 1$. If $\tau \geq 2$, assign j to $\mathcal{B}_{i+1}$.
5. $\tau \geq 2, p_j > (1+\alpha)/2$: let $\mathcal{B}_i$ be the open batch with largest index. Assign j to $\mathcal{B}_i$ if it fits. Otherwise, close batch $\mathcal{B}_i$ and set $\tau = \tau - 1$. If $\tau \geq 2$, assign j to $\mathcal{B}_{i-1}$.
6. $\tau = 1$: assign j to last open batch $\mathcal{B}_r$ if it fits.
7. $\mu \geq 3$: assign j to $M^{\min}$, close $M^{\min}$ and set $\mu = \mu - 1$.

Fig. 3. Algorithmic description of Phase 2 with no big bins

4 Conclusions

In this paper we have presented a best possible algorithm for the semi-online multiprocessor scheduling problem. It is a challenging open problem to close the large gap between lower and upper bound for the bin stretching problem.

References

Albers S., 2013, "Recent advances for a classical scheduling problem", in: Automata, languages and processing, Lecture Notes in Computer Science, Vol. 7966, Springer, Berlin, pp. 4-14.

Albers S. and M. Hellwig, 2012, "Semi-online scheduling revisited", *Theoretical Computer Science*, Vol. 443, pp. 1-9.

Angelelli E. , Nagy A.B., Speranza M.G. and Z. Tuza, 2004, "The on-line multiprocessor scheduling problem with known sum of the tasks", *Journal of Scheduling*, Vol. 7, pp. 421-428.

Azar Y. and O. Regev, 2001, "On-line bin-stretching", *Theoretical Computer Science*, Vol. 268, pp. 17-41.

Cheng T.C.E., Kellerer H. and V. Kotov, 2005, "Semi-on-line multiprocessor scheduling with given total processing time", *Theoretical Computer Science*, Vol. 337, pp. 134-146.

Faigle U., Kern W. and G. Turan, 1989, "On the performance of on-line algorithms for partition problems", *Acta Cybernetica*, Vol. 9 , pp. 107-119.

Fleischer R. and M. Wahl, 2000, "On-line scheduling revisited", *Journal of Scheduling*, Vol. 3, pp. 343-353.

Graham R.L., 1966, "Bounds for certain multiprocessor anomalies", *Bell System Technical Journal*, Vol. 45, pp. 1563-1581.

Kellerer H. and V. Kotov, 2013, "An efficient algorithm for bin stretching", *Operations Research Letters*, Vol. 41, pp. 343-346.

Kellerer H., Kotov V., Speranza M.G. and Z. Tuza, 1997, "Semi on–line algorithms for the partition problem", *Operations Research Letters*, Vol. 21, pp. 235–242.

Rudin III J.F., 2001, "Improved bounds for the on-line scheduling problem", Ph.D. Thesis, University of Texas.

The behaviour of combined neighbourhoods for large scale multi-location parallel machine scheduling problems

Louis-Philippe Kerkhove[1] and Mario Vanhoucke[234]

[1] Faculty of Economics and Business Administration, Ghent University, Belgium
louisphilippe.kerkhove@ugent.be
[2] Faculty of Economics and Business Administration, Ghent University, Belgium
mario.vanhoucke@ugent.be
[3] Technology and Operations Management Area, Vlerick Business School, Belgium
mario.vanhoucke@vlerick.com
[4] Department of Management Science and Innovation, University College London, United Kingdom
m.vanhoucke@ucl.ac.uk

Keywords: Parallel Machine Scheduling, Weighted Lateness/Tardiness, Meta-Heuristics, Neighbourhood Operators

1 Introduction

Meta-heuristic solution techniques have long been popular to solve large machine scheduling problems. These techniques often rely on neighbourhood operators to move trough the solution space in search of near optimal solutions. The quality of a neighbourhood operator can be judged based on the solution quality, measured by the relevant objective function, attained within a given time frame. Hence, the optimal neighbourhood is the one which yields the best performance within the available time frame.

Recent research [Kerkhove and Vanhoucke, 2013], has proven the effectiveness of meta-heuristic solution techniques for solving parallel machine scheduling problems with machines allocated to geographically dispersed production locations. This paper extends this previous research by investigating the behaviour of neighbourhood operators in greater detail.

The experiments presented in this paper show that the optimal neighbourhood for meta-heuristic solution procedures can be a combination of neighbourhoods, rather than a pure neighbourhood operator. More importantly, when examining the performance of different neighbourhood combinations by gradually increasing the relative fraction of one type of operator, the evolution the performance is not always linear. More specifically, unpredictable optimal points for certain combinations are consistently observed when testing on large datasets.

The optimal set of neighbourhood operators is also influenced by the size of the problem and the computation time available. Hence, the performance of neighbourhood strategies has to be controlled against both these dimensions.

The remainder of this document is organised as follows: section 2 presents an overview of the parallel machine scheduling problem being studied as well as the meta-heuristic solution techniques used. Next, section 3 shows the computational experiments and results. Finally conclusions are presented in section 4.

2 Problem description

The research presented here is based on a case study at a Belgian textile manufacturer, more precisely a production line of circular knitting machines used to produce fabrics for mattresses. The complete production line comprises multiple geographically dispersed production locations, at each of which a number of machines are operating in a parallel setup. These machines are *unrelated*, indicating that the production times of orders on these machines depend on both the type of job and the machine on which jobs are scheduled. The *changeover* times of these machines are *sequence dependent* [Zhu and Wilhelm, 2006], and jobs have *release- and due-date* restrictions. The former are strictly enforced, whereas the latter are penalised in the objective function, which is a combination of *weighted tardiness and lateness*. Lateness is defined as a binary variable indicating whether or not a job is late, whereas tardiness is defined as an integer variable signifying the number of days a given job is late. Since distinct costs are associated to these variables, they are both included in the objective function calculation.

The inclusion of geographically dispersed production locations has two very important implications for the scheduling problem. Firstly, the due date of a job is location dependent because transportation times differ between different production locations. Hence, production on distant production locations will have to be completed earlier if the same delivery deadlines are to be respected. Secondly, the objective function weights are impacted by the location where an order has been produced. More specifically, in case an order has been produced late the company will attempt to reduce the transportation time as much as possible, which of course means incurring additional costs. Naturally these costs depend on the distance an order has to travel to its delivery location. Hence, tardy orders which were produced at production locations further from customer delivery locations will have a more substantial impact on the objective function than other orders.

To avoid overfitting the specific case study, a data generation procedure was used to create sufficiently large datasets to allow for generalisations of the results. Three key datasets were created: a dataset for preliminary calibration ($P1, n = 162$)[5] containing a wide range of problem sizes, a dataset with small theoretical examples with 8 to 14 jobs to be scheduled on 2 machines ($DS1, n = 180$), and a dat set with realistically sized problem instances of 750 jobs to be scheduled on 50 to 75 machines ($DS2, n = 810$). More information on these datasets can be found in [Kerkhove and Vanhoucke, 2013].

Given the NP-hard nature of unrelated parallel machine scheduling problems [Allahverdi et al., 2008], and the very large size of the problem under study, meta-heuristic solution techniques are arguably most suited. In [Kerkhove and Vanhoucke, 2013] a hybrid meta-heuristic procedure was developed which combined simulated annealing and a genetic algorithm to solve this problem.

This approach was based on literature suggesting that combinations of memetic and local-search based meta-heuristic techniques are likely to outperform the individual strategies using either memetic or local search meta-heuristics individually [Behnamian et al., 2009]. Given that simulated annealing [Radhakrishnan and Ventura, 2000, Lin et al., 2011] and genetic algorithms [Behnamian and Ghomi, 2013] are frequently used, these techniques were chosen to be combined in an integrated solution procedure.

3 Computational experiments regarding neighbourhoods

Several neighbourhoods were tested for use in the simulated annealing component of this solution procedure: insertion, inversion and k-swaps. A k-swap is defined as a swap

[5] n signifies the number of problem instances in the dataset.

of k elements in the chromosome. Preliminary experiments indicated that these k-swap operators were superior to other neighbourhood operators and further experimentation focused on these neighbourhood operators.

To determine which of the k-swap operators yielded the best performance, more extensive experiments were carried out. These experiments also explored intermediate strategies, which combine different neighbourhoods according to predefined probabilities. These probabilities where varied with 20% increments. The results of these experiments are shown in figure 1. The x-axis on this figure indicates the fractions of swap operators used, a 2.2-swap operator indicating that a 2-swap operator is used 80% of the time and a 3-swap operator is used for the remaining 20% of the local search moves. These tests were all carried out on the preliminary dataset $P1$ with a stopping criterium of 40,000 schedule evaluations - approximately 1 second on a 2.5GHz processor.

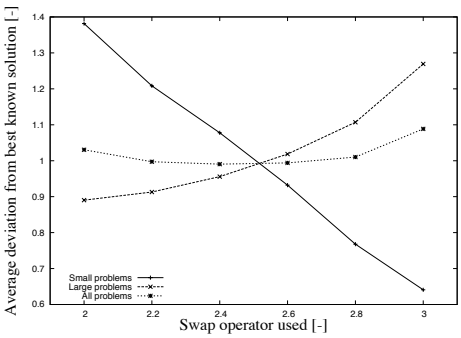


Fig. 1. Performance analysis of combinations of 2-swap and 3-swap neighbourhood operators for various problem sizes

This experiment led to the counter-intuitive result that the heuristic achieved the best solution quality by using large neighbourhoods on the smaller problem instances, and smaller neighbourhoods on the larger problem instances.

A second round of testing used a full-factorial approach testing all swap combinations up to 5-swaps, while also varying the available computation time on datasets DS1 and DS2. For smaller problems these tests confirmed the earlier observation that larger neighbourhoods indeed improved the performance of the meta-heuristic. However, most of this upward potential was tapped once 3-swap moves or higher were used.

The results for large problem instances (see table 1) revealed a much more complex pattern, where the effectiveness of the meta-heuristics did not have a strictly linear relationship with the increasing size of the neighbourhood. For relatively low computation times (1 million schedule evaluations - approximately 25 seconds on a 2.5GHz processor) the smallest neighbourhood remained the most effective. However, as the computation time increases, so does the size of the optimal neighbourhood. For situations where at least 5 million schedule evaluations were allowed, the optimal neighbourhood becomes a 2.8-swap operator.

This performance does not adhere to a simple linear trend. When first increasing the size of the neighbourhood by moving towards larger fractions of 3-swaps combined with 2-swaps the performance appears to decrease rather than increase. Notably, if only the performance of pure strategies had been tested, the optimal approach would still be the 2-swap neighbourhood - the solution quality of which is more than five times worse than that of the 2.8-swap move. This illustrates the potential gains from using combined neighbourhood strategies.

Table 1. Solution quality relative to best known solution for different neighbourhood combinations and calculation limits used in a simulated annealing meta-heuristics with re-heating and re-start capabilities.

Neighbourhood	Calculation limit (mio schedule evaluations)					
	1	2.5	5	10	20	40
2-swap	**3.5235**	**3.3576**	2.5563	2.3528	2.0811	2.021
2.2-swap	4.4727	3.4395	3.1967	2.7143	2.3867	2.3039
2.4-swap	5.6972	4.8404	3.4471	2.8238	2.7078	2.3918
2.6-swap	6.5826	4.3332	3.273	3.1459	2.3322	2.0599
2.8-swap	5.5564	3.8029	**1.4202**	**0.9023**	**0.5676**	**0.3551**
3-swap	10.0078	7.5349	5.0586	4.1312	3.2752	3.0054
3.2-swap	10.9892	9.7402	5.6465	5.1900	3.5546	3.4276
3.4-swap	11.7841	9.4398	6.4652	4.7653	4.4662	3.7759
3.6-swap	10.1112	9.2014	7.5856	5.5393	5.0843	4.4215
3.8-swap	13.2901	10.6292	8.4436	6.6913	6.187	5.5942
4-swap	14.6333	12.6707	10.8864	10.0622	8.5004	7.2355

4 Conclusions

The computational experiments above have shown that neighbourhoods for parallel machine scheduling problems may be substantially improved by using combined rather than pure strategies. More importantly, it has been shown that while there are linear trends in the behaviour of combined strategies, local optima which go against these trends do exist, and uncovering them may yield substantial improvements in terms of schedule quality.

The presentation at the PMS conference will focus on the following aspects. First of all, the specific problem under study, solution approach and experiment design will be explained in greater detail. Secondly, the detailed results of the computational experiments will be discussed. Finally, a number of recommendations for neighbourhood design will be presented.

Future research on this subject should uncover the degree in which these trends are problem specific or generally encountered in scheduling practice.

References

[Allahverdi et al., 2008] Allahverdi, A., Ng, C., Cheng, T., and Kovalyov, M. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187:985–1032.

[Behnamian and Ghomi, 2013] Behnamian, J. and Ghomi, S. F. (2013). The heterogeneous multi-factory production network scheduling with adaptive communication policy and parallel machine. *Information Sciences*, 219(0):181 – 196.

[Behnamian et al., 2009] Behnamian, J., Zandieh, M., and Fatemi Ghomi, S. (2009). Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. *Expert Systems with Applications*, 36(6):9637–9644.

[Kerkhove and Vanhoucke, 2013] Kerkhove, L.-P. and Vanhoucke, M. (2013). Scheduling of unrelated parallel machines with limited server availability on multiple production locations: a case study in knitted fabrics. *International Journal of Production Research*, (ahead-of-print):1–24.

[Lin et al., 2011] Lin, S.-W., Lee, Z.-J., Ying, K.-C., and Lu, C.-C. (2011). Minimization of maximum lateness on parallel machines with sequence-dependent setup times and job release dates. *Computers & Operations Research*, 38(5):809–815.

[Radhakrishnan and Ventura, 2000] Radhakrishnan, S. and Ventura, J. (2000). Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times. *International Journal of Production Research*, 38:2233–2252.

[Zhu and Wilhelm, 2006] Zhu, X. and Wilhelm, W. W. E. (2006). Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions*, 38(11):987–1007.

An exact algorithm for the parallel machine scheduling problem with conflicts

Daniel Kowalczyk and Roel Leus

ORSTAT, Faculty of Economics and Business, KU Leuven, Belgium
{daniel.kowalczyk ; roel.leus}@kuleuven.be

1 Introduction

We examine a generalization of the classic parallel machine scheduling problem $P||C_{\max}$. We schedule a set $J = \{1, \ldots, n\}$ of n independent jobs on a number of identical parallel machines without preemption such that the maximum completion time, or makespan, is minimized. Each job j has an associated processing time $p_j \in \mathbb{N}_0$ and is to be assigned to a single machine. The m parallel machines are gathered in set $M = \{1, \ldots, m\}$ and can each process at most one job at a time. An undirected graph $G = (J, E)$, subsequently referred to as *conflict graph*, is part of the input. If $\{j, j'\} \in E$ then jobs j and j' are *conflicting jobs*, and they need to be assigned to different machines. We call the resulting problem the parallel machine scheduling problem with conflicts (PMC). This problem is NP-hard, because PMC contains both $P||C_{\max}$ as well as the vertex coloring problem (VCP) as special cases. It can be seen that a feasible schedule exists if and only if the conflict graph can be colored with at most m colors; we will assume $m < n$ to avoid trivial solutions. We conclude that the problem at hand combines two very hard problems. Moreover, VCP is hard to approximate, and it can turn out to be hard to quickly find even a feasible schedule for a given instance.

The PMC is theoretically important because it generalizes two well-known problems in combinatorial optimization, but it also naturally arises as (sub-)problem in a number of practical applications in multiprocessor scheduling, TV advertisement scheduling and audit scheduling. Concretely, PMC is for instance a subproblem of scheduling computing services posed as the ROADEF/EURO Challenge 2012, which was furnished by Google; see ROADEF (2012). Another example stems from Gaur *et al.* (2009), who schedule television commercials in program breaks, where insertion of competing commercials into the same break is undesirable.

PMC was already studied by Bodlaender *et al.* (1994). They obtained some hardness and approximation results for specific graph types, but they did not develop any exact algorithm. Bodlaender *et al.* (1994) present approximation algorithms for the case where a k-coloring of the conflict graph is known a priori, where $k + 1 \leq m$; the worst-case ratio depends only on k and when $\frac{m}{k}$ tends to infinity then the worst-case ratio tends to 2. They also prove that, unless $P = NP$, no approximation algorithm can improve upon the worst-case ratio of 2.

Problem $P||C_{\max}$ can be seen as a sort of "dual" to the bin packing problem (BPP), where the bin capacities correspond to the makespan and the number of bins corresponds to the number of parallel machines. A similar pairing can be observed between PMC and the bin packing problem with conflicts (BPPC): the latter problem consists of packing items in a minimum number of bins of limited capacity while avoiding joint assignments of items that are in conflict. We immediately see that BPPC generalizes both BPP and VCP. This problem has been recently studied by a number of researchers, see Sadykov and Vanderbeck (2013) and Muritiba *et al.* (2010). The best exact algorithm was developed by Sadykov and Vanderbeck (2013). They used their black-box branch-and-price solver BaPCod, which

relies on a generic branching scheme and certain primal heuristics, together with a specific pricing oracle. We will use this relationship between PMC and BPPC to obtain an exact algorithm for PMC.

Notice that PMC is intuitively harder than BPPC: in PMC one has to assign jobs to m machines in such way that conflicting jobs are assigned to different machines and such that the makespan is minimized. From a VCP viewpoint, there is a hard constraint on the number of colors that can be used (namely m). As a result, for a given PMC instance we first need to verify whether there exists a m-coloring for the conflict graph: this is a priori not known for a given instance. In some sense, we have to control (minimize) the makespan and the number of colors at the same time to obtain a "good" feasible solution for PMC, whereas in BPPC we only need to control (minimize) the number of bins (colors).

2 A MIP formulation for PMC

We formulate a natural Mixed Integer Program (MIP) model for PMC, as follows. For every job j and machine i we introduce a binary variable x_{ij} that is equal to 1 if job j is assigned to machine i and 0 otherwise. We also introduce a real variable y that will equal the makespan of the schedule. A possible MIP model for PMC is then given by:

$$\text{minimize } y \tag{1}$$

$$\text{subject to } \sum_{i \in M} x_{ij} = 1 \qquad \forall j \in J \tag{2}$$

$$x_{ij} + x_{ij'} \leq 1 \qquad \forall \{j, j'\} \in E, \forall i \in M \tag{3}$$

$$\sum_{j \in J} p_j x_{ij} \leq y \qquad \forall i \in M \tag{4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall j \in J, \forall i \in M \tag{5}$$

$$y \in \mathbb{R}. \tag{6}$$

The first set of constraints (2) ensures that every job is scheduled on exactly one machine. Inequalities (3) force conflicting jobs to be scheduled on different machines. Inequalities (4) guarantee that for each machine the makespan y is at least the total processing time consumed on that machine. Although correct, this formulation is quite unpractical. One reason for the high intractability of the formulation is the inherent symmetry: re-arranging the indices of the machines leads to equivalent solutions. This has undesirable consequences in a branch-and-bound (B&B) scheme: the number of equivalent solutions is exponential in m and can lead to a lot of redundant work by a linear solver. One could resort to symmetry-breaking constraints for reducing the redundant work by a linear solver; see Berghman *et al.* (to appear). The work of Dell'Amico *et al.* (2008), however, suggests that the following approach will perform better.

Looking closely at formulation (1)–(6), one can see that the formulation is almost of a form on which Dantzig-Wolfe decomposition can be applied (see Martin, 1999). To

elaborate this, we first rewrite the formulation (1)–(6) in matrix notation, as follows:

$$\text{minimize } y \tag{7}$$

$$\text{subject to } \sum_{i \in M} \mathbb{I}_n x_i = e_n \quad , \tag{8}$$

$$\begin{pmatrix} -y \\ 0 \end{pmatrix} + \begin{pmatrix} p \\ A \end{pmatrix} x_i \leq \begin{pmatrix} 0 \\ e_{|E|} \end{pmatrix} \qquad \forall i \in M \tag{9}$$

$$x_i \in \{0,1\}^n \qquad \forall i \in M \tag{10}$$

$$y \in \mathbb{N}_0, \tag{11}$$

where $x_i = (x_{i1}, \ldots, x_{in})'$ for each $i \in M$, $p = (p_1, \ldots, p_n)$, A is the edge-node incidence matrix of the graph $G = (J, E)$, $e_n = (1, \ldots, 1)' \in \{0,1\}^n$, $e_{|E|} = (1, \ldots, 1)' \in \{0,1\}^{|E|}$ and $\mathbb{I}_n$ is the unity matrix of size n. In order to be able to apply Dantzig-Wolfe decomposition we will switch to the decision variant of the optimization problem: we introduce an upper bound C on the value of the objective function, and we denote the resulting decision problem by PMC-D$_C$ (decision problem of PMC with makespan C), which is to determine whether there exists a feasible schedule without conflicts and with maximum makespan C. Variable y then disappears form the constraints and the resulting constraint matrix has a block-angular structure.

PMC can then be solved by determining the smallest C for which PMC-D$_C$ yields a "yes" answer; this will be achieved by a binary search algorithm. We have identical machines and hence PMC-D$_C$ can be reformulated as follows: is it possible to partition the jobs in at most m stable sets of $G = (J, E)$ such that each stable set corresponds to a machine that consumes at most C time units? This coincides with the decision form of BPPC.

3 Solving PMC-D$_C$

Since PMC-D$_C$ is the decision variant of BPPC, one natural approach is to obtain sufficient conditions for a negative answer; this is done by searching for lower bounds for BPPC. Let $L(I)$ be a lower bound for an instance I, then we know that the answer to PMC-D$_C$ is "no" if $L(I) > m$. Muritiba *et al.* (2010) developed some heuristics and lower bounds for BPPC. The algorithms for the computation of the lower bound that they propose, do not make any assumptions on the conflict graph, and so we can implement their bounds into the binary search procedure. For this procedure, however, we also need an exact procedure for PMC-D$_C$, for the cases where the lower bounds are less than m. Here we can turn to solving the optimization version of PMC-D$_C$, which is of course BPPC. Notice that sometimes it can suffice to calculate the minimum objective of the LP relaxation of the optimization version of PMC-D$_C$.

Let $\mathcal{S}_C$ be the set containing all the inclusion-maximal stable sets S of $G = (J, E)$ with $\sum_{j \in S} p_j \leq C$. We introduce a binary variable λ_S for each $S \in \mathcal{S}_C$ such that λ_S is equal to 1 if the stable set is chosen and 0 otherwise. The goal is to minimize the number of stable sets of $G = (J, E)$, that we select, such that each job is contained in one machine schedule:

$$\text{minimize } \sum_{S \in \mathcal{S}_C} \lambda_S \tag{12}$$

$$\text{subject to } \sum_{S \in \mathcal{S}_C : j \in S} \lambda_S \geq 1 \qquad \text{for each } j \in J \tag{13}$$

$$\lambda_S \in \{0,1\} \qquad \text{for each } S \in \mathcal{S}_C \tag{14}$$

Objective function (12) minimizes the number of machines required for allowing a feasible solution. Constraints (13) impose that every job has to be executed on a machine. The answer to PMC-D$_C$ is "yes" if and only if the optimal solution to the problem (12)–(14) has a value that is smaller than or equal to m.

Formulation (12)–(14) is solved using branch and price (B&P): at each node of a B&B search tree, the linear relaxation of the above formulation is solved by column generation to provide a lower bound. The calculation of this bound is achieved by iteratively solving the restricted master problem (RMP), which is the linear relaxation of (12)–(14) with a restricted number of variables, and the pricing problem, which determines whether to include extra variables λ_S in the variable pool of RMP to improve its current solution (columns with negative reduced cost). The pricing problem can be seen to be a knapsack problem with conflicts (KPC); for solution procedures for KPC, we refer to Yamada *et al.* (1998), where both heuristics (greedy algorithm and local search) as well as an exact algorithm (implicit enumeration combined with an interval reduction method) are developed.

4 Some remarks

In our description supra, we utilize the relation between PMC and BPPC in the same way as Dell'Amico *et al.* (2008) used the relation between $P||C_{\max}$ and BPP to obtain an exact algorithm for $P||C_{\max}$, but their overall algorithm for $P||C_{\max}$ was composed of a scatter search heuristic followed by an exact algorithm based on binary search and B&P. They were able to solve to optimality all the tested instances for the parallel machine scheduling problem, which was at least partially the merit of the scatter search component of the overall algorithm. This metaheuristic was able to solve a large number of instances within few seconds on average, and provided good approximations for the remaining instances, for which the B&P component subsequently found the optimal solution quickly, because the gap between upper and lower bound was small. In order to produce a fully functional and efficient algorithm, we still need to introduce heuristics (local-search-based or other): for the overall algorithm to perform well, it is essential that the difference between lower bound and upper bound be small before we start the binary search. For PMC, however, finding a feasible schedule without conflicts amounts to coloring the associated conflict graph with at most m colors, which is not an easy task. A possible avenue to pursue to this respect is to follow Gaur *et al.* (2009) in turning the incompatibility constraint into a "soft" constraint, allowing incompatible job pairs to be allocated to the same machine at the expense of a penalty. Verifying the benefits of this idea will require further research.

References

Berghman, L., Leus, R. and Spieksma, F.C.R., "Optimal solutions for a dock assignment problem with trailer transportation", *Annals of Operations Research*, to appear.

Bodlaender, H., Jansen, K., and Woeginger, G.J., 1994, "Scheduling with incompatible jobs", *Discrete Applied Mathematics*, Vol. 55, pp. 219-232.

Dell'Amico, M., Iori, M., Martello, S. and Monaci, M., 2008, "Heuristic and exact algorithms for the identical parallel machine scheduling problem", *INFORMS Journal on Computing*, Vol. 20, pp. 333-344.

Gaur, D.R., Krishnamurti, R. and Kohli, R., 2009, " Conflict resolution in the scheduling of television commercials", *Operations Research*, Vol. 57, pp. 1098-1105.

Martin, R.K., 1999, " Large scale linear and integer optimization: a unified approach", *Kluwer Academics Publishers*, Massachusetts.

Muritiba, A.E.F., Iori, M., Malaguti, E. and Toth, P., 2010, "Algorithms for the bin packing problem with conflicts", *INFORMS Journal on Computing*, Vol. 22, pp. 401-415.

ROADEF, 2011, "Google ROADEF/EURO challenge 2012: Machine reassignment", *Document available at* `http://challenge.roadef.org/2012/files/problem_definition_v1.pdf`.

Sadykov, R. and Vanderbeck, F., 2013, "Bin Packing with conflicts: a generic branch-and-price algorithm", *INFORMS Journal on Computing*, Vol. 25, pp. 244-255.

Yamada, T., Kataoka, S. and Watanabe, K., 2002," Heuristic and exact algorithms for the disjunctively constrained knapsack problem", *Information Processing Society of Japan Journal*, Vol. 43, pp. 2864-2870.

Time index-based models for RCPSP/max-cal

Stefan Kreter and Jürgen Zimmermann

Clausthal University of Technology, Germany
{stefan.kreter, juergen.zimmermann}@tu-clausthal.de

Keywords: Project scheduling, RCPSP/max, Calendars.

1 Introduction

In this paper we extend the RCPSP/max by the concept of calendars. This problem is denoted by RCPSP/max-cal and was first introduced by Franck (1999). Many practicians argue that calendars have to be taken into account when considering real-life project scheduling problems because resources like manpower or machines are not available in some time periods.

We assume that the project in question consists of n real activities $1, \ldots, n$ as well as two ficticious ones; 0 and $n+1$, corresponding to the start and completion of the project, respectively. With $p_i \in \mathbf{Z}_{\geq 0}$ we denote the processing time and with $S_i \in \mathbf{Z}_{\geq 0}$ ($C_i \in \mathbf{Z}_{\geq 0}$) the start time (completion time) of activity i where $p_0 = p_{n+1} = 0$. A project starts at time zero, i.e., $S_0 = 0$. Then, S_{n+1} equals the project duration (makespan). Between the start times of activities prescribed minimum time lags, $S_j - S_i \geq d_{ij}^{min}$, and maximum time lags, $S_j - S_i \leq d_{ij}^{max}$, are given. Activities and time lags are represented by an activity-on-node network $N = (V, A; \delta)$, where V represents the set of nodes (i.e., activities), A represents the set of arcs (i.e., time lags between activities) and δ describes the arc weights. Moreover, $\mathcal{R}$ identifies the set of renewable resources required for carrying out the activities and r_{ik} represents the amount of resource $k \in \mathcal{R}$ used during execution of activity i. For each resource $k \in \mathcal{R}$ a limited capacity $R_k \in \mathbf{Z}_{\geq 0}$ is given that must not be exceeded. The goal is to find a vector of start times $S = (S_0, S_1, \ldots, S_{n+1})$ (called a schedule) such that all described time lags and resource capacities as well as constraints resulting from calendars are satisfied and the project makespan is minimized.

In Section 2 we describe the concept of calendarization (Zhan 1992, Franck 1999), the resulting constraints, and depict a conceptional model formulation. Section 3 contains mixed-integer linear programming (MIP) formulations for RCPSP/max-cal. An experimental performance study and conclusions are given in Section 4.

2 The RCPSP/max-cal

Concerning calendarization we distinguish activities that may be interrupted and activities that cannot be interrupted due to technical reasons. The set of interruptable activities is denoted by $V^{bi} \subset V$ and the set of non-interruptable activities is given by $V^{ni} = V \setminus V^{bi}$, with $i \in V^{ni}$ if $p_i = 0$, i. e., the ficticious activities are in the set V^{ni}.

Definition 1. *A calendar is a function* $B : \mathbf{Z}_{\geq 0} \to \{0, 1\}$ *with:*

$$B(t) := \begin{cases} 1, & \text{if period } t+1 \text{ is a working period} \\ 0, & \text{if period } t+1 \text{ is a break.} \end{cases}$$

For each resource $k \in \mathcal{R}$ a calendar $B_k^{\mathcal{R}}$ that describes the time periods where k is not available (e.g., weekends or holidays) is given. Based on the resource calendars of a project one can derive calendars for each activity and time lag. With $\mathcal{R}_i := \{k \in \mathcal{R} \mid r_{ik} > 0\}$ we

denote the set of resources that is necessary to carry out activity $i \in V$. Then, an activity calendar $B_i : \mathbf{Z}_{\geq 0} \to \{0,1\}$ with

$$B_i(t) := \begin{cases} \min_{k \in \mathcal{R}_i} B_k^{\mathcal{R}}(t), & \text{if } \mathcal{R}_i \neq \emptyset \\ 1, & \text{else} \end{cases}$$

can be constructed for each activity $i \in V$. Activities $i \in V^{bi}$ can only be interrupted at a point in time t with $B_i(t) = 0$. Furthermore, the execution has immediately to be continued at the next point in time $t' > t$ with $B_i(t') = 1$. This convention is generally accepted in practice, in particular, if the processing of activities requires the setup of certain resources like machines or reactors. For non-interruptable activities, $i \in V^{ni}$, $C_i := S_i + p_i = S_i + \sum_{t=S_i}^{S_i+p_i-1} B_i(t)$ holds, i.e., a non-interruptable activity has to start at a point in time such that p_i working periods follow in the activity calendar. The completion time of the interruptable activities, $i \in V^{bi}$, depends on S_i and is given by $C_i := \min\{t \geq S_i + p_i | \sum_{\tau=S_i}^{t-1} B_i(\tau) = p_i\}$.

When regarding calendars we still concentrate on start-to-start time lags because arbitrary time lags can be converted into start-to-start ones by introducing dummy activities and dummy arcs (Franck 1999). For every time lag between activities $i, j \in V$ an arc calendar B_{ij} resulting from the activity calendars can be calculated.

The renewable resources are divided into two groups. We differentiate between resources that remain engaged during an interruption of activity $i \in V^{bi}$, like most machines, and resources that are released during an interruption, e.g., workers. Moreover, parameter $\rho_k \in \{0,1\}$, $k \in \mathcal{R}$, equals 1 if resource k stays engaged during an interruption of an activity i that requires k. If resource k is released during an interruption of activity i, it can be used to carry out other activities and ρ_k equals 0.

Given a schedule S and a point in time t the active set is the set of activities that are in process (Neumann *et al.* 2003). Just like in project scheduling without calendars the active set can be given by $\mathcal{A}(S,t) := \{i \in V | S_i \leq t < C_i\}$. The resource utilization $r_k^{cal}(S,t)$ of resource $k \in \mathcal{R}$ at time t according to schedule S can be computed by

$$r_k^{cal}(S,t) := \sum_{i: i \in \mathcal{A}(S,t) \wedge B_i(t)=1} r_{ik} + \sum_{i: i \in \mathcal{A}(S,t) \wedge B_i(t)=0} r_{ik}\, \rho_k.$$

Franck (1999) formulated the following conceptional model for the RCPSP/max-cal, where $\overline{d}^{cal}$ is an upper bound for the shortest project duration in calendarization:

$$\begin{aligned}
&\text{Minimize} && S_{n+1} && \\
&\text{subject to} && \sum_{t=S_i}^{S_j-1} B_{ij}(t) - \sum_{t=S_j}^{S_i-1} B_{ij}(t) \geq \delta_{ij} && \langle i,j \rangle \in A && (1)\\
& && \sum_{t=S_i}^{S_i+p_i-1} B_i(t) = p_i && i \in V^{ni} && (2)\\
& && B_i(S_i) = 1 && i \in V^{bi} && (3)\\
& && r_k^{cal}(S,t) \leq R_k && k \in \mathcal{R}, t \in \{0,1,\ldots,\overline{d}^{cal}\} && (4)\\
& && S_i \in \{0,1,\ldots,\overline{d}^{cal}\} && i \in V && (5)
\end{aligned}$$

Constraints (1) ensure the given temporal constraints. Equalities (2) force activities $i \in V^{ni}$ to start at a point in time with p_i working periods following, i.e., activities $i \in V^{ni}$ are carried out without interruption. Every activity $i \in V^{bi}$ starts at the beginning of a working period because of constraints (3). Inequalities (4) ensure that the given resource capacities R_k are not exceeded at any time and constraints (5) restrict the domain of the start times of activities.

3 MIP formulations for RCPSP/max-cal

To the best of our knowledge MIP formulations or exact solution procedures for the RCPSP/max-cal have not been discussed in the literature before. Based on the introduced conceptional model we will present different MIP formulations for the RCPSP/max-cal. The models are based on time-indexed binary decision variables x_{it} for each activity i and point in time t. In the first MIP formulation $x_{it} = 1$ if activity i starts at time t and $x_{it} = 0$ otherwise (Formulation-START). In the second formulation $x_{it} = 1$ if activity i starts at time t or earlier (Formulation-CHANGEOVER), and in the third formulation $x_{it} = 1$ if activity i is in progress at time t, where $x_{it} = 0$ if $B_i(t) = 0$ (Formulation-EXECUTION).

For reasons of space we focus on one of the three formulations in the following, namely Formulation-START. In our performance study all three formulations are investigated.

Procedures to determine the earliest (ES_i) and latest start time (LS_i) for each activity ($i \in V$) were developed by Franck (1999), Franck *et al.* (2001b), and Neumann *et al.* (2003). The set of all integer values from ES_i to LS_i is given by W_i and T represents the planning horizon. The absolute time that has to pass between the start times of avtivities i and j ($\langle i,j \rangle \in A$) if activity i starts at time t can be calculated with respect to the arc calendar B_{ij} and is denoted by d_{ijt}. Then, Formulation-START can be given as follows:

$$\text{Minimize} \quad \sum_{t \in W_{n+1}} t\, x_{n+1,t}$$

$$\text{subject to} \quad \sum_{t \in W_i} x_{it} = 1 \qquad i \in V \tag{6}$$

$$\sum_{t \in W_i} (t + d_{ijt})\, x_{it} \leq \sum_{t \in W_j} t\, x_{jt} \qquad \langle i,j \rangle \in A \tag{7}$$

$$\sum_{t \in W_i} x_{it} \sum_{\tau=t}^{\min(T,t+p_i-1)} B_i(\tau) = p_i \qquad i \in V^{ni} \tag{8}$$

$$\sum_{t \in W_i} x_{it}\, B_i(t) = 1 \qquad i \in V^{bi} \tag{9}$$

$$\sum_{i \in V} r_{ik} \sum_{\tau \in \mathcal{T}_{it}} x_{i\tau}\, B^{kit} \leq R_k \qquad k \in \mathcal{R}, t \in \{0,1,\ldots,T\} \tag{10}$$

$$x_{it} \in \{0,1\} \qquad i \in V, t \in W_i \tag{11}$$

The objective is to minimize the project duration. Constraints (7) ensure the given time lags. Equations (8) guarantee that activities from set V^{ni} are not interrupted during their execution. Due to constraints (6),(9), and (11) every activity starts at a feasible point in time where the corresponding activity calendar equals 1. Constraints (10) are the resource constraints where $\mathcal{T}_{it} := \left\{ \max\left\{ \tau \in \{0,1,\ldots,t-p_i\} \mid \sum_{z=\tau}^{t-1} B_i(z) = p_i \right\} + 1, \ldots, t \right\} \cap W_i$ contains all feasible start times of activity i for which i is in process at time t ($\max \emptyset := -1$). $B^{kit} := B_i(t) + (1 - B_i(t))\, \rho_k$ is always 1 if resource k stays engaged during an interruption of activity i. If resource k is released during an interruption of activity i, B^{kit} is 1 if the activity calendar of activity i at time t is 1 and 0 otherwise. Therefore the left side of constraints (10) displays the resource requirement on resource k at time t and due to these equations all resource constraints are fulfilled.

In order to reduce the number of decision variables and with this to improve the MIP formulations, we determine an upper bound $\overline{d}^{cal}$ on the project duration by adapting the serial schedule generation scheme with unscheduling step for the RCPSP/max (Franck 1999).

Concerning calendarization the length of a longest path (indirect temporal constraint) between a pair of activities $i, j \in V$ in the project network N depends on the start time of activity i. In order to create a matrix of longest path lengths between every pair of activities, we developed a method based on the Floyd-Warshall triple algorithm (Ahuja *et al.* 1993). With the help of this method a set of infeasible start times $\mathcal{I}_i$ can be calculated for each activity $i \in V$ and the MIP formulations START and CHANGEOVER can be further improved. The improved formulations have the suffix $\overline{d}^{cal}$ and $\mathcal{I}$, respectively.

4 Performance analysis & conclusion

The computational tests have been performed on instances that we adapted from the UBO test instances (Franck *et al.* 2001a). The planning horizon T was set to $T := \sum_{i \in V} \max\{p_i, \max_{\langle i,j \rangle}\{\delta_{ij}\}\}$, four different calendars were assigned to the resources, and the amount of interruptable activities was set to 60% and 80%, respectively. The MIP formulations were tested in the different versions basic, $\overline{d}^{cal}$, and $\mathcal{I}$. All tests were conducted on an Intel Core i7 CPU 990X with 3.47 GHz and 24 GB RAM under Windows 7 with CPLEX 12.5. The following table summarizes the results, where each block with always 90 instances is denoted by the number of activities and percentage of interruptable activities. Besides the average computation times in seconds, the percentage of activities for which an optimal solution was found and proven, for which infeasibility was proven, and for which neither optimality nor infeasibility was proven within six hours are given for the best version of each formulation.

Table 1. Average runtimes in seconds and percentage of instances solved

	START						CHANGEOVER						EXECUTION					
test set	basic	$\overline{d}^{cal}$	$\mathcal{I}$	opt	inf	h>6	basic	$\overline{d}^{cal}$	$\mathcal{I}$	opt	inf	h>6	basic	$\overline{d}^{cal}$	opt	inf	h>6
10-60	0.7	0.3	0.1	79	21	0	0.7	0.3	0.2	79	21	0	17.1	9.7	79	21	0
10-80	0.5	0.4	0.1	84	16	0	0.7	0.3	0.2	84	16	0	17.0	6.1	84	16	0
20-60	1316.5	841.9	560.7	75	25	0	69.4	50.6	35.1	75	25	0	4754.9	2193.8	75	19	6
20-80	992.1	788.8	636.5	78	21	1	52.2	41.0	35.1	78	22	0	4887.9	1696.0	78	18	4
50-60	/	/	8264.7	60	4	36	/	/	6226.2	69	8	23	/	/	/	/	/
50-80	/	/	7930.2	66	0	34	/	/	6210.3	72	5	23	/	/	/	/	/

As expected, the average runtimes decrease if fewer decision variables have to be taken into account. Therefore, the best versions of the three formulations are START-$\mathcal{I}$, CHANGEOVER-$\mathcal{I}$, and EXECUTION-$\overline{d}^{cal}$. As EXECUTION cannot stick with the other formulations the instances with 50 activities are tested exclusively with START-$\mathcal{I}$ and CHANGEOVER-$\mathcal{I}$. Summarizing, CHANGEOVER-$\mathcal{I}$ achieves the best results.

References

Ahuja R.K., T.L. Magnanti, and J.B. Orlin, 1993, "Network Flows", Prentice Hall, Englewood Cliffs.

Franck B., 1999, "Prioritätsregelverfahren für die ressourcenbeschränkte Projektplanung mit und ohne Kalender", Shaker, Aachen.

Franck B., K. Neumann, and C. Schwindt, 2001a, "Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling", *OR Spektrum*, Vol. 23, pp. 297–324.

Franck B., K. Neumann, and C. Schwindt, 2001b, "Project scheduling with calendars", *OR Spektrum*, Vol. 23, pp. 325–334.

Neumann K., C. Schwindt, and J. Zimmermann, 2003, "Project Scheduling with Time Windows and Scarce Resources", Springer, Berlin.

Zhan J., 1992, "Calendarization of time-planning in MPM networks", *ZOR - Methods and Models of Operations Research*, Vol. 36, pp. 423–438.

A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations

Patricio Lamas[1] and Erik Demeulemeester[1]

KU Leuven, Faculty of Economics and Business, Department of Decision Sciences and Information Management, Naamsestraat 69, B-3000 Leuven, Belgium
patricio.lamasvilches@kuleuven.be, erik.demeulemeester@kuleuven.be

Keywords: proactive RCPSP, chance-constrained programming, SAA.

1 Introduction

In reality projects are subject to high levels of uncertainty. This might lead to schedule disruptions that make the schedules obtained by solving the traditional resource-constrained project scheduling problem (RCPSP) completely different from the actually executed schedules. That fact triggered the incorporation of uncertainty in the models and methods for solving the RCPSP that were developed during the last decade.

The RCPSP is a complex problem to solve even when deterministic parameters are assumed (Blazewicz *et al.* 1983). Consequently, solving the RCPSP incorporating uncertainty is an extremely challenging problem. Therefore, the published methods for solving the RCPSP under uncertainty have been based on its division into two easier steps. One is the generation of an initial proactive robust baseline schedule, which is protected as much as possible against disruptions. The second step aims to define a reactive policy, which is deployed when a disruption occurs. The drawback of such a division is that it does not consider the dependency between the two steps, given that all the used robustness measures (solution stability, expected makespan and timely project completion probability) depend on both the proactive schedule and the reactive policy.

The contributions of this work are the following. First, we introduced a new robustness measure, defined as the probability that the actually executed policy is identical to the baseline schedule, which is independent of the reactive policy applied. Second, we propose a formulation for the deterministic RCPSP, which is easily extensible for the case of stochastic parameters. Finally, we developed a branch and cut algorithm to efficiently find a robust (purely) proactive baseline schedule, considering this new robustness measure.

2 Problem formulation

The most frequently considered approach in the literature in order to cope with uncertainty is to assume that the non-dummy activity durations are stochastic variables with a known probability distribution. Thus, we extend the traditional RCPSP as the problem of finding a non-preemptive schedule of minimal makespan subject to the probability that the actually executed policy is identical to the baseline schedule is larger than or equal to a predefined level.

We introduce a chance-constrained programming formulation for the RCPSP (C-C RCPSP), based on the disjunctive graph $G(V, E)$ introduced in Balas (1969). Therefore, we will assume an activity-on-the-node (AoN) representation of the project. Each node $i \in V$ represents an activity and each edge $(i, j) \in E$ represents a precedence relation. Also, we will use the concept of a minimal forbidden set that was introduced in Igelmund

and Radermacher (1983). A forbidden set is a set of activities (without precedence relations between them) that cannot be in progress simultaneously because of resource limitations due to some resource type. A minimal forbidden set is a forbidden set such that each of its subsets is not a forbidden set. Let Φ be the set of all minimal forbidden sets.

Binary variable x_{ij} is equal to 1 if edge (i, j) is selected and 0 otherwise. Positive integer variable S_i represents the starting time of activity $i \in V$.

$$\min S_{n+1} \tag{1}$$

subject to

$$x_{ij} = 1 \qquad \forall (i,j) \in E \tag{2}$$

$$x_{ij} + x_{ji} \leq 1 \qquad \forall (i,j) \in V \times V, i \neq j \tag{3}$$

$$\sum_{(i,j) \in F \times F, i \neq j} x_{ij} \geq 1 \qquad \forall F \in \Phi \tag{4}$$

$$P\begin{pmatrix} S_j - S_i \geq M\,(x_{ij} - 1) + d_i \\ \forall\,(i,j) \in V \times V, i \neq j \end{pmatrix} \geq 1 - \alpha \tag{5}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in V \times V, i \neq j \tag{6}$$

$$S_i \in \mathbb{Z}^+ \qquad \forall i \in V \tag{7}$$

Objective function (1) represents the starting time of (dummy) activity $n+1$ which is equivalent to the project makespan. The set of constraints (2) states that for each original precedence relation in E there is a selected edge in the graph. Constraints (3) ensure that if i preceeds j, then j does not preceed i (and vice versa). Constraints (4) state that for each minimal forbidden set F there is at least one selected edge. Constraint (5) ensures that the probability that all the precedence relations hold is larger than or equal to the confidence level $1 - \alpha$ (with M any upper bound on the project makespan). Constraints (6) and (7) are the integrality constraints. Given the assumption that the vector of activity durations d is integer, the integrality constraints (7) can be relaxed.

The above model is based on a formulation for the RCPSP that is slightly different from the one introduced in Alvarez-Valdés and Tamarit (1993). In the latter, the edges represent either direct or transitive precedence relations, whereas in the formulation (1)-(7) the edges represent only direct precedence relations. An advantage of our formulation is that it does not require the introduction of the transitivity constraints.

3 Solution method

There are three main difficulties that make the C-C RCPSP model impractical to solve by directly using the mixed integer programming methods: the feasible region defined by the probabilistic constraint is not convex (Luedtke and Ahmed 2008), the probabilistic constraint is hard to compute (Luedtke and Ahmed 2008) and the number of minimal forbidden sets is exponential in the number of activities. The first and second difficulties are tackled by a sample average approximation. The last difficulty is approached by a branch-and-cut algorithm.

We use the sample average approximation (SAA) that is presented in Luedtke *et al.* (2010). In such a method, the original distribution of the random parameters, in our problem the activity durations $d \in \mathbb{N}_0^{n+2}$, is replaced by an empirical distribution obtained from a random sample. Under some conditions, a feasible solution of the SAA problem will be feasible in the original C-C programming problem with a high probability. The only

assumption made on the distribution of the random parameters is that it can be sampled from.

A sample W is a set of realizations of the random activity durations vector, such that $W = \left\{d^1, \ldots, d^{|W|}\right\}$. Let a scenario $w \in \{1, \ldots, |W|\}$ be a realization $d^w \in W$.

The basic idea of the reformulation introduced in Ruszczynski (2002) is to solve a problem that is infeasible for at most $\lfloor |W| \cdot \epsilon \rfloor$ realizations, thus the solution will be feasible with a confidence level (for the SAA problem) of at least $(1-\epsilon)$, with $(1-\epsilon) > (1-\alpha)$. Based on that idea, we reformulate C-C RCPSP as follows:

Let y_w be a binary variable that takes the value 1 if the obtained solution is not necessarily feasible for scenario w and 0 otherwise.

Probabilistic constraint (5) can then be replaced by the following constraints:

$$S_j - S_i \geq M\left(x_{ij} - 1\right) + \left(1 - y_w\right) d_{iw} \qquad \forall (i,j) \in V \times V, i \neq j, w \in W \tag{8}$$

$$\sum_{w \in W} y_w \leq \lfloor |W| \cdot \epsilon \rfloor \tag{9}$$

$$y_w \in \{0,1\} \qquad \forall w \in W \tag{10}$$

Constraints (8) state that each precedence relation holds if $y_w = 0$. Constraint (9) imposes that the maximum number of scenarios for which the solution is not necessarily feasible is $\lfloor |W| \cdot \epsilon \rfloor$. Thus, the SAA formulation for the original C-C RCPSP (SAA RCPSP) is: minimize (1) subject to: (2)-(4), (6)-(10).

The lower bound obtained by solving the LP relaxation of the SAA RCPSP is weak in general. We include a set of strong valid inequalities for general chance constrained programming problems that were introduced in Luedtke *et al.* (2010). Also we propose a heuristic method for the generation of good feasible solutions and a lower bound for the SAA problem.

Given that the number of minimal forbidden sets is exponential in the number of activities, we implement a delayed constraint generation method for including constraints (4) in SAA RCPSP only when they are violated. We have proven that the separation problem for these constraints is NP-hard. However, if we consider (non-necessarily minimal) forbidden sets, the constraints (4) can be separated in polynomial time. We also propose a heuristic separation method for generating only one violated forbidden set constraint per iteration.

We embedded the constraint generation methods in the branch-and-bound tree, adding the violated constraints as cuts. The combination of both methods generates a branch-and-cut algorithm.

4 Results and conclusions

We tested the presented procedures on the 480 instances composed by 30 non-dummy activities and 4 types of resources belonging to the PSPLIB library (Kolisch and Sprecher 1997). Since those instances were created for the deterministic RCPSP, we modified them in the following way: the activity durations d_i for each non-dummy activity i follow a discretized beta distribution with shape parameters 2 and 5 and an expected value equal to the duration in the original RCPSP instances. Also, we will consider three different levels of variability, which are obtained by changing the upper and lower limits of the distributions: low variability $[0.75 \cdot E(d_i), 1.625 \cdot E(d_i)]$, medium variability $[0.5 \cdot E(d_i), 2.25 \cdot E(d_i)]$ and high variability $[0.25 \cdot E(d_i), 2.875 \cdot E(d_i)]$.

The branch-and-cut algorithm was implemented in C++ using the CPLEX 12.5 API and ran on a personal computer equipped with an Intel® Core™ i7-2720QM 2.20 GHz

and 4 Gb RAM. All the remaining algorithms presented in this paper were implemented in C++ and ran on the same computer as well.

Our algorithm took an average time of approximately 1 second for optimally (or approximately) solving the instances. However, as soon as the confidence level decreases, the performance of the algorithm drastically decreases. It was possible to obtain feasible solutions for all the instances with confidence levels larger than 70% in computation times smaller than 10 seconds. As expected, our algorithm in general outperformed two alternative methods published in the literature (Bruni *et al.* 2011, Van de Vonder *et al.* 2008) considering this new robustness measure. Even more, it tended to outperform the other methods considering the traditional measures.

Finally, we found two contributions from an algorithmic point of view. First, our branch-and-cut algorithm can be applied to solve the traditional deterministic RCPSP. We can predict a good performance, considering the relatively good computation times for the RCPSP with stochastic activity durations. Also, the presented methods for obtaining lower and upper bounds are quite flexible, therefore they could be applied for general mixed integer chance-constrained programming problems.

References

Alvarez-Valdés, R. and Tamarit, J., 1993, "The project scheduling polyhedron: Dimension, facets and lifting theorems", *European Journal of Operational Research*, Vol. 67, pp. 204-220.

Balas, E., 1969, "Machine sequencing via disjunctive graphs: an implicit enumeration algorithm", *Operations research*, Vol. 17, pp. 941-957.

Blazewicz, J., Lenstra, J.K. and Kan, A.H.G. 1983, "Scheduling subject to resource constraints: classification and complexity", *Discrete Applied Mathematics*, Vol. 5, pp. 11-24.

Bruni, M., Beraldi, P., Guerriero, F. and Pinto, E. 2011, "A heuristic approach for resource constrained project scheduling with uncertain activity durations", *Computers & Operations Research*, Vol. 38, pp. 1305-1318.

Igelmund, G. and Radermacher, J. 1983, "Preselective strategies for the optimization of stochastic project networks under resource constraints", *Networks*, Vol. 13, pp. 1-28.

Kolisch, R. and Sprecher, A., 1997, "PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program", *European Journal of Operational Research*, Vol. 96, pp. 205-216.

Luedtke, J. and Ahmed, S., 2008, "A sample approximation approach for optimization with probabilistic constraints", *SIAM Journal on Optimization*, Vol. 19, pp. 674-699.

Luedtke, J., Ahmed, S. and Nemhauser, G., 2010, "An integer programming approach for linear programs with probabilistic constraints", *Mathematical Programming*, Vol. 122, pp. 247-272.

Ruszczynski, A., 2002, "Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra", *Mathematical Programming*, Vol. 93, pp. 195-215.

Van de Vonder, S., Demeulemeester, E. and Herroelen, W., 2008, "Proactive heuristic procedures for robust project scheduling: An experimental analysis", *European Journal of Operational Research*, Vol. 189, pp. 723-733.

Threshold Policy for Call Centers with a Callback Option

Benjamin Legros[1], Oualid Jouini[1] and Ger Koole[2]

[1] Ecole Centrale Paris, France

benjamin.legros, oualid.jouini@ecp.fr

[2] VU University Amsterdam, The Netherlands

ger.koole@vu.nl

Keywords: Call center, optimization, dynamic programming, threshold, blending.

1 Introduction

In the context of highly congested call centers, the use of an alternative service channel can be proposed to customers so as to balance workload and avoid excessive abandonments. For example, the call back option can increase performance measures (see Armony and Maglaras (2004a,b)).

In this project, we consider a call center modeling with a single customer type and a callback option. We develop a method based on Markov chains to evaluate its performance measures. We also provide an efficient routing policy based on the number of reserved agents for the inbound calls and the number of outbound calls.

First, we present the modeling of a call center with a callback option. Second, we evaluate the performance measures. Third, we numerically study the general form of the optimal curve relating the threshold and the number of waiting outbound calls.

2 Modeling

The arrival process of inbound calls is assumed to be a homogeneous Poisson process with rate λ. There are s identical agents who can handle both types of jobs. We assume that the service times of inbound or outbound calls are exponentially distributed with the same rate μ.

The callback option is proposed to a newly arrival only when her expected waiting time is too long (i.e. too many waiting calls in queue 1). We define a limit k ($k \in \mathbb{N}$) in the number of waiting calls in queue 1. Upon her arrival, a customer can find three situations. If at least one agent is available then she is routed to one of the idle agents. Otherwise if the number of waiting calls in queue 1 is strictly lower than k, the callback option is not proposed and she waits in queue 1. If this number is higher than or equal to k the callback option is proposed. The customer reaction is assumed to be probabilistic. Directly upon arrival, she decides to accept the callback with a probability q and goes to queue 2 or she decides to wait in queue 1 with a probability $1-q$ ($q \in [0,1]$).

We consider a threshold policy inspired by Bhulai and Koole (2003) and Legros et al. (2013). The inbound calls have a non preemptive priority over the outbound ones. Let us denote the threshold by c, $1 \leq c \leq s$. When an agent becomes idle, she handles the call in queue 1, if any. If not, the agent may either handle a call in of queue 2 if any, or she remains idle. If the number of idle agents (excluding her) is at least $s-c$, then the agent in question handles an outbound call (from queue 2). Otherwise, she remains idle. In other words, there are $s-c$ agents that are reserved for inbound calls.

In this project, we propose a threshold policy which adjusts the threshold as a function of the number of outbound calls in queue 2, denoted by y ($y \in \mathbb{N}$). We want to minimize the expected waiting time of the outbound calls, denoted by $E(W_2)$ and respect a constraint of service level, denoted by w_1^* on the expected waiting time of the inbound calls, denoted by $E(W_1)$.

3 Performance Measures in the case $c(y) = c$

In Theorem 1, we give the expression of the expected waiting time for inbound calls, $E(W_1)$, and the proportion of customers who asks for a callback, π. We also provide a numerical method to compute $E(W_2)$.

Theorem 1. *For $1 \leq c \leq s$, we have*

$$\pi = p_{0,0} \frac{q\left(\frac{a}{s}\right)^k \frac{a^{s-c}c!}{s!}}{1-\frac{a(1-q)}{s}} \frac{\frac{a^c}{c!}}{1-q\frac{a}{c}\frac{a^{s-c}c!}{s!}\frac{a^k}{s^k}\frac{1}{1-\frac{a(1-q)}{s}}}, \tag{1}$$

$$E(W_1) = \frac{p_{0,0}}{\lambda} \frac{a^{s-c} c!}{s!} \left(\sum_{x=0}^{k-1} x \frac{a^x}{s^x} + \left(\frac{a}{s}\right)^k \left(\frac{k\left(1 - \frac{a(1-q)}{s}\right) + \frac{a(1-q)}{s}}{\left(1 - \frac{a(1-q)}{s}\right)^2} \right) \right) \frac{\frac{a^c}{c!}}{1 - q\frac{a}{c}\frac{a^{s-c}c!}{s!}\frac{a^k}{s^k}\frac{1}{1-\frac{a(1-q)}{s}}}. \tag{2}$$

4 Optimal Curve via Dynamic Programming

In this section we propose to find the optimal curve via dynamic programming. We denote by c_1 and c_2 the costs of a waiting customer in queue 1 and in queue 2, respectively. Note that we assume $c_2 < c_1$. We denote by $V_n(x, y)$ the expected costs over n steps. We have

$$\begin{aligned} V'_{n+1}(x,y) &= c_1(x-s)^+ + c_2 y \\ &+ \frac{\lambda}{\lambda + s\mu} \left(1_{x<k+s} V_n(x+1,y) + 1_{x \geq k+s}(qV_n(x,y+1) + (1-q)V_n(x+1,y)\right) \\ &+ \frac{\mu}{\lambda + s\mu} \left(\min(x,s)V_n(x-1,y) + (s - \min(x,s))V_n(x,y)\right), \end{aligned} \tag{3}$$

and

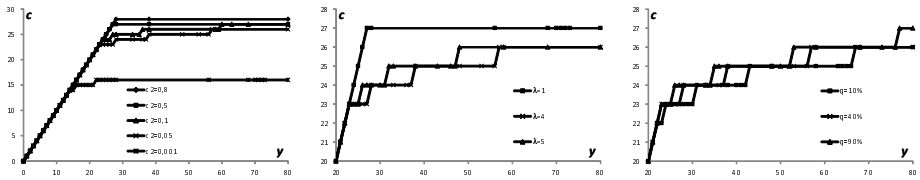
$$V_{n+1}(x,y) = \begin{cases} V'_{n+1}(x,y) \text{ if } x \geq s \\ \min(V'_{n+1}(x,y), V'_{n+1}(x+1,y-1)) \text{ if } x < s \end{cases} \tag{4}$$

The long-term average optimal actions are a solution of the optimality equation $TV = g + V$. Another way of obtaining them is through value iteration, by recursively defining $V_{n+1} = TV_n$, for arbitrary V_0. In the expression it is optimal to schedule an outbound call only if $V(x+1, y-1) < V(x, y)$. With this relation we can numerically build the optimal curve $y(c)$.

5 Numerical Analysis

In Figure 1 we present various examples of optimal curves corresponding to different values of the parameters c_2, λ and q. Note that the impact of k would be similar to the one of q. The optimal curves $c(y)$ are increasing and concave in y. Figure 1(a) informs that the threshold decreases as c_2 decreases. The reason is the relative importance to inbound calls compared to outbound ones. When this difference is important there is an opportunity to reserve agents for inbound calls. The opposite is true when this difference is not significant (see example with $c_2 = 0.8$). Figure 1(b) reveals that the threshold is

not a monotonic function of the workload. For light workload (see example with $\lambda = 1$) because most of the calls do not wait, the reservation for inbound calls does not need to be important. Recall that the reservation always deteriorates the overall expected waiting time. For higher workload, when the expected waiting time is significant for inbound and outbound calls, we have shown with the evaluation of the performance measures that the proportion of outbound calls increases as a function of the workload. Thus the reservation for inbound calls has to reduce since the inbound calls are relatively less numerous in the call center. This explains the position of the curves in Figure 1(b). Figure 1(c) confirms the impact of q. As q increases, the proportion of outbound calls increases and the overall cost of outbound calls in queue 2 also increases. Thus the reservation for inbound calls should decreases and the threshold increases.



(a) Impact of c_2 ($\lambda = 4$, $q =$ 40%)

(b) Impact of λ ($c_2 = 0.05$, $q = 40\%$)

(c) Impact of q ($c_2 = 0.05$, $\lambda = 4$)

Fig. 1. Optimal curve($\mu = 0.2$, $s = 28$, $c_1 = 1$, $k = 5$)

6 Conclusion

The optimization problem is answered through the building of an optimal curve $c(y)$. The general equation of this curve is $c(y) = c_{n^*}(y) + \mathbb{I}_{y \geq y_1} + \mathbb{I}_{y \geq y_2} + \cdots + \mathbb{I}_{y \geq y_r}$, with $1 \leq n^* < y_1 < y_2 < \cdots < y_r$ and $1 \leq r \leq s - n^*$. For future research, it would be interesting to evaluate via closed form expressions, the performance measures on the optimal curve.

Bibliography

Armony, M. and Maglaras, C. (2004a). Contact Centers with a Call-Back Option and Real-Time Delay Information. *Operations Research*, 52:527–545.

Armony, M. and Maglaras, C. (2004b). On Customer Contact Centers with a Call-Back Option: Customer Decisions, Routing Rules and System Design. *Operations Research*, 52(2):271–292.

Bhulai, S. and Koole, G. (2003). A Queueing Model for Call Blending in Call Centers. *IEEE Transactions on Automatic Control*, 48:1434–1438.

Legros, B., Jouini, O., and Koole, G. (2013). Adaptative Call Center Blending. Working paper. Ecole Centrale Paris.

Payment models and net present value optimisation for project scheduling

Pieter Leyman[1] and Mario Vanhoucke[1,2,3]

[1] Ghent University, Belgium
[2] Vlerick Business School, Belgium
[3] University College London, UK
pieter.leyman@ugent.be, mario.vanhoucke@ugent.be

Keywords: Net present value, payment models, project scheduling.

1 Introduction

Project scheduling, and in specific resource scheduling, has been extensively discussed in academic literature. Based on the overview of Hartmann and Briskorn (2010) it can be seen that recently relatively little research has been done regarding net present value (NPV) maximisation. We hope to remedy this by doing research on project scheduling, net present value and payment models. The latter determine when and how much cash is received and paid, and have a direct influence on the project NPV. Furthermore, the payment models in literature typically assume payment times which are fixed in advance, whereas e.g. Dayanand and Padman (2001) determine the payment times as part of their proposed solution method. We aim to find a middle ground between on the one hand the payment models with fixed payment times and the method proposed by Dayanand and Padman (2001) on the other hand, by fine–tuning our algorithm on a selected payment model and not simply take the payment structure as given.

In this abstract, a brief overview is given of the current focus of the research with respect to payment models and NPV. As such, section 2 discusses the concrete problem under consideration. In section 3 the crucial parts of the proposed algorithm are highlighted, whereas section 4 displays some preliminary results. Finally, section 5 offers both a brief summary and ideas for future research.

2 Problem description

The project scheduling problem which we discuss is the multi–mode resource–constrained project scheduling problem with discounted cash flows (MMRCPSPDC). This problem extends the traditional resource–constrained project scheduling problem (RCPSP) by adding multiple execution modes for each activity, and by including cash flows. This means any method which aims to solve the MMRCPSPDC will have to make an additional trade–off, i.e. between the different modes of each activity. Furthermore, the problem's objective is no longer to minimise project duration, but rather to maximise the project NPV. Hence, both extensions greatly increase the problem's complexity. For a mathematical model formulation of the MMRCPSPDC, we refer to Mika *et. al.* (2005).

In the presentation we will discuss the MMRCPSPDC and make use of the payment at activities' completion times (PAC) payment model, others may be discussed as extensions. We analyse the model from the point of view of the contractor and assume cash outflows occur once an activity has been completed, since this is often the case when we work with subcontractors and these are paid once an activity has been completed. For the PAC model cash inflows also occur at the end of each activity. As such, both cash flows can be summed up to yield a single net cash flow, which will be either positive or negative. This way, each

activity will contribute either positively or negatively to the project NPV, and the former should be scheduled as early as possible, whereas the latter as late as possible.

3 Methods

Although the MMRCPSPDC has already been discussed in several papers (see e.g. Ulusoy *et. al.* (2001) and Mika *et. al.* (2005)), none of these algorithms are fine–tuned on a specific payment model. We propose a genetic algorithm (GA), where for each payment model a set of scheduling methods and genetic operators is chosen, which lead to the best results for that payment model. Since the majority of our current research is on the PAC model, we will only discuss our proposed algorithm for this model.

For preprocessing we employ the same three steps as used by Hartmann (2001) in order eliminate unfeasible and inefficient modes, and redundant non–renewable resources. We start from a population consisting of both a set of elements with random mode lists, and one with mode lists determined based on the best performing mode selection rules. Doing so results in a quite diverse initial population.

Next, we have tested several schedule generation schemes, and our results are in line with Vanhoucke (2010), who conclude that a bi–directional generation scheme combined with a multi–pass forward–backward improvement method performs best. For testing, we have employed the j10, j20 and j30 datasets from PSPLIB. Since these datasets do not contain any data on activity cash flows, we have generated our own cash flow data. As stated above (see section 2.2) for the PAC payment model a single net cash flow suffices since both cash in– and outflows occur at activity completion times. Therefore, we have decided to generate cash flows with a predetermined percentage of negative cash flows, which varies throughout the datasets from 0 to 100% in steps of 20%. The results are shown in table 1. Note that the table illustrates some of the alternatives which we tested, but definitely not all of them.

The first column shows the datasets, whereas the next columns display the results for the GA with the proposed scheduling method, with only a serial schedule generation scheme (SSGS), and with a SSGS extended with a backward improvement. It is clear that the GA–Bi option outperforms both others for all three datasets, based on an ANOVA test. Only the difference between GA–Bi and GA–SS+ for the j10 dataset is not significant at the 5% significance level.

Table 1. Comparison of scheduling methods

Dataset	**GA–Bi**	**GA–SS**	**GA–SS+**
j10	2330.59	2318.34	2326.75
j20	4195.02	4159.72	4168.19
j30	5871.30	5797.44	5808.87

In terms of a selection operator, tournament selection performed best, whereas the one–point crossover outperformed all other crossover operators. Finally, for mutation we apply a swap operator which not only swaps modes in the mode list, but also activities in the priority list. Furthermore, the mutation percentage changes dynamically as the algorithm progresses.

4 Results

We have compared our GA with a memetic algorithm (MA) proposed by Chen and Chyu (2008). The results are shown in table 2. The first column displays the tested dataset, whereas columns 2 and 3 show the average NPV for both algorithms. Column 4 gives the percentage improvement of the GA compared to the MA, and the final column displays the p–value of the corresponding paired–samples T–test.

Table 2. Comparison between GA and MA

Dataset	GA	MA	GA vs MA	p–value
j10	2330.59	2239.87	+3.87%	0.000
j20	4195.02	3887.28	+7.92%	0.000
j30	5871.30	5335.07	+10.05%	0.000

Based on the results in table 2 it should be clear that our proposed GA outperforms the MA for all three datasets. Furthermore, the larger the number of activities in a project, the better the GA performs in comparison with the MA.

5 Conclusions and future research

In this abstract we have discussed payment models for project scheduling problems, concretely for the MMRCPSPDC. Next, we have given an overview of our GA and compared it with an existing algorithm from literature. Finally, we have shown that our proposed algorithm outperforms the MA of Chen and Chyu (2008).

In the future, we aim not only to compare our GA with more algorithms from literature, but also to extend it so it can be efficiently applied to other payment models.

References

Chen A.H.L., Chyu C.C., 2008, "A Memetic Algorithm for Maximizing Net Present Value in Resource–Constrained Project Scheduling Problem", *IEEE Congres on Evolutionary Computation*, pp. 2401-2408.

Dayanand N. and Padman R., 2001, "Project Contracts and Payment Schedules: The Client's Problem", *MS*, Vol. 47(12), pp. 1654-1667.

Hartmann S., 2001, "Project Scheduling with Multiple Modes: A Genetic Algorithm", *AOR*, Vol. 102, pp. 111-135.

Hartmann S. and Briskorn D., 2010, "A Survey of Variants and Extension of the Resource–Constrained Project Scheduling Problem", *EJOR*, Vol. 207, pp. 1-14.

Mika M., Waligóra G. and Weglarz J., 2005, "Simulated Annealing and Tabu Search for Multi-Mode Resource-Constrained Project Scheduling with Positive Discounted Cash Flows and Different Payment Models", *EJOR*, Vol. 164, pp. 639-668.

Ulusoy G., Sivrikaya-Şerifoğlu F. and Şahin S., 2001, "Four Payment Models for the Multi-Mode Resource Constrained Project Scheduling Problem with Discounted Cash Flows", *AOR*, Vol. 102, pp. 237-261.

Vanhoucke M., 2010, "A Scatter Search Procedure for Maximizing the Net Present Value of a Resource-Constrained Project with Fixed Activity Cash Flow", *IJPR*, Vol. 48(7), pp. 1983-2001.

A Metaheuristic Approach to Metascheduling of Workflow Applications in Computational Grids

Mika M[1]

[1]Poznan University of Technology, Poznań, Poland
e-mail: Marek.Mika@cs.pur.poznan.pl

Keywords: grid scheduling, resource allocation, network resources, metaheuristics.

1. Introduction

Computational grid is a computing environment dedicated to execute applications with large computational requirements. These applications are mainly scientific applications developed for using in many areas, such as astronomy, bioinformatics, biology, climatology, high-energy physics, meteorology, oceanography, seismology and others. Many of them are composed of multiple simpler components (tasks) that process large data sets, execute scientific simulations, communicate and interact with each other over the course of the application in order to share data and pass the control. The tasks are very often precedence-related, and the precedence constraints usually follow from the transmission of data files (i.e., output of one task becomes an input for the next task) and/or the control flow between two computational tasks. Such complex applications consisting of various precedence-related transformations (tasks) performed on certain data between which data files have to be transmitted very often are called workflow applications. In general, two types of workflows can be distinguished: data-intensive, where transmitted files are very large and therefore file transfer times are comparable or greater than the times of computational tasks, and compute-intensive, for which file transmission tasks do not occur or transferred files are so small that the transfer times can be neglected. Workflow applications are usually very time-consuming (even if single tasks are short) and input/output data files for tasks can be large. Execution time of a single workflow application usually ranges from several hours to several days, although it may as well be much larger. For efficient execution of both types of workflows, high computing power is required, due to a large amount of computations and data involved. This computational power can be provided by a computational grid, because the tasks of workflow applications have one interesting feature – they can be executed asynchronously.

There are at least several approaches to grid resource allocation. They differ among themselves depending on the grid architecture, purpose of a particular grid, and grid management policies. Depending on the architecture, two types of grids can be distinguished: peer-to-peer and centralized grids. In the peer-to-peer grid all services are equal and communicate using a peer-to-peer model of the network. In the centralized grid a grid resource management system plays a central role and is surrounded by many other grid services structured in a layered architecture. In such grids, there is usually one common, central grid broker or grid resource manager that serves all users and their jobs. Such a situation is considered in this work.

The modern grid resource management involves possibly several layers of schedulers. At the highest level are metaschedulers (or grid-level schedulers), which usually have a more general view of the resources but it does not own any resource of the grid where the application will eventually run. At the lowest level is a local resource management system that manages a specific resource or a set of resources, but it does not know too much about other resources of the grid. In this paper we consider the metascheduler level only as a mechanism which should be superior to others.

2. Problem formulation

In this paper we use the model which was presented by Mika et al. (2011) where the following assumptions were made concerning both the workflow application and the computational grid.

The workflow application consists of two sets V and E of interrelated tasks. These sets contain computational and data transmission tasks, respectively. The structure of a workflow is represented by a directed acyclic graph $W = (V, E)$, where each vertex corresponds to a computational task, and each arc represents a precedence relation between two computational

tasks. Each arc $(v_i, v_j) \in E$ corresponds to a transmission task, i.e., represents a data transmission between two successive computational tasks $v_i \in V$ and $v_j \in V$. Computational tasks are nonpreemptable, i.e., once started they have to be completed with no interruptions and no changes in resource allocation. Each computational task $v_i \in V$ may be executed in exactly one node, and is characterized by three values: its size p_i, i.e., the execution time on a standard processor (processors), the number r_i of processors required for its execution, and the minimal speed factor ω_i of the processors required. The actual processing time of a task $v_i \in V$ executed on processors with speed factor ϖ_κ (such that $\varpi_\kappa \geq \omega_i$) is calculated using function $f_i(p_i, \varpi_\kappa)$. We assume that f_i is calculated by dividing the task size by the speed factor of the processor (processors) on which this task is scheduled, i.e., $f_i = f = p_i / \varpi_\kappa$. Transmission tasks $(v_i, v_j) \in E$ are also nonpreemptable, and they are characterized by two values: the size F^{ij} of the data file (files) to be transmitted, and the required bandwidth B^{ij} between the two nodes between which the transmission is to be performed. The transmission time (i.e. the execution time of a transmission task) is calculated as $g(F^{ij}, B^{ij}) = F^{ij} / B^{ij}$ can take one of the following two values: the data file size divided by the bandwidth, when successive computational tasks v_i and v_j are executed in different resource nodes, or zero, when they are executed in the very same resource node.

A grid is a set of network nodes connected by fast network links. Its topology is represented by an undirected multigraph $\Gamma = (\mathrm{N}, \Psi)$, where N is the set of nodes and Ψ is the set of network links. There are two types of nodes in the network: resource nodes (containing computational resources) represented by set X and non-resource nodes (considered only with respect to the network topology) represented by set Π. Of course, $\mathrm{X} \cup \Pi = \mathrm{N}$. Between two given network nodes $\mu \in \mathrm{N}$ and $\nu \in \mathrm{N}$ there can be more than one network link $(\mu, \nu)_\psi: \mu, \nu \in \mathrm{N}, \psi = 1,2, \ldots, \Psi_{\mu\nu}$, and these links may have different parameters. Bandwidth $\Psi_\psi^{\mu\nu}$ of each network link between each two connected nodes (i.e. links $(\mu, \nu)_\psi$) is given, and it is identical in both directions. However, these are alternative links and they cannot be merged in order to increase bandwidth. Bandwidth within a given node is unlimited. Processors in each resource node X_χ, $\chi = 1,2, \ldots, |\mathrm{X}|$ are divided into types, depending on their power. The power (processing speed) of each processor is given by a function of some standard unit. The form of this function is identical for all processors of a given type. We assume a linear form of the processing speed function, as "a speed factor multiplied by the standard unit", where speed factor is equal to ϖ_κ. A processor with a speed factor equal to 1 we call a standard processor. The number of processors of type κ available in resource node X_χ is equal to $\mathrm{P}_{\kappa\chi}$.

3. Resource allocation

There are several types of computational applications, which can be executed in the computational grid environment. Some of them consists of many short computational tasks which often communicate among themselves sending small volumes of data. But there exist other types of applications, commonly known as workflow applications, which consist of a set of precedence-related and very time consuming computational tasks performed on some large data files.

It is not justified to apply procedures of metascheduler for the first case, because the time necessary to obtain a good schedule may be longer than the time by which the schedule length will be improved, but it is strongly recommended in the case of workflow applications, because the time necessary to find a good resource allocation is very short in comparison to the execution time of the workflow application.

In this paper we propose a metascheduling approach consisting of two phases. In the first phase a set of feasible resource allocation for the workflow is found. In the second phase, we try to find the one resource allocation for which the obtained schedule optimizes the chosen optimization criterion. In this work we consider the makespan as a scheduling criterion, but some other measures like resource levelling or financial objectives including energy saving.

The first phase of the presented approach is used to find a feasible resource allocation, which is defined as a process of assigning computational tasks to resource nodes, and assigning transmission tasks to connections of required bandwidths. For this the algorithm RA-TT proposed by Mika et al. (2011) is used to find feasible resource allocations for so called tri-tasks. A tri-task $\langle v_i, v_j \rangle$ is defined as a triple $(v_i, (v_i, v_j), v_j)$ consisting of two consecutive computational tasks v_i and v_j as well as a transmission task (v_i, v_j) between them. A feasible resource allocation RA^{ij} for

tri-task $\langle v_i, v_j \rangle$ is a pair of resource nodes (X_χ, X_θ) such that: (i) each computational task is assigned to a resource node which is capable of executing this task, (ii) a transmission task is performed over the path such that bandwidths of all network links of this path are greater than or equal to the required bandwidth of the transmission task. Algorithm RA-TT is executed for each tri-task $\langle v_i, v_j \rangle$ in the following way. Firstly, it removes from the graph representing the structure of the grid all connections for which transmission requirements are not fulfilled. Next, it finds in the reduced graph, all maximal connected subgraphs (so called subgrids). For each subgrid it finds two sets of resource nodes: a set X_χ of locations where task v_i can be executed and a set X_θ where task v_j can be executed. As a result a set A^{ij} that is the set of all feasible resource allocations RA^{ij} for tri-task $\langle v_i, v_j \rangle$ of a workflow W is obtained. This set contains pairs of locations (X_χ, X_θ), where X_χ is a location where task v_i can be performed, X_θ is a location where task v_j can be performed, and there exists a path between these locations with bandwidth at least equal to the required bandwidth of a transmission task (v_i, v_j).

After the execution of the algorithm RA-TT for each tri-task occurring in the workflow application we obtain a set of all feasible resource allocations for all tri-tasks. Unfortunately, some of these resource allocations, i.e., pairs of locations (X_χ, X_θ) are infeasible, if task dependencies are taken into account. There are three types of task dependencies. Two tri-tasks $\langle v_i, v_j \rangle$ and $\langle v_k, v_m \rangle$ are dependent tri-tasks, if one of the following cases occur: 1) $i = k$, 2) $j = k$, or 3) $j = m$. Resources have to be allocated to dependent tri-tasks in such a way that the computational task occurring in both dependent tri-tasks is assigned to the same resource node of the grid Γ. Thus, another algorithm is required to remove all such infeasible resource allocations resulting from tri-task dependencies. Mika et al. (2011) proposed for this purpose another algorithm called RA-W, which results in finding one feasible resource allocation. In this work an alternative approach is proposed. We use a part of the algorithm RA-W to remove all infeasible resource allocations for all dependent tri-tasks, i.e., it removes from the sets A^{ij} all resource allocations for which there is a conflict with resource allocations for at least one dependent tri-task. As a result we obtain sets of resource allocations A_w^{ij}. The generation of the feasible resource allocation for the entire workflow has been moved to the second phase of the presented approach.

4. Metascheduling

The problem of finding a schedule for which the makespan is minimized is NP-hard because if we take into account only one resource allocation then the problem can be treated as MRCPSP, which is also NP-hard. Thus, it is justified to use metaheuristic approaches to solve the considered problem. Since, some time ago (Józefowska et al. 2001) developed a simple and effective simulated annealing algorithm for the MRCPSP, we use some ideas from that approach in this work developing simulated annealing algorithm for the considered problem. The main idea of the new approach is to use a representation of the solution that allow to represent different feasible resource allocation as well as different schedules for this resource allocation. The representation will be described later. Now it will be described how this algorithm works.

After the execution of the resource allocation phase we have the sets of feasible resource allocations A_w^{ij} of all tri-tasks, and we can start to construct a feasible resource allocation for the entire workflow W. Let's assume that pairs $R_w^{ij} = (X_\chi, X_\theta)$ can be treated as domino bones with χ spots on the left end, and θ spots on the right end. Now, the problem consist in covering all arcs of workflow W with those domino bones in such a way that arc (v_i, v_j) is covered with a bone from set A_w^{ij} and, of course, in each node every bone must have the same of spots on the nod-adjacent end. Notice, that the orientation of a domino bone is important, because we try to cover directed graph, and therefore the left end of the domino bone have to be placed to the start of the arc, and right end of the same domino bone have to be placed to the end of the same arc. Of course, it is possible that in some cases there will be more than one feasible resource allocation for the entire workflow possible. In such a case it is necessary to define rules for choosing only one such allocation. In the proposed approach feasible resource allocation is generated according to the part of the solution representation responsible for assigning computational tasks to resource nodes, and assigning transmission tasks to connections of required bandwidths.

Now, for a given feasible resource allocation the scheduling algorithm works as follows. Firstly, we remove all resource constraints and find the length of the critical path using the Critical

Path Method (CPM) assuming that all computational tasks are executed on the fastest available processors, and the transmission tasks are performed over the shortest connection path. The obtained schedule length is treated as a lower bound for the considered instance of the problem. Next we check if any resource conflict occurs. If not, then the obtained schedule is the optimal one for this resource allocation, and the obtained schedule as well as feasible resource allocation are stored in the memory, and we pass to the generation of the next feasible resource allocation. If yes, then we check if it concerns network resources only. In such a case, we try to find alternative path. If it is impossible or resource conflicts concern computational resources we treat this schedule as infeasible one, and try to find a schedule in another more complicated way using simulated annealing algorithm in which a special representation of the solution as well as the corresponding neighbourhood generation mechanism are used.

In this representation we use three lists. The first one, with size equal to the number of tri-tasks, represents the choices of elements from sets A_w^{ij}. Of, course these choices have to represent a feasible resource allocation for the entire workflow W. The second one, which size is equal to the number of computational tasks, contains the execution modes of the corresponding computational tasks. The third one, which size is equal to the sum of numbers of computational tasks and transmission tasks is a precedence feasible lists of tasks (both computational and transmission ones), which represent the priority of the task. This priority is used during the procedure which is used to construct a feasible schedule. A list of tasks is precedence feasible if: i) each computational task occurs after all its direct predecessors and before all its direct successors, ii) each transmission task (v_i, v_j) occurs after the computational tasks v_i and before computational task v_j. We use a modified version of the well-known Serial SGS (Schedule Generation Scheme) as a decoding rule which is used to construct a schedule from the considered two list representation of the solution. The main idea of this procedure is very simple. It takes consecutive tasks from the list of tasks according to their priorities and try to schedule them as soon as possible. It means that the earliest feasible start time for a given task always occurs after the completion time of all the direct predecessors of it. Moreover, in order to schedule a given task at time t it have to be guaranteed that all required resources are available in sufficient amounts for the whole duration of this task. The main differences between the standard version of the serial SGS and our modified version is that in the presented approach we have resources distributed over several locations, and that we have two types of tasks. Computational tasks are scheduled as soon as possible after all its directly preceding transmission tasks are completed and sufficient amounts of required resources are available for the duration of this activity in the proper location. Transmission tasks are scheduled as soon as possible after the directly preceding computational task is finished. Network resources are allocated to the transmission task according to the shortest (in terms of the number of nodes) path calculated during the execution of the considered procedure.

Now, when we have defined the representation of the solution and the decoding rule we can define the neighbourhood generation mechanism. The first list is always generated in random way taking into account the dependencies of tri-tasks. The new mode assignment is obtained by randomly choosing an activity and executing it on the next group of processors with lower or higher speed factors. Finally, the priority of all tasks defined by the third list can be changed using the shift operator.

The performance of the proposed approach has been checked on the basis of computational experiment that uses some synthetic data.

Acknowledgements

This work has been funded by the Polish National Science Centre as a research project in the years 2013–2016 under grant No. 2013/08/A/ST6/00296.

References (Final headings style)

Józefowska J., M. Mika, R. Różycki, G. Waligóra, J. Węglarz, 2001, "Simulated annealing for multi-mode resource-constrained project scheduling", Annals of Operational Research, Vol. 102, pp. 137-155.

Mika M., G. Waligóra, J. Węglarz, 2011, "Modelling and solving grid resource allocation problem with network resources for workflow applications", Journal of Scheduling, Vol. pp. 291-306.

A Continuous-Time Model for the Resource-Constrained Project Scheduling with Flexible Profiles

Anulark Naber[1] and Rainer Kolisch[2]

TUM School of Management, Technische Universität München, Germany
[1]anulark.naber@tum.de, [2]rainer.kolisch@tum.de

Keywords: project scheduling, flexible profiles, continuous-time model, continuous resources.

1. Introduction

The classical resource-constrained project scheduling problem (RCPSP) deals with scheduling of *n* activities to minimize the project makespan, subject to the technological precedence constraints and the limited availability of resources. While the resource allocation over the duration of each activity is given and normally constant in the RCPSPs, in this research, it is not necessarily the case. Here, we allow resource usage to vary over the activity duration, thus, yielding a nonconstant resource usage profile. Under these new circumstances, the resource usage in each time period and the duration of each activity are unknown a priori and, thus, need to be simultaneously determined while scheduling activities by their starting times. This problem is termed here as the RCPSP with **F**lexible resource profiles (FRCPSP), a generalization of the RCPSP.

The FRCPSP simultaneously determines an optimal schedule of activities and their resource profiles that minimize the project makespan. Each activity requires, once started, one nonpreemptive profile per resource over its entire processing time. The project schedule must also observe all precedence requirements depicted in a precedence network, namely finish-start with zero time lag between two activities. The flexible profiles must additionally satisfy the following constraints: (1) The total amount of each required resource assigned to each activity over its duration must at least satisfy its resource requirement. (2) There must be at least a number of consecutive periods having a constant resource usage, called the minimum block length (Fündeling and Trautmann 2010). (3) The resource usage per time must be within a specified range designated by a lower and an upper bound. In this research, all resource amounts are assumed continuous and all resources are renewable.

Resources required by an activity are categorized here into the three categories, namely principal, dependent, and independent resources. A principal resource of an activity is the main resource whose usage amount may be depended upon or used as a computational basis by dependent resources. Although different principal resources may be required by different activities, only one principal is designated per activity. Dependency between resources is characterized by a linear resource function. An independent resource, on the other hand, is independent from other resources, although its start and finish times must synchronize with those of other resources required by the same activity. Regardless of the resource category, the availability of all resources per time is constant, independent, and not subject to any resource function.

The FRCPSP was firstly introduced by Kolisch *et al.* (2003) and subsequently studied by, for example, Fündeling and Trautmann (2010) and Ranjbar and Kianfar (2010). Most existing works propose heuristic methods for solving the FRCPSP with discrete resources. Lately, Naber and Kolisch (2013) propose and compare four discrete-time MIP models, in which the discrete-time system is commonly used for all activities and resources. As such, each activity must start at the beginning of a time period and finish at the end of a time period. According to Koné *et al.* (2011) who evaluate efficiency of several discrete- and continous-time models for the RCPSP, the continuous-time models, in general, are more efficient for instances of large scheduling horizons than the discrete-time models. Thus, we propose, in this paper, a continuous-time event-based model for the FRCPSP, using a single continuous-time grid system (Castro and Grossmann, 2012) to syncronize activities and resources.

2. Outline of A Continuous-Time MIP Model for FRCPSPs

In this section, we outline a continuous-time MIP model formulation for the FRCPSP, based on the FT-DT3 discrete-time model proposed by Naber and Kolisch (2013) and the Start/End event-

based model of Koné *et al.* (2011). We define three types of events that may occur, namely *Start*, *End*, and *Change* (in resource quantity per time) events. Events of several activities may occur simultaneously at the same point in time. Through time, the interval between two consecutive events designates the time difference between the occurrences of the two events.

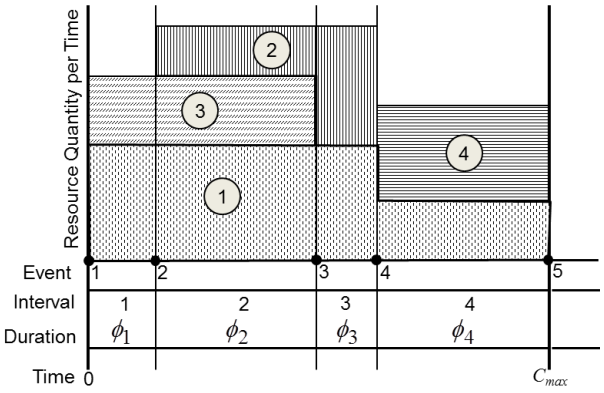


Figure 1. Example of events and a continuous-time schedule

Figure 1 demonstrates, as an example, for activity 2, Start at event 2 or at the beginning of interval 2, Change at event 3 or at the beginning of interval 3 and End at event 4 or at the beginning of interval 4 or the end of interval 3. Despite their different definitions, events and intervals are used here interchangeably.

In terms of decision variables, we define ϕ_e as the duration of interval e, $\forall e \in \mathbb{E}$, the set of discrete intervals, given an upper bound of the number of intervals or events. Continuous durations of intervals are, thus, to be solved by the model to minimize the project makespan, which is indeed the sum of all interval durations. Additional continuous variable q_{rje}, termed *interval quantity*, is defined as the cumulative quantity of resource r allocated to activity j in interval e. In Figure 1, q_{rje} is a rectangle area equal to ϕ_e (width) multiplied by the resource quantity per time (height). Similarly defined as in the model FP-DT3 of Naber and Kolisch (2013), binary decision variables are used to indicate Starting and Stopping events of activities and Change events in resource quantity per time.

The main constraints are outlined as follows: (1) A lower and an upper bound on the continuous makespan; (2) Nonpreemption requiring the continuity of processing status of activity j, once started; (3) Precedence relationships between activities; (4) Total resource requirement for each activity; (5) Resource functions between dependent and principal resources; (6) Upper and lower bounds on resource quantity per time; (7) Limited resource availability; (8) Change in resource quantity per time from one event to the next. The change is characterized by either change in interval quantity or time duration or both.

3. Implementation and Preliminary Results

Similar to other RCPSP models, our continuous-time MIP model is also pseudo-polynomial in the number of events. It is, thus, useful to evaluate an upper bound of the makespan. Given a heuristic discrete schedule obtained by Naber and Kolisch (2013), we perform a simple right-cut-left-paste of each activity, where possible. Starting from Time 0, the algorithm iteratively cuts short an activity to which excessive resources are allocated due to the discrete time periods and left-pastes the successively scheduled activities, so they can start earlier. To shorten an activity, the minimum block length must also be observed. Figure 2 illustrates an example of the continuous-time schedule (b) obtained from applying the right-cut-left-paste algorithm to a discrete-time schedule (a).

Through the forward and backward preprocessing calculations as presented in Naber and Kolisch (2013) for the discrete-time models, we can also compute the continuous earliest and latest start and finish times of activity j. These time parameters are then ordered in a nondecreasing order to form intervals, which are, in turn, divided by the minimum value among the minimum block

length of activities that concurrently compete for resources. The earliest and latest time parameters are then heuristically mapped to the earliest and latest event parameters accordingly.

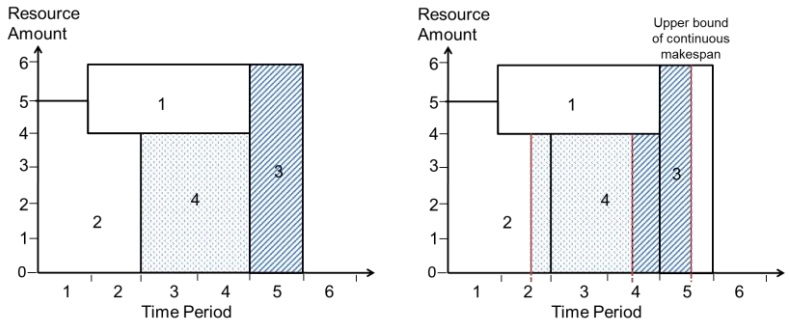


Figure 2. An example of the continuous-time schedule derived from a discrete-time schedule

The number of events in this model is, in general, significantly lower than the number of discrete-time periods in the discrete-time models, implying more than 50% lower number of both binary and continuous variables. However, the preliminary computational results, even on small instances, do not at all suggest faster computational times in solving the continuous-time model. On the contrary, we observe that, although optimal solutions are often found relatively fast in earlier branches, the convergence to optimality is indeed poor, due to the nonimprovement of the lower bound of the branch-and-cut tree. This phenomenon suggests a possibility to obtain good heuristic solutions from the truncated branch-and-cut.

One may conjecture of a very loose LP relaxation polytope in comparison with the convex hull of MIP solutions, due to the infinite number of possible time durations and consequently resource allocations. In contrast to the discrete-time models, the standard cuts implemented in IBM ILOG CPLEX cannot be applied to the continuous-time model as aggressively and effectively. Especially, the clique and cover inequalities are rarely applied to the continuous-time model. In order to tighten the LP relaxation polytope, our first attempt to speed up the solution times is then to append simple global valid inequalities to the model. These inequalities are imposed on the relations between time durations and activity variables. However, the need for further research to derive stronger valid inequalities for faster convergence is still evident.

Acknowledgements

This research is funded by the German Research Foundation (Deutsche Forschungs-gemeinschaft, DFG) of the German government.

References

Castro, P. M., I. E. Grossmann, 2012, "From time presentation in scheduling to the solution of strip packing problems", Computers and Chemical Engineering, Vol. 44, pp. 45-57.

Fündeling, C.-U., N. Trautmann, 2010, "A priority-rule method for project scheduling with work-content constraints", European Journal of Operational Research, Vol. 203, pp. 568-574.

Kolisch, R., K. Meyer, R. Mohr, C. Schwindt and M. Urmann, 2003, „Ablaufplanung für die Leitstrukturoptimierung in der Pharmaforschung", Zeitschrift für Betriebswirtschaft, Vol. 73, pp. 825-848.

Koné, O., C. Artigues, P. Lopez and M. Mongeau, 2011, "Event-based MILP models for resource-constrained project scheduling problems", Computers & Operations Research, Vol. 38, pp. 3-13.

Ranjbar, M., F. Kianfar, 2010, "Resource-constrained project scheduling problem with flexible work profiles: a genetic algorithm approach", Transaction E: Industrial Engineering, Vol. 17, pp. 25-35.

Naber, A., R. Kolisch, 2013, "MIP models for resource-constrained project scheduling with flexible resource profiles", Working paper, TUM School of Management, Technische Universität München, Germany.

A polynomial satisfiability test using energetic reasoning for energy-constraint scheduling

Margaux NATTAF[1,2], Christian ARTIGUES[1,2] and Pierre LOPEZ[1,2]

[1]CNRS; LAAS; 7 Avenue du colonel Roche, F-31077 Toulouse, France
[2] Univ de Toulouse, LAAS, F-31400 Toulouse, France
{nattaf,lopez,artigues}@laas.fr

Keywords: continuous resource, energy constraints, energetic reasoning, satisfiability test.

1 Introduction

A famous cumulative global constraint is the cumulative one which leads to the Cumulative Scheduling Problem (CuSP). In this problem, given a resource with a limited capacity and a set of activities each one having a release date, a due date, a duration and a resource requirement, we want to schedule all activities in their time window and without exceeding the capacity limit of the resource.

For this NP-complete problem, solution methods exist and, recently, techniques using satisfiability formulas have been developed. Considering more particularly constraint-based scheduling, which can be seen as a way to solve scheduling problems using constraint programming (Baptiste *et. al.* 2001), a technique using energetic reasoning provides a strong (although incomplete) polynomial satisfiability test known as the "left-shift/right-shift" conditions (Baptiste *et. al.* 1999).

In this paper, the idea is to use the "left-shift/right-shift" test and energetic reasoning propagation algorithms (Erschler and Lopez 1990), (Lopez and Esquirol 1996) in order to compute a polynomial satisfiability test for a generalization of CuSP, the Continuous Energy-Constrained Scheduling Problem (CECSP). The CECSP has the following particularity: activities use a continuously-divisible resource and each of them can take any shape bounded by its time window, a minimum and maximum resource requirement and a fixed energy requirement that has to be brought via a power processing rate function.

As the CECSP deals with a continuous resource, propagation techniques already developed can not be applied directly. Thus, it is an interesting question to know whether a polynomial satisfiability test can be found for CECSP. In our study, we adapt the "left-shift/right-shift" test for CuSP to CESCP.

2 Problem statement

In the CECSP problem, we have as input a set $A = \{1, \ldots, n\}$ of activities and a continuous resource which is available in a limited capacity B. Each activity has to be performed between its release date r_i and its deadline $\tilde{d}_i$. Instead of being defined by its duration and resource requirement, in CECSP an activity is defined by an energy requirement W_i, a minimal and maximal resource requirement b_i^{min} and b_i^{max}.

To solve the CECSP, we have to find for each activity its starting time st_i, its finishing time ft_i and a function $b_i(t)$, for all $t \in \mathcal{T}$ (where $\mathcal{T} = [\min_{i \in A} r_i, \max_{i \in A} \tilde{d}_i]$), representing the amount of the resource allocated to this activity. These variables have to verify the

following equations :

$$r_i \leq st_i \leq ft_i \leq \tilde{d}_i \qquad (i \in A) \qquad (1)$$

$$b_i^{min} \leq b_i(t) \leq b_i^{max} \qquad (i \in A; t \in [st_i, ft_i]) \qquad (2)$$

$$b_i(t) = 0 \qquad (i \in A; i \in \mathcal{T} \setminus [st_i, ft_i]) \qquad (3)$$

$$\int_{st_i}^{ft_i} f_i(b_i(t))dt = W_i \qquad (i \in A) \qquad (4)$$

$$\sum_{i \in A} b_i(t) \leq B \qquad (t \in \mathcal{T}) \qquad (5)$$

where $f_i(b)$ is a continuous non-decreasing power processing rate function.

We define $p_i = ft_i - st_i$ as the activity duration. We remark that if $b_i^{min} = b_i^{max} = b_i$ and $f_i(b_i(t)) = b_i(t),\ \forall i \in A$, then we can set p_i to $\frac{W_i}{b_i}$ and, if all inputs are integers, we obtain an instance of CuSP. Thus, the CECSP is NP-complete.

We first have looked at the case where function $f_i(b)$ is linear or piecewise linear. In these cases, we have adapted a satisfiability test for the CECSP, where $f_i(b_i(t)) = b_i(t),\ \forall i \in A$ (Artigues *et. al.* 2009), to this more complex case.

3 Energetic reasoning for linear function

We first present the case where the function $f_i(b)$ has the form $a_i b + c_i$ with $a_i > 0$ and $c_i > 0$.

Before explaining how energetic reasoning yields a polynomial satisfiability test for CECSP, we present an elementary satisfiability condition to check whether the activity data is consistent. This condition is the following: if we can find an activity $i \in A$ such that $f_i(b_i^{max})(\tilde{d}_i - r_i) < W_i$ then the CECSP can not have a solution. This comes from the fact that, since $f_i(b)$ is a non-decreasing function, scheduling i at its maximum resource requirement b_i^{max} inside $[r_i, \tilde{d}_i]$ gives the largest amount of energy.

In order to apply energetic reasoning to our problem, we have considered the minimum energy requirement and resource consumption of an activity i over an interval $[t_1, t_2]$. These values are denoted by $\underline{w}(i, t_1, t_2)$ and $\underline{b}(i, t_1, t_2)$ respectively and defined by :

$$\underline{w}(i, t_1, t_2) = \min \int_{t_1}^{t_2} f_i(b_i(t))dt \qquad \text{subject to (1)–(4)}$$

$$\underline{b}(i, t_1, t_2) = \min \int_{t_1}^{t_2} b_i(t)dt \qquad \text{subject to (1)–(4)}$$

We have used these values to compute the slack of interval $[t_1, t_2]$ which is defined by: $SL(t_1, t_2) = B(t_2 - t_1) - \sum_{i \in A} \underline{b}(i, t_1, t_2)$. The satisfiability test consists to determine whether there exists an interval $[t_1, t_2]$, with $t_1 < t_2$, such that $SL(t_1, t_2) < 0$. If such an interval exists then the CECSP has no solution.

This proposition is the center of the "left-shift/right-shift" necessary condition. In (Baptiste *et. al.* 1999), it was shown that this test can be performed only on a polynomial number of intervals for CuSP. For CECSP, we have to check whether a polynomial number of intervals is sufficient to perform the satisfiability test.

To achieve this, we have analyzed possible configurations of the minimum resource consumption. First, since $f_i(b)$ is a non-decreasing function, we can observe that, given an interval $[t_1, t_2]$, the minimum consumption always corresponds to a configuration where activity i is either left-shifted (the activity starts at r_i and is scheduled at its maximum requirement between r_i and t_1) or right-shifted (the activity ends at $\tilde{d}_i$ and is scheduled

at its maximum requirement between t_2 and $\tilde{d}_i$), or both. We will denote by I the interval over which the activity is scheduled at b_i^{max} outside interval $[t_1, t_2]$. Then, the minimum energy requirement in $[t_1, t_2]$ is: $\underline{w}(i, t_1, t_2) = W_i - |I| * f_i(b_i^{max})$.
We still have to compute the minimum required resource consumption over $[t_1, t_2] \setminus I$ to obtain this energy. There are two cases to consider :

- the remaining interval is sufficiently large to schedule the activity at its minimum requirement, i.e. $|[t_1, t_2] \setminus I| \geq \frac{\underline{w}(i,t_1,t_2)}{f_i(b_i^{min})}$, and then $\underline{b}(i, t_1, t_2) = b_i^{min}(\frac{\underline{w}(i,t_1,t_2)}{f_i(b_i^{min})})$
- the remaining interval is not large enough to schedule the activity at its minimum requirement and finding $\underline{b}(i, t_1, t_2)$ is equivalent to solving :

$$\text{minimize} \int_{[t_1,t_2]\setminus I} b_i(t)dt$$
$$\text{subject to} \int_{[t_1,t_2]\setminus I} f_i(b_i(t))dt \geq \underline{w}(i, t_1, t_2)$$

Then $\underline{b}(i, t_1, t_2) = \frac{1}{a_i}(\underline{w}(i, t_1, t_2) - |[t_1, t_2] \setminus I|c_i)$

The function $\underline{b}(i, t_1, t_2)$ defined in this way is a bivariate continuous piecewise linear function. This remark allows us to establish a theorem which states that we can perform the satisfiability test only on a polynomial number of intervals. Indeed, we want to check whether an interval $[t_1, t_2]$ over which the slack function is negative exists. Since the slack function is a two dimensional piecewise linear function, we only have to check whether this occur at the extreme point of one of the convex polygon on which it is linear.

The break line segments of the slack function is the same as the ones of the sum of the minimum consumption for each activity. Thus, each extreme point of the slack function is the intersection of two segments, each segment corresponding to the break line segment of an individual minimum consumption function. Thus, we only have to perform the satisfiability test on these intersection points whose number is quadratic in the number of activities.

4 Piecewise linear function

Consider now the case where $f_i(b)$ is a continuous non-decreasing piecewise linear function. As the function is non-decreasing we can perform the same test as the one for linear functions to check whether the activity data is consistent. The minimum required energy and resource consumption $\underline{w}(i, t_1, t_2)$ and $\underline{b}(i, t_1, t_2)$ are defined in the same way.

To compute the slack function of an interval $[t_1, t_2]$, we need an analytical expression of function $\underline{b}(i, t_1, t_2)$. To achieve this, the function $\underline{w}(i, t_1, t_2)$ is computed. Actually, possible configurations of the minimum resource consumption are the same as for the case of linear functions (left-shifted activity, right-shifted or both). Thus, we can compute $\underline{w}(i, t_1, t_2)$ in the same way.

For a piecewise linear function, the difficulty lies in the computation of $\underline{b}(i, t_1, t_2)$. In this case, we are not able to derive the expression of minimum resource consumption from a linear program. However, we can analyze the function $f_i(b)$ to find the point of best energetic efficiency, i.e. the point for which $\frac{f_i(b)}{b}$ is maximal for $b_i^{min} \leq b \leq b_i^{max}$. Let γ be this point. Once γ is calculated, we can use it to exhibit a lower bound for $\underline{b}(i, t_1, t_2)$. Indeed, we know that to provide the required energy to the activity, the minimum resource consumption is obtained by allocating this amount of energy to it during a sufficiently large time. Thus, $\underline{b}(i, t_1, t_2) \geq \gamma \frac{\underline{w}(i,t_1,t_2)}{f_i(\gamma)}$.

We can perform the satisfiability test by setting $\underline{b}(i, t_1, t_2)$ to $\gamma \frac{\underline{w}(i,t_1,t_2)}{f_i(\gamma)}$ and (as $\underline{b}(i, t_1, t_2)$

is also a bivariate continuous piecewise linear function) a polynomial number of intervals is sufficient. However, the time required to compute γ depends on the number of definition intervals of the function $f_i(b)$. But it seems difficult to compute $\underline{b}(i, t_1, t_2)$ in a time independent of this number.

5 Conclusion

We have presented a polynomial satisfiability test for two variants of the CECSP. This work is still in progress, especially for finding the exact minimum required resource consumption expression for piecewise linear function.

For future research, in order to provide better applications to actual scheduling problems under energy constraints, it will be interesting to study the case where function $f_i(b)$ is no longer linear. Another interesting problem is to integrate energetic reasoning in other bounding techniques such as linear programming or network flow. In addition we may also extend the time windows adjustment included in energetic reasoning for the CuSP to our problem.

References

C. Artigues , P. Lopez , A. Haït , 2009, "Scheduling under energy constraints", *International Conference on Industrial Engineering and Systems Management (IESM 2009)*, Montréal, Canada.

P. Baptiste , C. Le Pape , W. Nuijten , 1999, "Satisfiability tests and time-bound adjustments for cumulative scheduling problems", *Annals of Operations Research*, Vol. 92, pp. 305-333.

P. Baptiste , C. Le Pape , W. Nuijten , 2001, *Constraint-based scheduling*, Kluwer Academic Publishers, Boston/Dordrecht/London .

J. Błażewicz , K. H. Ecker , E. Pesch , G. Schmidt , J. Węglarz 2001, *Scheduling computer and manufacturing processes*, Springer-Verlag, Berlin/Heidelberg.

J. Erschler , P. Lopez , 1990, "Energy-based approach for task scheduling under time and resources constraints", *2nd International Workshop on Project Management ans Scheduling*, pp. 115-121, Compiègne, France.

P. Lopez and P. Esquirol , 1996, "Consistency enforcing in scheduling: A general formulation based on energetic reasoning", *5th International Workshop on Project Management and Scheduling*, pp. 155-158, Poznań, Poland.

Column-Generation Based Lower Bounds for Resource Leveling Problems

Tobias Paetz[1] and Christoph Schwindt[1]

Clausthal University of Technology, Germany
{tobias.paetz,christoph.schwindt}@tu-clausthal.de

Keywords: Resource leveling, lower bounds, column generation, generalized precedence relations.

1 Introduction

In this paper we present a method for computing tight lower bounds on the optimum objective function value of resource leveling problems arising in project management or production scheduling. The method relies on the representation of a schedule as a sequence of antichains of the precedence order. By associating a decision variable with the execution time of each antichain, leveling criteria like resource usage variance or resource overload cost can be expressed as linear objective functions. By substituting the precedence relationships between the activities into disjunctive constraints and allowing activities to be interrupted during their execution, the leveling problem is relaxed to a large-scale linear program. This linear program can be solved using a column generation procedure, where the pricing problem represents a convex stable set problem on a perfect graph. To enhance the quality of the lower bounds that arise from solving the linear program, we tighten the relaxation by identifying antichains that must be executed in each feasible schedule. We evaluate the quality of the lower bounds using different test sets from literature.

Resource leveling deals with the problem of smoothing the usage of renewable resources over time subject to a prescribed maximum project duration. The first procedures for resource leveling date back to the early days of project scheduling in the 1960's. A recent review of exact algorithms for resource leveling can be found in Rieck and Zimmermann (2014). The performance of these algorithms strongly depends on the effectiveness of the lower bounding procedures. Computing tight lower bounds for resource leveling problems is difficult because lower approximations to the resource usage over time that are based on interval workload considerations tend to be rather loose. This observation is particularly true for the resource usage variance problem, which will be investigated in more detail in what follows. The concepts presented in this paper carry over to further leveling problems like the resource overload cost problem. Related models for computing lower bounds on the minimum duration of resource-constrained projects with ordinary precedence relations were considered by Mingozzi et al. (1998) and Brucker and Knust (2000).

Consider a project with activities $i \in V$ of durations $p_i > 0$ for which a schedule S assigning the start times S_i has to be determined. During its execution, each activity i requires r_{ik} units of several renewable resources $k \in \mathcal{R}$ like personnel or machinery. We assume that the activities must not be interrupted once they have been started. For certain pairs of activities $(i, j) \in E$, prescribed time lags δ_{ij} between start times S_i and S_j define generalized precedence relationships $S_j \geq S_i + \delta_{ij}$ that have to be observed when scheduling the activities. Furthermore, the project must be completed within a specified maximum time period $\overline{d}$. The resource leveling problem consists in scheduling the activities in such a way that some given measure in the variability of the resource usages over time is minimized. We assume that the variability is expressed in terms of the weighted sum of usage variances. By removing the constants, the objective function can be written as $\sum_{k\in\mathcal{R}} w_k \int_0^d r_k^2(S,t)\,dt$ with weights $w_k \geq 0$ and $r_k(S,t) := \sum_{i\in V: S_i \leq t \leq S_i+p_i} r_{ik}$ denoting the usage of resource k at time t. The leveling problem under consideration can now be

formulated in the following way:

$$(P)\quad \begin{cases} \text{Minimize} & f(S) = \sum_{k\in\mathcal{R}} w_k \int_0^{\overline{d}} r_k^2(S,t)\,dt \\ \text{subject to} & S_j \geq S_i + \delta_{ij} \quad ((i,j)\in E) \\ & S_i + p_i \leq \overline{d} \quad (i\in V) \\ & S_i \geq 0 \quad (i\in V) \end{cases}$$

The balance of the paper is organized as follows. In Section 2 we reformulate problem (P) using the antichain representation of schedules and explain how we obtain a linear relaxation of the problem. The column generation algorithm for solving the relaxation is explained in Section 3. In Section 4 we describe preprocessing techniques that were applied to tighten the relaxation and the resulting lower bounds. Section 5 is devoted to an experimental performance analysis of different variants of our lower bounding procedure.

2 Reformulation and linear relaxation

Let d_{ij} denote the transitive time lag between activities i and j that is implied by the prescribed time lags δ_{gh} with $(g,h)\in E$ and the project deadline $\overline{d}$. Time lags d_{ij} can be calculated as longest path lengths in an appropriate activity-on-node network (see, e. g., Neumann et al. 2003, Sect. 1.3 for details). The generalized precedence relations give rise to a precedence order $\Theta = \{(i,j)\in V\times V \mid d_{ij}\geq p_i\}$ among the activities. The set $\mathcal{A}$ of antichains of strict order Θ contains all sets $A\subseteq V$ of activities that can be processed simultaneously at some point in time. Each feasible schedule S can be encoded as a sequence of antichains with durations $x_A > 0$ and resource requirements $r_{Ak} = \sum_{i\in A} r_{ik}$. Based on this encoding, the objective function value $f(S)$ can be expressed as the linear function $g(x) = \sum_{A\in\mathcal{A}}(\sum_{k\in\mathcal{R}} c_k r_{Ak}^2)\,x_A$. Considering x_A with $A\in\mathcal{A}$ to be decision variables leads to the following reformulation (P') of leveling problem (P), where $c_A = \sum_{k\in\mathcal{R}} w_k r_{Ak}^2$.

$$(P')\quad \begin{cases} \text{Minimize} & g(x) = \sum_{A\in\mathcal{A}} c_A x_A & & \\ \text{subject to} & \sum_{A\in\mathcal{A}:i\in A} x_A = p_i & (i\in V) & (1) \\ & \sum_{A\in\mathcal{A}} x_A = \overline{d} & & (2) \\ & x_A \geq 0 & (A\in\mathcal{A}) & \\ & \text{Side constraints} & & \end{cases}$$

From equations (1) and (2) it follows that within the project lifespan, each activity $i\in V$ is carried out completely and that the project is terminated on time. The side constraints ensure that the original time lags δ_{ij} between single activities $i,j\in V$ are satisfied and that no activity i is interrupted during execution. It is easily seen that these conditions can be represented as the constraints of a single machine problem with generalized precedence relations among the jobs, where the jobs correspond to the antichains A with positive durations $x_A > 0$. By deleting the side constraints from problem (P'), we obtain a linear program (LP) whose size grows exponentially with the number of activities. Solving (LP) delivers a lower bound on the optimum objective function value of leveling problem (P). In the next section, we explain how (LP) can be solved efficiently using column generation.

3 Column generation algorithm

The main idea of column generation consists in generating the columns of the coefficient matrix of the linear program as required rather than in advance (see Goldfarb and Todd 1989, Sect. 6). Starting with an initial basic solution x, in each iteration a nonbasic variable x_A with minimum reduced costs ζ_A is computed; the problem of identifying such a nonbasic variable is called the pricing problem. If $\zeta_A < 0$, the procedure pivots according to the

minimum ratio rule and proceeds with the resulting basic solution until the pricing problem proves the optimality of x by yielding $\zeta_A = 0$. The runtime of the algorithm is mainly determined by the time required to solve the pricing problems. That is why the tractability of the pricing problem is crucial to the performance of the algorithm.

Let u_i and v denote the simplex multipliers associated with equations (1) and (2). Given some basic solution x with corresponding basic matrix B, the values of the multipliers are computed as $(u^\top, v) = ([c^B]^\top, 0) \cdot B^{-1}$. Since the constraint $\sum_{i \in A} u_i + v \leq c_A$ $(A \in \mathcal{A})$ of the dual of (LP) is equivalent to optimality condition $\zeta_A \geq 0$ $(A \in \mathcal{A})$, we know that

$$\zeta_A = c_A - \textstyle\sum_{i \in A} u_i - v \quad (A \in \mathcal{A})$$

The pricing problem consists in finding an antichain A^* with minimum reduced costs ζ_{A^*}. To formulate the problem we introduce a binary variable y_i for each activity $i \in V$, where $y_i = 1$ means that $i \in A^*$. Using the variables y_i, cost coefficient c_{A^*} can be expressed as $\sum_{k \in \mathcal{R}} w_k (\sum_{i \in V} r_{ik} y_i)^2$ and the pricing problem can be stated as binary convex program

$$(PP(u,v)) \quad \begin{cases} \text{Minimize} & h(y) = \sum_{k \in \mathcal{R}} w_k (\sum_{i \in V} r_{ik} y_i)^2 - \sum_{i \in V} u_i y_i - v \\ \text{subject to} & y_i + y_j \leq 1 \qquad ((i,j) \in \Theta) \\ & y_i \in \{0,1\} \qquad (i \in V) \end{cases} \tag{3}$$

The disjunctive constraint (3) models the requirement that A^* must be an antichain of precedence order Θ. The pricing problem can be interpreted as a stable set problem on the comparability graph G of Θ. More precisely, it represents a convex stable set problem on a perfect graph, since G is transitively orientable and h is a convex quadratic function. In the experimental performance analysis discussed in Section 5 it turned out that this type of problem is solved quite efficiently by commercial MIQCP solvers.

4 Preprocessing techniques

The quality of the lower bounds can be improved by tightening the linear program using preprocessing techniques. We applied two different methods:

1. Replace positive completion-to-start-time lags $\delta_{ij} - p_i$ by dummy activities.
2. Identify unavoidable antichains A, which must be in execution in any feasible schedule.

The first method is very simple, but it may be of considerable benefit because the disjunctive constraints between activities i, j and the new dummy activity are taken into account in the pricing problem. The second method is of particular interest to resource leveling problems. For this method we take advantage of the following

Proposition 1. *Let $\emptyset \neq A \subseteq V$. Then all activities $i \in A$ are processed in parallel during at least $p(A) = \max\{0, \min_{i,j \in A}(d_{ij} + p_j)\}$ time units. The bound is tight, i. e., there always exists a feasible schedule with $x_A = p(A)$ provided that the problem is feasible.*

It is easily seen that $p(A) = \min_{i \in A} p(A \setminus \{i\})$ for all antichains A with $|A| \geq 2$. Therefore the unavoidable antichains can be computed recursively as cliques in a graph $G' = (V, E')$ with set of edges $E' = \{\{i,j\} \mid p(\{i,j\}) > 0\}$. The second preprocessing method now works as follows. We start by selecting some inclusion-maximal clique A of graph G'. Next, we add the corresponding antichain A as a new activity i_A with duration $p_{i_A} = p(A)$ and resource requirements $r_{i_A k} = r_{Ak}$ to the project. Then, we reduce the minimum durations $p(A')$ of all intersecting antichains A' by the maximum overlap time $\Delta = [\min\{p(A), p(A'), -\max_{i \in A, j \in A'} \max\{d_{ij} + p_i, d_{ji} + p_j\}\}]^+$ between A and A' and update the edge set E' of graph G' accordingly. We proceed in the same way until $E' = \emptyset$. Finally, we enhance pricing problem $(PP(u,v))$ by the disjunctive constraint $y_{i_A} + y_{i_{A'}} \leq 1$ for all new activities $i_A, i_{A'}$ with $A \neq A'$ and $A \cap A' \neq \emptyset$. This condition ensures that any two intersecting antichains A and A' added to the project are not executed in parallel. To facilitate the construction of a feasible initial basic solution, the condition is added to the pricing problem via a penalty term in the objective function.

5 Computational results

We tested the efficiency of our methods using three standard test sets from literature. The column generation algorithm was coded under GAMS 24.0 invoking Gurobi 5.0 as LP and as MIQCP solver. The computational experiments were performed on a dual core PC with 3.16 GHz clock pulse and 3 GB RAM operating under Windows 7. Four variants of the lower bounding method were tested: column generation without preprocessing (CG1), with positive completion-to-start-time lags as dummy activities (CG2), with pre-generated unavoidable antichains (CG3), and with both preprocessing techniques (CG4).

Table 5 summarizes the results that we obtained for the j10, j20, and j30 test sets of Kolisch et al. (1999), each containing 270 instances with 10, 20, or 30 activities. The hardness of the instances is largely influenced by the deadline factor DF, which varies between 1.0, 1.1, and 1.5 for all test sets. The deadline factor is defined to be the ratio of the project deadline $\overline{d}$ and the minimum project duration $\max_{i,j \in V}(d_{ij} + p_j)$. Roughly speaking, the higher the deadline factor the harder the instances of a resource leveling problem. The values displayed in Table 5 are the mean relative deviations from the optimum objective function values published by Rieck et al. (2012) for the three test sets. Reference values for the j30 test set with $DF = 1.5$ are not available. As expected, the quality of the lower bounds gradually improves from CG1 to CG4. Especially for the hard instances with $DF = 1.5$, variant CG4 performed quite well, yielding gaps of less than one percent. The mean computation time per instance varied between 4 to 343 sec. with a mean number of pivot iterations between 11 and 1159. The relatively high maximum CPU times and iteration numbers are due to the fact that for certain instances the preprocessing techniques generated several hundreds of dummy activities. Similar mean relative deviations from optimum objective function values were obtained for the resource overload cost problem.

Table 1. Mean relative deviations from optimum objective function values

	j10			j20			j30	
DF	1.0	1.1	1.5	1.0	1.1	1.5	1.0	1.1
CG1	7.78 %	6.06 %	1.86 %	8.85 %	5.43 %	1.85 %	9.64 %	5.79 %
CG2	4.43 %	5.23 %	1.83 %	5.51 %	4.73 %	1.83 %	6.16 %	4.91 %
CG3	2.66 %	4.42 %	1.01 %	3.38 %	4.20 %	0.79 %	4.75 %	5.15 %
CG4	1.93 %	3.96 %	0.99 %	2.49 %	3.67 %	0.78 %	3.27 %	4.30 %

References

Brucker P., S. Knust, 2000, "A linear programming and constraint propagation-based lower bound for the RCPSP", *European Journal of Operational Research*, Vol. 127, pp. 355 – 362

Goldfarb D., M.J. Todd, 1989, "Linear programming", in: Nemhauser G.L., A.H.G. Rinnooy Kan, and M.J. Todd (eds.), *Handbook in Operations Research and Management Science, Vol. 1: Optimization*. North-Holland, Amsterdam, pp. 73 – 170

Kolisch R., C. Schwindt, and A. Sprecher, 1999, "Benchmark instances for project scheduling problems", in: Węglarz J. (ed.), *Project Scheduling: Recent Models, Algorithms and Applications*. Kluwer, Boston, pp. 197 – 212

Mingozzi A., V. Maniezzo, S. Ricciardelli, and L. Bianco, 1998, "An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation", *Management Science*, Vol. 44, pp. 714 – 729

Neumann K., C. Schwindt, and J. Zimmermann, 2003, *Project Scheduling with Time Windows and Scarce Resources*. Springer, Berlin

Rieck J., J. Zimmermann, 2014, "Exact methods for resource leveling problems", in Schwindt C., J. Zimmermann (eds.) *Handbook on Project Management and Scheduling*. Springer, Berlin, to appear

Rieck J., J. Zimmermann, and T. Gather, 2012, "Mixed-integer linear programming for resource leveling problems", *European Journal of Operational Research*, Vol. 221, pp. 27 – 37

Two-agent scheduling problem with flowtime objective: Analysis of problem and exact method

Paz Perez-Gonzalez[1], Jose M. Framinan[1], Manuel Dios[1]

Industrial Management, University of Seville, Spain
pazperez@us.es, framinan@us.es, mdios@us.es

Keywords: single machine, two-agent scheduling problem, interfering jobs problem, flowtime.

1 Introduction

Interfering jobs problems, or multi-agent scheduling problems, consist on scheduling jobs from different sets, each one with its own objective, and competing for the same machines (Agnetis *et. al.* 2004). This is an emergent topic, and a recent review of the problem is presented by Perez-Gonzalez and Framinan (2013), from whom we adopt the notation. In this paper, we consider a single machine scheduling problem with two sets of jobs $\mathcal{J}^A$ and $\mathcal{J}^B$ ($\mathcal{J}^A \cap \mathcal{J}^B = \varnothing$), each one with n^A and n^B jobs respectively. Let $\mathcal{J} = \mathcal{J}^A \cup \mathcal{J}^B$ with $n = n^A + n^B$ jobs. Given a sequence σ formed by jobs in both sets, the completion time of job $i \in \mathcal{J}^A$ is denoted as $C_i^A(\sigma)$ and $C_{sum}^A(\sigma) = \sum_{i \in \mathcal{J}^A} C_i^A(\sigma)$ is the total flowtime of σ for jobs in $\mathcal{J}^A$. The total flowtime of jobs in $\mathcal{J}^B$ is $C_{sum}^B(\sigma)$. The objective considered here is to minimize C_{sum}^A subject to $C_{sum}^B \leq \epsilon$, and the problem is denoted $1||\epsilon(C_{sum}^A/C_{sum}^B)$. This problem is shown to be weakly NP-hard by Agnetis *et. al.* (2004), who present a Dynamic Programming (DP) algorithm with running time $\mathcal{O}(n^A n^B \epsilon)$. In this paper, we try to gain some understanding of this problem. In Section 2, we analyse their structure of solutions, as it is well-known that NP-hard problems can be easy to be solved by heuristic methods since there may be many solutions close to the optimal), and vice versa. In Section 3, we derive some specific properties of this problem and a more efficient codification of solutions, which is embedded in a Branch and Bound procedure that outperforms existing exact methods.

2 Analysis of the structure of solutions

The problem under consideration is weakly NP-hard, so it is not possible to find the optimal solution in polynomial time. However, there are strongly NP-hard problems for which is not "difficult" to find solutions close to the optimum due to its structure of solutions. For example, Perez-Gonzalez and Framinan (2009) study a strongly NP-hard scheduling problem which, in some cases, almost all solutions (99.6%) have an approximation percentage to the optimal value of the objective function less than 2%. So, finding a good solution by heuristic methods is "easy". Taillard (1990) and Armentano and Ronconi (1999) concludes that other scheduling problems are "harder" using the same approach. To the best of our knowledge, this kind of studies have been not carried out for interfering jobs problems. In our case, not all schedules are feasible so the analysis must be based on two aspects:

1. The percentage of feasible solutions. The hardness of the problem depends on the percentage of feasible solutions with respect to the total number of solutions. This ratio clearly depends on the value of ϵ. In our case, we compute ϵ as in Agnetis *et. al.* (2009), $\epsilon \in [\epsilon_{min}, \epsilon_{max}]$, for a given $\delta \in (0,1)$: $\epsilon = \epsilon_{min} + \delta(\epsilon_{max} - \epsilon_{min})$, with $\epsilon_{\min} = C_{sum}^B(\sigma_{SPT}^B \cup \sigma_{SPT}^A)$, and $\epsilon_{\max} = C_{sum}^B(\sigma_{SPT}^A \cup \sigma_{SPT}^B)$.

2. The distribution of solutions provides the distance to the optimal solution of each feasible solution obtained by complete enumeration for each instance. Unfeasible solutions are discarded. If a high proportion of feasible solutions are close to the optimal solution, the problem is considered to be "easy".

We have generated ten small instances (since the computational time to evaluate all sequences of each instance is high) for each problem combining all values of $n^A \in \{5, 10, 15\}$, $n^B \in \{5, 10, 15\}$ and $\delta \in \{0.2, 0.4, 0.6, 0.8\}$. Table 1 shows the percentage (average) of feasible solutions for the 10 instances solved for each problem. Moreover, the distributions of feasible solutions for all cases have the same shape (see Figure 1 as an example for instances of size 15×15). Regarding δ, Table 1 shows that as ϵ (δ) increases, the number of feasible solutions increases too. Figure 1 shows that as ϵ increases, the solutions are more distant to the optimal. Then, the difficulty of the problem suggests different approaches to tackle it depending on the value of ϵ, i.e.: a) Smaller values of ϵ ($\delta \in \{0.2, 0.4\}$) mean few feasible solutions, however, the distribution shows that feasible solutions in this case are close to the optimal, so any feasible solution can be a good solution. Then, the main focus is on finding feasible solutions. b) Bigger values of ϵ ($\delta \in \{0.6, 0.8\}$) mean a great percentage of feasible solutions, however, the distribution shows that feasible solutions in this case may be far from the optimal. Then, the main difficulty here is on finding good solutions (among the feasible solutions).

With respect to problem sizes, in Table 1 it can be seen that, on average, the problem seems more difficult when $n^A \leq n^B$: 5×10 (64.5704) vs 10×5 (68.4882); 5×15 (64.6340) vs 15×5 (68.7018); 10×15 (68.6646) vs 15×10 (68.7524). This conclusion is the opposite that the obtained by Agnetis *et. al.* (2009) for the weighted case $1||\epsilon(C^A_{wsum}/C^B_{wsum})$.

Table 1. Percentage of feasible solutions

		δ				
n^A	n^B	0.2	0.4	0.6	0.8	Aver.
5	5	19.2063	55.3968	84.2857	96.6270	63.8790
	10	14.9950	56.2970	88.3350	98.6547	64.5704
	15	13.1127	56.4880	89.8013	99.1338	64.6340
10	5	19.8968	63.4266	91.5018	99.1275	68.4882
	10	13.8711	65.6915	95.3331	99.8345	68.6825
	15	11.2345	66.8304	96.6568	99.9368	68.6646
15	5	18.2598	64.0473	92.9870	99.5130	68.7018
	10	11.5637	66.8426	96.6568	99.9464	68.7524
	15	8.6749	68.3683	98.0317	99.9868	68.7654

3 Codification of Solutions and Branch and Bound procedure

The classical encoding scheme used to represent a sequence for one-machine scheduling problems is the *permutation codification*, where each job $j \in \mathcal{J}$ is represented by a number, $j = 1, \ldots, n$. However, our problem has some properties that allow developing a more efficient codification. More specifically, Agnetis *et. al.* (2004) show that, in an optimal schedule, jobs in $\mathcal{J}^A$ and jobs in $\mathcal{J}^B$ follow the shortest processing time first rule (SPT). Schedules verifying this property are called *SPT schedules*. Note that the SPT rule does not apply for jobs belonging to different sets. Without loss of generality, we will assume that processing times of jobs in $\mathcal{J}^A$ and $\mathcal{J}^B$ are given in SPT order respectively. Based on

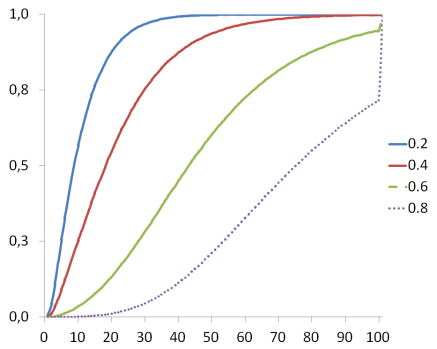


Fig. 1. Distribution of feasible solutions for $n^A \times n^B = 15 \times 15$ for different δ

this property, we can define a new encoding scheme called *binary codification*, where jobs in $\mathcal{J}^A$ are coded by zeros, and jobs in $\mathcal{J}^B$ by ones. Any schedule formed by zeros and ones represents only one SPT schedule. The first zero in the schedule is the job in $\mathcal{J}^A$ with the smallest processing time, the second is the second one in $\mathcal{J}^A$ with smallest processing time, and so on. Note that this codification reduces the search space from $n!$ possible schedules to only $\frac{(n^A+n^B)!}{n^A!n^B!}$ of SPT schedules.

Taking into account this binary codification as well as some properties of the problem that are omitted due to the lack of space, we develop a Branch and Bound (B&B) algorithm (pseudo-code in Figure 2), where UB is the upper bound used in the method, computed by the total flowtime of $\mathcal{J}^A$ for a given sequence provided by the fastest method presented by (Perez-Gonzalez and Framinan 2012). It is compared to the DP algorithm by Agnetis *et. al.* (2004) and to a MILP model of the problem using the Gurobi solver. Ten instances of sizes $n^A \in \{5, 10, 15, 20\}$ and $n^B \in \{5, 10, 15, 20\}$ are generated with random $[1, 99]$ processing times. δ has been randomly generated in the interval $[0.4, 0.6]$. Table 2 shows the average CPU time for each problem size. Note that Gurobi and DP are slower (not being able to find the optimal solution in less than 24 hours per instance for the largest size), while the B&B is faster for all instances except for the smallest sizes (5×5 and 5×10).

References

Agnetis, A. and De Pascale, G. and Pacciarelli, D. A lagrangian approach to single-machine scheduling problems with two competing agents *Journal of Scheduling*, 51(4):401-415, 2009.

Agnetis, A. and Mirchandani, P. B. and Pacciarelli, D. and Pacifici, A. Scheduling problems with two competing agents *Operations Research*, 52(2):229-242, 2004.

Armentano, V. A. and Ronconi, D. P. Tabu search for total tardiness minimization in flowshop scheduling problems *Computers & Operations Research*, 26(3):219-235, 1999.

P. Perez-Gonzalez and J. M. Framinan. Scheduling permutation flowshops with initial availability constraint: Analysis of solutions and constructive heuristics. *Computers & Operations Research*, 36(10):2866-2876, 2009.

P. Perez-Gonzalez and J. M. Framinan. A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems *European Journal of Operational Research*, 235(1):1-16, 2014.

P. Perez-Gonzalez and J. M. Framinan. Approximate algorithms for one-machine scheduling with interfering jobs *XXV European Conference On Operational Research*, Vilnius, Lituania, 2012.

Taillard, E. D. Some efficient heuristic methods for the flow shop sequencing problem *European Journal of Operational Research*, 47(1):65-74, 1990.

```
procedure B&B
    % STEP 1: Initial Solution and Upper Bound
    σ := Initial_Improved_Solution;
    UB = C^A_sum(σ);
    σ = ∅; σ^A = [0, ..., 0]; σ^B = [1, ..., 1];
    % STEP 2: UpdateTree(σ)
    while σ ≠ ∅ do
        if σ is complete;
            if C^A_sum(σ) < UB
                UB = C^A_sum(σ);
            end if
        else
            LB_A = C^A_sum(σ ⋃ σ^A)
            LB_B = C^B_sum(σ ⋃ σ^B)
            if σ^A ≠ ∅
                if LB_B ≤ ϵ & LB_A ≤ UB
                    σ = σ ⋃ 0; σ^A = σ − 0;
                    UpdateTree(σ)
                end if
            end if
            if σ^B ≠ ∅
                if LB_B ≤ ϵ & LB_A ≤ UB
                    σ = σ ⋃ 1; σ^B = σ − 1 ;
                    UpdateTree(σ)
                end if
            end if
        end if
    end while
```

Fig. 2. Pseudo-code of B&B

Table 2. Average CPU time (seconds)

sizes	DP	Grb	BB
5×5	0.0004	0.0478	0.0045
5×10	0.0013	0.0744	0.0060
5×15	0.0058	0.2234	0.0030
5×20	0.0201	0.4674	0.0075
10×5	0.0016	0.0711	0.0030
10×10	0.0857	0.5811	0.0120
10×15	1.5313	5.3496	0.0369
10×20	15.2514	31.2691	0.2383
15×5	0.0107	0.1795	0.0030
15×10	1.8997	5.7652	0.0324
15×15	83.5664	170.3985	1.1713
15×20	1875.0860	2629.9187	14.6230
20×5	0.0454	0.4195	0.0060
20×10	23.1112	39.2189	0.1683
20×15	2236.2690	2330.0252	11.2597
20×20	> 86400	> 86400	439.5048
Average	**282.4591**	**347.6006**	**29.1925**

A Flow-Based Tabu Search Algorithm for the RCPSP with Transfer Times

Jens Poppenborg[1], Sigrid Knust[2]

[1] Clausthal University of Technology, 38678 Clausthal-Zellerfeld, Germany
jens.poppenborg@tu-clausthal.de
[2] University of Osnabrück, 49069 Osnabrück, Germany
sigrid@informatik.uni-osnabrueck.de

Keywords: RCPSP, transfer times, tabu search

1 Introduction

In this work we propose a tabu search algorithm for the resource-constrained project scheduling problem (RCPSP) with transfer times. This problem has been introduced in Krüger and Scholl (2009) and Krüger and Scholl (2010) where an IP formulation and priority-based heuristics were developed. In the RCPSP with transfer times a set $\mathcal{R}$ consisting of r renewable resources $k = 1, \ldots, r$ with limited capacities $R_k \geq 0$ as well as a set V consisting of n activities $i = 1, \ldots, n$ with processing times $p_i \geq 0$ and resource requirements $r_{ik} \geq 0$ for $k \in \mathcal{R}$ are given. Furthermore, a dummy source activity 0 as well as a dummy sink activity $n+1$ with processing times $p_0 = p_{n+1} = 0$ and resource requirements $r_{0k} = r_{n+1,k} = R_k$ for all $k \in \mathcal{R}$ are introduced. It is assumed that all resource units are initially located at the dummy source activity 0 and have to be collected by the dummy sink activity $n+1$ at the end of the project. In the following, let $V_{all} = \{0, 1, \ldots, n, n+1\}$, $V_0 = \{0, 1, \ldots, n\}$ and $V_* = \{1, \ldots, n, n+1\}$. Furthermore, precedence constraints $A = \{(i,j) \,|\, i,j \in V_{all}\}$ between pairs of activities $i, j \in V, i \neq j$ may exist requiring that activity j can only start after activity i has been completed. Additionally, sequence- and resource-dependent transfer (or changeover) times $\Delta_{ijk} \geq 0$ are given for all $i, j \in V_{all}$ and $k \in \mathcal{R}$ denoting the amount of time required to transfer resource k from activity i to activity j (e.g. for transporting a crane from one construction site to another or for cleaning a resource between producing different products). The special case $\Delta_{ijk} = 0$ corresponds to the classical RCPSP. All parameters are assumed to be integer. The objective is to schedule all activities $j \in V_*$ without preemption with respect to the given precedence and resource constraints. Additionally, if some units of resource $k \in \mathcal{R}$ are transferred from activity $i \in V_0$ to activity j, the transfer time Δ_{ijk} has to be observed between the completion time $C_i = S_i + p_i$ of activity i and the starting time S_j of activity j, i.e. $S_j \geq C_i + \Delta_{ijk}$ has to hold. We consider the makespan objective $C_{\max}$, i.e. we minimize the time required to complete all activities of a given project which is determined by the completion time C_{n+1} of the dummy sink activity $n+1$.

2 Solution Representation

For the classical RCPSP different solution representations have been proposed. Heuristics often operate on an indirect representation based on activity lists. These activity lists are then transformed into a schedule by using a schedule generation scheme that schedules the activities in the order specified by the activity list. For the RCPSP with resource transfers (where each resource transfer needs a certain time), a drawback of this representation is that it does not represent from which activities resource units are transferred to other activities. Instead, resource transfers have to be selected separately, for example, based on

a priority rule. Due to this drawback, we use an alternative solution representation based on resource flows (cf. Fortemps and Hapke (1997) and Artigues and Roubellat (2000)). The disadvantage of a larger solution space is compensated by the fact that with this representation it is ensured that the solution space always contains an optimal schedule.

Resource flows are an extension of the disjunctive graph model for shop scheduling problems. While a disjunctive graph represents the sequence in which disjunctive machines are used to process operations in a shop scheduling problem, this representation is extended for the RCPSP to also denote the amounts of resources $k \in \mathcal{R}$ transferred from an activity $i \in V_0$ to another activity $j \in V_*$. In the following, f_{ijk} denotes the amount of resource $k \in \mathcal{R}$ transferred from activity $i \in V_0$ to activity $j \in V_*$. All resource transfers f_{ijk} of a resource $k \in \mathcal{R}$ between activities $i \in V_0$ and $j \in V_*$ are then collected in a resource flow $\mathcal{F}_k$ while a set of resource flows $\mathcal{F}_k$ for all resources $k \in \mathcal{R}$ is denoted by $\mathcal{F}$.

A feasible resource flow $\mathcal{F}$ is defined as a flow that satisfies the following conditions. First of all, all resource units are initially located at the dummy source activity 0 and have to be collected by the dummy sink activity $n+1$ at the end of the project. Furthermore, for each activity $i \in V$ and each resource $k \in \mathcal{R}$, the amount of all incoming resource units as well as the amount of all outgoing resource units have to be equal to the resource requirement r_{ik} (flow conservation constraints). Finally, the resource flow has to be acyclic, i.e. no activity sends (directly or indirectly) resource units to itself and the flow must observe the given precedence constraints (i.e. for $(i,j) \in A$ activity j may not send (directly or indirectly) resource units to activity i).

A feasible resource flow $\mathcal{F}$ can be represented by a (multi-)graph where each activity $i \in V$ is modelled as a node and each resource transfer $f_{ijk} > 0$ of a resource $k \in \mathcal{R}$ from activity $i \in V_0$ to activity $j \in V_*$ is represented by a directed arc $(i,j)_k$ from node i to node j. Similar to the disjunctive graph model for shop scheduling problems, an AON-flow network (cf. Artigues *et. al.* (2003)) incorporates the activity-on-node network representing the precedence constraints $(i,j) \in A$ as well as the graph representing a resource flow $\mathcal{F}$. In this graph, each precedence arc $(i,j) \in A$ is weighted with the processing time p_i of activity i, each flow arc $(i,j)_k$ satisfying $f_{ijk} > 0$ is weighted with the value $p_i + \Delta_{ijk}$.

In the same way as for the classical RCPSP, for each extended AON-flow network it is possible to generate an earliest start schedule (in which all activities start as early as possible with respect to the given precedence constraints $(i,j) \in A$ and the resource transfers f_{ijk} from the given resource flow $\mathcal{F}$) by longest path calculations. Here, the length of the arc (i,j) equals $\max_{\{k|f_{ijk}>0\}} \{p_i + \Delta_{ijk}\}$ for a flow arc and p_i for a precedence arc.

Theorem 1. *For the RCPSP with transfer times the set of schedules represented by resource flows always contains an optimal schedule.*

3 A Tabu Search Algorithm

Based on the resource flow solution representation, we use neighborhoods defined by parallel and serial modifications as suggested in Fortemps and Hapke (1997). However, in Fortemps and Hapke (1997) the ideas are only sketched and no further results are reported (neither experimental not theoretical).

First of all, a neighborhood $\mathcal{N}_{reroute}$ is introduced based on the parallel modification. For a modification in this neighborhood, two arcs $(i,j)_k$ and $(u,v)_k$ between activities $i,u \in V_0$ and $j,v \in V_*$ representing resource transfers $f_{ijk} > 0$ and $f_{uvk} > 0$ of a resource $k \in \mathcal{R}$ are selected in a resource flow $\mathcal{F}$ such that no directed path exists from either activity j to activity u or from activity v to activity i. Then, an amount of $q \in \{1,\ldots,\min\{f_{ijk}, f_{uvk}\}\}$ units of resource k is rerouted from activity i to activity v as well as from activity u to

activity j. This results in a resource flow $\mathcal{F}'$ with the modified resource transfers $f'_{ijk} = f_{ijk} - q$, $f'_{uvk} = f_{uvk} - q$, $f'_{ivk} = f_{ivk} + q$, and $f'_{vjk} = f_{vjk} + q$ between the activities.

Next, a neighborhood $\mathcal{N}_{reverse}$ is introduced based on the serial modification. For a modification in this neighborhood, two activities $i \in V$ and $j \in V$ are selected in a resource flow $\mathcal{F}$ such that at least one arc $(i,j)_k$ for a resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ exists between these activities and such that $(i,j) \notin A$ and no other path exists from activity i to activity j via other activities $h \in V$ in the AON-flow network. Furthermore, groups U_k with $f_{uik} > 0$ for all activities $u \in U_k$ as well as groups V_k with $f_{jvk} > 0$ for all activities $v \in V_k$ have to be selected for each resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ such that $\sum_{u \in U_k} f_{uik} \geq f_{ijk}$ and $\sum_{v \in V_k} f_{jvk} \geq f_{ijk}$ holds. Then, a serial modification first reverses the direction of all arcs $(i,j)_k$ for resources $k \in \mathcal{R}$ with $f_{ijk} > 0$ between activities i and j such that an amount of $f'_{jik} = f_{ijk}$ units of resource $k \in \mathcal{R}$ is transferred from activity j to activity i in the resulting resource flow $\mathcal{F}'$. Additionally, f_{ijk} units of each resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ have to be redirected from activities $u \in U_k$ to activity j as well as from activity i to activities $v \in V_k$ in order to maintain flow conservation in the resulting resource flow $\mathcal{F}'$.

It can be shown that neither the neighborhood $\mathcal{N}_{reroute}$ nor the neighborhood $\mathcal{N}_{reverse}$ alone is connected (and even not opt-connected) for the RCPSP (and hence also not for the more general situation with transfer times). However, the larger neighborhood $\mathcal{N}_1 = \mathcal{N}_{reroute} \cup \mathcal{N}_{reverse}$ is connected.

Theorem 2. *For the RCPSP with transfer times the neighborhood $\mathcal{N}_1$ is connected.*

Due to the large size of this neighborhood, we reduce both neighborhoods. First of all, similar to Fortemps and Hapke (1997), we introduce the neighborhood $\mathcal{N}^{ca}_{reverse}$ in which reverse moves are limited to critical activities $i, j \in V$. It can be shown that these neighbors are always feasible. The size of this reduced neighborhood is limited to $\mathcal{O}(n)$. Next, we introduce a neighborhood $\mathcal{N}^{max,ca}_{reroute}$ in which one critical arc $(i,j)_k$ as well as an arbitrary arc $(u,v)_k$ representing resource transfers $f_{ijk} > 0$ and $f_{uvk} > 0$ are selected instead of two arbitrary arcs. Additionally, similar to Fortemps and Hapke (1997), always the maximal amount of $q = \min\{f_{ijk}, f_{uvk}\}$ units of resource k is rerouted. The size of this neighborhood is then limited to $\mathcal{O}(rn^3)$.

In order to evaluate the solution representation as well as the neighborhoods described above, we implemented a tabu search algorithm. An initial solution is generated by a parallel schedule generation scheme as it has been described in Krüger and Scholl (2009). In each iteration of the tabu search, a solution is chosen from the neighborhood $\mathcal{N}_2 = \mathcal{N}^{max,ca}_{reroute} \cup \mathcal{N}^{ca}_{reverse}$. Here, a priority rule ES is used to select sets U_k for a reverse modification according to non-decreasing earliest arrival times of the resource units at the receiving activity while a priority rule TT is used to select sets V_k according to non-decreasing transfer times between the activities. It should be noted that only modifications that are not tabu are evaluated. If an improving solution with an objective function value smaller than the objective function value of the current solution is found, this solution is selected immediately. Otherwise, the best non-tabu solution from the neighborhood is chosen. Afterwards, the tabu list is updated based on the selected modification.

4 Computational Results

In order to evaluate the tabu search algorithm presented above, we implemented it in Java and tested it on a computer with an Intel Core 2 Quad Q6600 (2.4 GHz) processor with 4 GB RAM. We used this algorithm to solve the problem instances for the classical RCPSP consisting of 30, 60, 90, and 120 activities as they have been generated by Kolisch and Sprecher (1997). Furthermore, we solve the problem instances for the RCPSP with

transfer times consisting of 30 and 60 activities generated by Krüger and Scholl (2009). The tabu search algorithm is stopped after at most 10 000 iterations.

Computational results for the classical RCPSP instances are presented in the left part of Table 1. In the column Δ_{av} we report the average deviations (in %) from the optimal solutions for the problem instances with 30 activities as well as the average deviations from the critical path lower bound LB_0 for the problem instances with 60, 90, and 120 activities. Furthermore, the average computation times t_{av} (in seconds) are listed. Finally, in the column Δ_{UB} we present the average deviations of the currently best known heuristic solutions reported at `http://www.om-db.wi.tum.de/psplib/main.html` from LB_0.

Results for instances with transfer times are presented in the right part of Table 1. Since the optimal values for all these instances are known, we report the average deviations Δ_{av} from the optimal solutions. Additionally, in the column Δ_{GA} analogous deviations are shown for the best results of the genetic algorithm by Krüger (2009) stopping after 5 000 generated schedules.

Table 1. Computational results for classical RCPSP instances and instances with transfer times.

n	Δ_{av}	t_{av}	Δ_{UB}
30	0.66	7.8	–
60	13.02	74.1	10.37
90	12.79	250.2	9.48
120	38.32	1298.0	29.18

n	Δ_{av}	t_{av}	Δ_{GA}
30	1.27	8.8	0.16
60	0.98	31.8	0.38

It can be seen that the tabu search algorithm obtains ample solutions in an acceptable amount of time. For the classical RCPSP it is clearly outperformed by the state-of-the-art heuristics especially designed for this special situation. This may be explained by the fact that the solution space of the flow based representation is quite large and it is difficult to guide the neighborhood search to good solutions. However, the flow based approach offers a promising adaptability for further extensions of the RCPSP, in particular the RCPSP with higher-tier resource transfers as it has been introduced in Krüger and Scholl (2010).

References

Artigues, C. and F. Roubellat, 2000, "A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes", *Eur. J. of Oper. Res.* 127, pp. 297–316.

Artigues, C., P. Michelon and S. Reusser, 2003, "Insertion techniques for static and dynamic resource-constrained project scheduling", *Eur. J. of Oper. Res.* 149, pp. 249–267.

Fortemps, P. and M. Hapke, 1997, "On the disjunctive graph for project scheduling", *Found. of Comp. and Dec. Sc.* 22, pp. 195–209.

Kolisch and Sprecher, 1997, "PSPLIB - A project scheduling problem library", *Eur. J. of Oper. Res.* 96, pp. 205–216.

Krüger, D. 2009, "Multi-Project Scheduling with Resource Transfers", PhD dissertation, University of Jena.

Krüger, D. and A. Scholl, 2009, "A heuristic solution framework for the resource constrained (multi-) project scheduling problem with sequence-dependent transfer times", *Eur. J. of Oper. Res.* 197, pp. 492–508.

Krüger, D. and A. Scholl, 2010, "Managing and modelling general resource transfers in (multi-) project scheduling", *OR Spectrum* 32, pp. 369–394.

Ant Algorithms for a Truck Loading Problem with Multiple Destinations

Jean Respen[1] and Nicolas Zufferey[1]

GSEM, University of Geneva, Switzerland, jean.respen@unige.ch, n.zufferey@unige.ch

Keywords: truck loading, bin-packing, evolutionary algorithm.

1 Introduction

The French car manufacturer *Renault* is daily facing a NP-hard combinatorial optimization problem – called here the *Renault* truck loading problem ($RTLP$) – where items need to be placed in a truck while satisfying different constraints. More than a thousand trucks are daily considered, which have to deliver goods to several car factories. As a single truck can deliver goods to different delivery points, classes of items have been defined, where a class is associated with a delivery point. Each problem instance contains the size of the truck (in millimeters) and the various sizes of all the items that must fit in (in millimeters). The heights of the items can be ignored as they rely on complex factory constraints which are supposed to be already satisfied. At first sight, this problem seems related to a strip-packing 2D problem with rotation, which has been already covered by many research papers (Lodi *et. al.* 2002). New features are proposed by *Renault* in $RTLP$: different *classes* of items, and a significant *number* of items per truck in conjunction with a large *standard deviation* of the sizes of the items. Such elements make the considered problem more complex, but more relevant to modern and realistic issues. In this paper, we propose to tackle $RTLP$ with *Constructive Ant Systems* (CAS), which are evolutionary population-based metaheuristics. A good survey on ant algorithms can be found in (Dorigo *et. al.* 2006).
Based on (Respen and Zufferey 2013), the problem and the solution space structure are formally described in Section 2, where an efficient decoding greedy algorithm is also designed. In Section 3 are proposed different ant algorithms for $RTLP$. Numerical experiments are reported and discussed in Section 4.

2 Description of the problem and solution space structure

$RTLP$ can be formally described as follows. A number n of rectangular items have to be placed in a truck (of width W_t and length L_t), and 90° rotations of items are allowed. For each item i, we know its width w_i, its length l_i, its initial orientation, and its class C_j (where $j \in \{1, \ldots, m\}$ with $m \leq n$). In addition, the classes must be placed in an increasing fashion from the front to the rear of the truck. More precisely, the ordinate of the origin item which belongs to class C_i (label 1 on Figure 1) must be strictly smaller than the ordinate of the extremity of any item of class C_{i+1} (label 2 on Figure 1). The goal consists in minimizing the ordinate f of the extremity item (the closest one to the rear) of class C_m (label 3 on Figure 1).
To tackle bin-packing problems, one can work either with *direct coded* solutions or *indirect coded* solutions. A direct coded solution directly represents a real loading of the items in

the truck, which means that the position of each item has to be continuously known. Then, if an item is added to or removed from the truck, the new position of the loaded items is very hard to recompute, which is a major drawback. Working with indirect coded solutions require the use of a *decoding* algorithm to built the associated direct coded solution and to get its value. Such an approach has the main advantage of being very flexible when adding (resp. removing) an item to (resp. from) the solution. Therefore, a decision has been made to work with *indirect coded* solutions. More formally, an indirect coded solution s is a sequence of elements. To build a direct coded solution $\hat{s}$ and compute its value $f(s)$, a *decoding greedy algorithm* (DGA) is performed on the indirect coded solution s. Component i of the indirect coded solution s takes the form $s_i = (ID_i, \mathcal{C}_i)$, where $ID \in \{1, \ldots, n\}$, and $\mathcal{C} \in \{1, \ldots, m\}$. DGA decodes the vector s into a real solution $\hat{s}$ (i.e. a true loading) by inserting in $\hat{s}$ the items from s in a $FIFO$ order. At each step, DGA pops the next item i of s and orients it greedily (i.e. by minimizing the augmentation of the loading length). The complexity of DGA is $\mathcal{O}(n)$. For instances with $n = 60$, DGA requires less than 0.1 second (on the used computer) to decode a solution into a real loading. In summary, DGA is requested to built a direct coded solution (i.e. a loading of the truck) from an indirect coded solution (i.e. a loading sequence of items) and to compute the length of the obtained loading (i.e. the value of the solution).
Preliminary tests showed that the performance of DGA is rather poor. Indeed, at each iteration, DGA orients the involved item without having any visibility on the next items to insert. To reduce this drawback, the following *look-ahead* process is proposed. At each iteration, DGA evaluates the orientation of the involved unloaded item (say i) as follows. DGA examines the insertion of the next σ (parameter tuned to 3) insertions subsequent to the insertion of item i (i.e. a total number of $\sigma + 1$ items are tested). DGA tries each possible orientation (i.e. 90°-rotated or not-rotated) for each of the $\sigma + 1$ items in order to minimize the augmentation of the resulting loading length (in other words, $2^{(\sigma+1)}$ options are evaluated). The orientation of item i is then the one associated with the best option. For instances with $n = 60$, DGA requires less than 0.5 second (on the used computer) to decode a solution into a real loading.

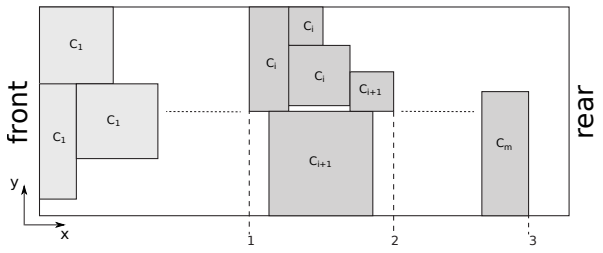


Fig. 1. A possible solution for $RTLP$

3 Ant algorithms for $RTLP$

We propose three different constructive ant algorithms to tackle $RTLP$ (denoted $CAS(1)$, $CAS(2)$ and $CAS(3)$), where each ant is a constructive heuristic able to build an indirect coded solution, and thus a loading. N (parameter tuned to 10) ants are used at each generation. For each ant, at iteration i of its construction process, let $s = (s_1, s_2, \ldots, s_{i-1})$

be the partial indirect coded solution containing $i-1$ items. As in classical ant algorithms, the selection of the next move (corresponding here as the next item to load) is based on the *greedy force* (defined as the short term profit) and the *trail system* (which is a central memory based on the history of the search, allowing ants to exchange information). In the proposed ant algorithms, the GF's and the TR's are normalized within interval [0,1] in order to better control these two types of information.
The greedy force $GF(i)$ of an item i is given by $GF(i) = UB - DGA(s + \{i\})$, where $UB = 4 \cdot L_t$ is an upper bound of the loading. We can see that the larger $GF(i)$ is, the shorter is the loading $s + \{i\}$, which is consistent with the notion of short term profit.
To define the trail $TR(i)$ of item i, we first define the *attractiveness* $Attract(j,k)$ between two items j and k in an indirect coded solution s as $n - dist(j,k)$, where $dist(j,k)$ is defined as the number of items between j and k in s (but $dist(j,k) = 0$ if k appears before j in s). At the end of each generation (i.e. when each ant of the population has provided a solution), each trail $TR(j,k)$ is updated as follows: $TR(j,k) = \rho \cdot TR(j,k) + (1-\rho) \cdot \Delta TR(j,k)$, where $\rho \in]0,1[$ is an *evaporation* coefficient (fixed to 0.9, as in most of the ant algorithms) and $\Delta TR(j,k)$ is a *reinforcement* term. $\Delta TR(j,k)$ is defined as the average attractiveness between j and k in the $b\%$ (parameter tuned to 50%) best solutions of the current generation. The trail value $TR(i)$ of item i can now be defined as $TR(i) = \sum_{j \in s} w_{ji} \cdot TR(j,i)$, where w_{ji} is a weight defined as $index(j)/index(i)$, where $index(j)$ is for example three is item j is the 3rd component of s. These weights allows to give more importance to pairs of items which are closer in s (according to the distance function $dist$).
In $CAS(1)$, when constructing an indirect coded solution, each ant selects the next item i to load – in the set Ω of non already inserted items – using the standard probability function $prob$ defined as in Equation (1), where α and β are parameters. Interestingly, preliminary experiments showed that better results are obtained with parameter α (tuned to 0.5) smaller than parameter β (tuned to 2). It means that more importance should be given to the greedy force rather than to the trail.

$$prob(i) = \frac{GF(i)^\alpha \cdot TR(i)^\beta}{\sum\limits_{j \in \Omega} GF(j)^\alpha \cdot TR(j)^\beta} \quad (1)$$

In $CAS(2)$, an ant chooses the next item i using Equation (1) with probability p (parameter tuned to 0.35), but the item which maximizes $GF(j) \cdot TR(j)$ otherwise (i.e. with probability $(1-p)$). In other words, $CAS(2)$ is more aggressive than $CAS(1)$ as the usual tradeoff between the greedy forces and the trails only occurs with probability p.
In $CAS(3)$, each ant selects the next item i as follows. First, it generates the set A with the $q\%$ (parameter tuned to 0.75) largest TR values. Then among the set A, it selects the item with the largest GF value (ties are broken randomly). In contrast with $CAS(1)$ and $CAS(2)$, the greedy forces and the trails are *successively* used in order to select the next item (instead of *jointly*).

4 Results

In order to better benchmark the results, we compare the ant algorithms with an exhaustive greedy method EG. EG builds an indirect coded solution s from scratch, and at each step greedily inserts the next item in s (with the use of the *look-ahead* process). We consider a set of 30 real benchmark instances provided by *Renault*. Tests were performed on an Intel Quad-core i7 @ 3.4 GHz with 8 GB DDR3 of RAM memory, with a time limit T of 900 seconds. In order to have fair comparisons, EG is restarted as long as T is not reached,

whereas the CAS's results are averaged over 10 runs. Table 1 presents the results of the different algorithms. For each instance ID are first given the number n of items and the number m of classes. Column $f^\star$ indicates the best objective function value returned by any of the considered algorithms. The second column EG shows the percentage gap between the best solution value returned by EG and $f^\star$. The remaining columns show the same information for the ant algorithms. The last row indicates the average gap of each method. We remark that all the CAS methods outperform EG, which means that the trail system is relevant. $CAS(2)$ outperforms $CAS(1)$, which shows that an aggressive selection process seems to be more interesting than the use of Equation (1). As the best methods is $CAS(3)$, it seems that a sequential use of the trails and the greedy forces in the selection process of an ant seems to be more efficient than the joint use of these quantities (as in $CAS(1)$, $CAS(2)$, and most of the state-of-the-art ant algorithms). This confirms the observations of (Zufferey 2012) for the famous graph coloring problem.

Table 1. Obtained results

ID	n	m	$f^\star$	EG	CAS**(1)**	CAS**(2)**	CAS**(3)**
1	23	1	12970	1.62%	0.80%	1.63%	0.13%
2	25	1	13226	2.07%	0.70%	0.64%	1.48%
3	24	1	12950	2.59%	1.06%	1.38%	0.70%
4	25	1	13170	2.75%	1.32%	1.02%	2.37%
5	26	1	13470	0.07%	1.02%	0.80%	1.02%
6	20	2	14000	2.07%	2.07%	1.73%	1.77%
7	23	1	12980	4.21%	2.74%	2.24%	1.26%
8	25	1	14288	3.26%	1.95%	2.31%	2.37%
9	18	4	13369	0.44%	3.55%	3.40%	4.50%
10	23	3	13068	3.48%	3.92%	3.46%	4.02%
11	20	2	13560	2.06%	1.79%	1.92%	1.89%
12	17	3	12992	2.42%	0.81%	0.88%	0.80%
13	25	1	13470	1.97%	0.93%	1.23%	0.54%
14	20	2	13240	3.55%	3.50%	3.74%	3.44%
15	20	4	13685	2.26%	2.83%	3.27%	2.99%
16	24	1	13070	3.21%	1.71%	2.10%	0.88%
17	23	4	13078	2.03%	1.43%	1.05%	2.09%
18	24	1	13380	4.56%	1.45%	2.27%	0.68%
19	24	1	13380	4.56%	1.45%	2.27%	0.68%
20	23	1	13070	5.05%	2.41%	2.28%	1.29%
21	25	1	13146	2.84%	1.64%	1.44%	2.03%
22	25	1	13470	1.60%	1.04%	1.00%	0.09%
23	24	1	13380	4.04%	1.34%	1.75%	0.97%
24	18	2	11640	2.41%	2.72%	1.67%	1.86%
25	23	1	12550	1.20%	1.87%	1.16%	1.79%
26	19	2	12220	1.06%	1.91%	1.48%	1.69%
27	23	1	13250	5.86%	1.35%	2.08%	0.77%
28	25	1	13416	1.97%	1.66%	1.14%	0.64%
29	20	1	13500	10.74%	5.13%	4.02%	4.79%
30	25	1	13196	3.80%	1.93%	1.07%	2.10%
AVG				**2.99%**	**1.93%**	**1.88%**	**1.72%**

References

Dorigo, M., Birattari, M., and Stuetzle, T., 2006, "Ant colony optimization - artificial ants as a computational intelligence technique", *IEEE Computational Intelligence Magazine*, Vol. 1, pp. 28-39.

Lodi, A., Martello, S., and Monaci, M., 2002, "Two-dimensional packing problems: A survey", *European Journal of Operational Research*, Vol. 141, pp. 241-252.

Respen, J., Zufferey, N., 2013, "A Renault truck loading problem: from benchmarking to improvements", *Proceedings of the 14th EU/ME Workshop, Hamburg, Germany. February 28 to March 1.*, pp. 79-84.

Zufferey, N., "Optimization by ant algorithms: Possible roles for an individual ant", *Optimization Letters*, Vol. 6 (5), pp. 963-973.

A multi-start procedure for underground mine scheduling

J. Rieck, M. Schulze, and J. Zimmermann

Clausthal University of Technology, Germany
{julia.rieck, marco.schulze, juergen.zimmermann}@tu-clausthal.de

Keywords: Underground mine scheduling, Machine scheduling, Multi-start procedure.

1 Problem description

The paper addresses the short-term *underground mine production scheduling problem* that consists in specifying the sequence in which blocks should be removed from the mine. The aim is to minimize the makespan subject to a variety of constraints that are related to the mining extraction sequence, resource capacities, and safety-related conditions. The interaction of constraints lead to synergy effects resulting in an efficient utilization of resources (machines). A comprehensive survey on problems and optimization methods in mining is given by Newman *et al.* (2010).

The extraction of the considered German potash mine is done by the room-and-pillar method. Hence, the mined material is extracted across a horizontal plane while leaving pillars of untouched material to support the roof of the mine. Thus, open areas (rooms) emerge between pillars and a grid-like pattern of rooms and pillars is formed. The excavation of potash is usually performed by drilling and blasting. This kind of underground mining is characterized by nine consecutive steps (operations): (1) scaling the roof, (2) removing the scaled material, (3) bolting the roof with expansion-shell bolts, (4) drilling large diameter boreholes, (5) removing the drilled material, (6) drilling blast holes, (7) filling blast holes with explosive substances, (8) blasting, and (9) transporting broken material to a crusher. The execution of all steps can be treated as a "production cycle". For each step, except for (8) blasting, one special mobile machine is required, i.e. (1) requires scaling machines, (2) small loaders, (3) anchor drilling trucks etc. In order to excavate one block of a certain underground location, e.g., block j_6 at location a_2 in Fig. 1, it is necessary that all steps of the preceding production cycle have been finished, i.e., the preceding block j_5 is removed.

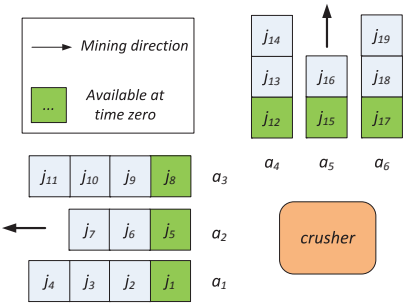


Fig. 1. Mining region: Six underground locations (a_1–a_6) and 2–4 blocks per underground location (j_1–j_{19})

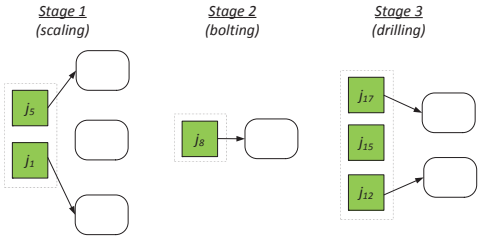


Fig. 2. Planning situation: Assignment of jobs (blocks) to machines on different stages

After the completion of step (1), the remaining operations ought to be finished within a certain time limit $\tau \geq 0$. If an operation cannot be finished within τ time units, a safety precaution is needed in which the roof is scaled once more.[1] When the scaling process is then finished, the next operation of the original production cycle is continued.

The described excavation process represents a manufacturing environment with eight stages (steps (1)–(7) and (9) of the production cycle). Each stage k consists of $M^{(k)}$ machines and the jobs (blocks) must visit the stages consecutively. Figure 2 shows a planning situation at a specific point in time, where six underground locations are in progress and five of the corresponding six first jobs are assigned to idle machines on stages (1)–(3). The problem can be identified as a *hybrid flow shop scheduling problem* (e.g., Ruiz and Vázquez-Rodríguez 2010) with additional underground mining restrictions.

2 Multi-start procedure

In Schulze and Zimmermann (2011) a mathematical formulation for the described underground mine scheduling problem is given that can be used to solve small-scale instances with up to 25 jobs. In what follows, we will present a priority rule-based construction heuristic that is developed in order to solve medium- and large-scale instances heuristically. The underlying schedule generation scheme is shown in Fig. 3 and is adapted from the classical parallel schedule generation scheme (Kolisch 1996).

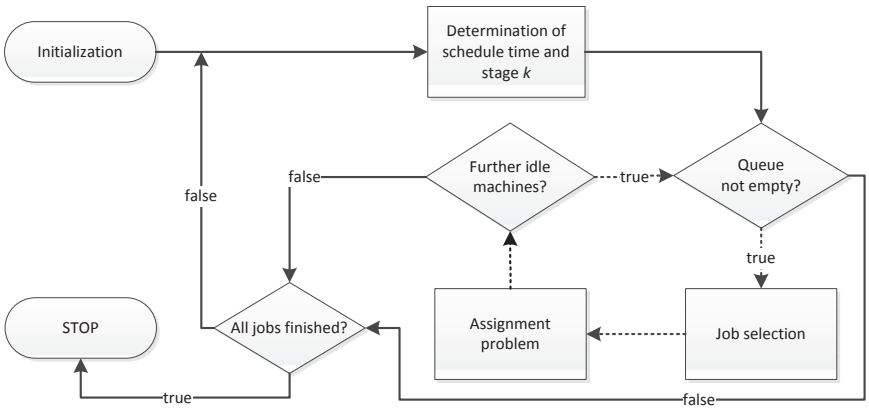


Fig. 3. Flow chart of the priority rule-based construction heuristic

All jobs that have to be processed at a certain stage are inserted in an artificial queue in front of the stage (cf. Fig. 2). Then, the *initialization* comprises the insertion of the first jobs (that are available at time zero) of the underground locations into the corresponding queues depending on the current mining progress. In the main part of the procedure, the *schedule time* $t \geq 0$ is determined, i.e., the earliest point in time at which a machine, say for example machine m on stage k, is available for use.[2] The queue Q_k in front of stage k contains all unscheduled jobs that are candidates for scheduling. In case the queue is not empty, a job is selected. The *job selection* chooses job j with the best priority-value. In the *assignment problem*, job j is assigned to machine m (if several machines are available on k the "fastest" idle machine is used). With the assignment, the completion time $C_{jk} \geq 0$ of

[1] The problem in which jobs can visit each stage several times is called re-entry or recirculation, cf. Pinedo (2008) or Kim and Lee (2009).

[2] If several stages have the same schedule time, the stage with the smallest index is selected.

job j on stage k is clearly recognizable. If k is the last stage (i.e., $k = 8$), job j is set as finished and its possible successor in the underground location is initiated. If $k < 8$ and time limit τ is not exceeded, job j follows the steps of the production cycle, otherwise j returns to step (1). The *job selection* and the *assignment problem* are solved on stage k as long as further idle machines exist and queue Q_k is not empty (the respective loop is depicted by dotted arrows). The algorithm terminates if all jobs are finished. The construction heuristic is embedded in a multi-start procedure that generates different solutions, where priority-values are employed directly in order to determine selection probabilities for jobs (Kolisch 1996).

Within the *assignment problem*, we also used a more sophisticated method for the case that only a single machine is idle at stage k in time interval $[t, t + \delta]$. At that point, the procedure returns to the *job selection* in order to search for a job $j' \neq j$, where $j' \notin Q_k$, j' will arrive at k within $\delta \geq 0$ time units, and its priority-value is better than the one of j. If such a job can be found, the current machine is blocked for j'. Consequently, job j has to wait and its start time is delayed.

3 Experimental study

The computational tests have been performed on instances that were derived from real-world data. The instances comprise 60 and 120 jobs and consist of 10 and 15 underground locations, respectively. In a first setting, the number of jobs per underground location is always *identical*, whereas the number of jobs in the second setting (*average*) varied between a predefined range. For each setting, five instances are generated, where the processing times are randomly determined between stage-job-related bounds and the progress of the first jobs is generated uniformly from the set {stage (1), ..., stage (8)}. Regarding the number of parallel machines at each stage, we distinguish between three cases: each stage contains exactly one machine, exactly two machines, or the number of machines is determined randomly between one and three. In total, we investigated $2 \cdot 2 \cdot 5 \cdot 3 = 60$ instances.

Preliminary tests have shown that the use of expedient priority rules (e.g., first-in first-out, shortest processing time first) in the course of our multi-start procedure leads to similar results. This is due to the fact that the number of jobs per queue is relatively small; it is up to 10 (15) for instances with 60 (120) jobs. However, we identified the *most-work-remaining* (MWR) rule as an adequate rule and used it in our experiments. The MWR-value of job j is defined as the total time it takes to process j and all its succeeding jobs in the respective underground location, assuming that each job is always processed on the fastest machine. We generated 10,000 solutions for each instance, where we defined δ to be equal to 0%, 20%, or 30% of the average processing time of all jobs. The procedure was implemented in C++ under Windows 7 and compiled with Microsoft Visual Studio 2010. Table 1 shows the results. In column "#Opt" the number of optimal solutions found (at most 5 in each cell) is given; for these instances the lower bound is equal to the heuristic solution. Column "∅Gap" displays the average gap [%] between the best solution found and a lower bound; "$\max_{\text{Gap}}$" shows the maximal gap [%] in the respective test set. In column "∅Range" the average deviation [%] between the best and the worst solution found is given; "$\max_{\text{Range}}$" shows the maximal value [%]. Lower bounds are calculated on the basis of stage-based lower bounds adapted from Hidri and Haouari (2011).

4 Discussion and future research

Table 1 shows that both the multi-start procedure and the lower bounds perform well for instances with exactly one machine per stage or with one to three parallel machines per

Table 1. Computational results

	identical					average				
60 jobs	#Opt	∅Gap	$\max_{Gap}$	∅Range	$\max_{Range}$	#Opt	∅Gap	$\max_{Gap}$	∅Range	$\max_{Range}$
1 machine	3	1.2	4.3	11.9	14.6	2	1.0	3.0	19.9	25.3
2 machines	0	9.6	12.5	19.5	22.2	0	6.8	9.9	22.3	27.4
1-3 machines	1	1.2	2.5	16.3	20.5	0	2.2	4.3	21.6	29.4
	identical					average				
120 jobs	#Opt	∅Gap	$\max_{Gap}$	∅Range	$\max_{Range}$	#Opt	∅Gap	$\max_{Gap}$	∅Range	$\max_{Range}$
1 machine	2	0.6	1.4	7.1	9.4	3	0.1	0.4	10.5	11.6
2 machines	0	8.5	10.4	18.5	22.6	0	7.9	9.1	21.8	26.0
1-3 machines	1	3.1	8.6	14.6	26.9	2	0.6	1.6	12.1	14.7

stage. 50% of all instances with one machine and 20% of all instances with 1–3 machines could be solved to optimality and for the remaining instances the gaps are relatively small (lower than, or equal to, 3.1%). The "$\max_{Gap}$"-value of 8.6% for one instance can be ascribed to the fact that for this instance many (2–3) machines exist on the stages and no bottleneck can be identified. A bottleneck occurs if the average waiting time of all jobs at a stage is relatively high (i.e., many jobs in the corresponding queue, large average processing times at the stage, and/or few parallel machines).

The results for instances with exactly two machines per stage are not as good as expected. The main reason for this is that the stage-based lower bounds are weak in environments without bottlenecks.

Regarding the average (and maximal) range between the generated solutions, the values are higher for instances with an "average" number of jobs per underground location. In this case, the number of jobs per queue is usually smaller than in the "identical" case and the selection probability of a job with a small priority-value is overvalued.

Our further research will contain the improvement of lower bounds and the development of a local search heuristic, where the presented multi-start procedure is used as a basis. Moreover, a larger test set must be investigated in order to analyze the behavior of the procedure if the number of machines/jobs increases, or the number of jobs per underground location fluctuates strongly.

References

Hidri L., M. Haouari, 2011, "Bounding strategies for the hybrid flow shop scheduling problem", *Applied Mathematics and Computation*, Vol. 217, pp. 8248–8263.

Kim H.-W., D.-H. Lee, 2009, "Heuristic algorithms for re-entrant hybrid flow shop scheduling with unrelated parallel machines", *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vol. 223, Issue 4, pp. 433-442.

Kolisch R., 1996, "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation", *European Journal of Operational Research*, Vol. 90, Issue 2, pp. 320-333.

Newman A.M., E. Rubio, R. Caro, A. Weintraub and K. Eurek, 2010, "A review of operations research in mine planning", *Interfaces*, Vol. 40, Issue 3, pp. 222-245.

Pinedo M.L., 2008, "Scheduling: Theory, Algorithms, and Systems", 3rd edition, *Springer*, New York.

Ruiz R., J.A. Vázquez-Rodríguez, 2010, "The hybrid flow shop scheduling problem", *European Journal of Operational Research*, Vol. 205, Issue 1, pp. 1-18.

Schulze M., J. Zimmermann, 2011, "Scheduling in the Context of Underground Mining", in *Operations Research Proceedings 2010*, eds. Hu B., K. Morasch, S. Pickl and M. Siegle, Springer, Berlin, pp. 611-616.

A MIP-based Decomposition Heuristic for Resource-constrained Project Scheduling

Tom Rihm and Norbert Trautmann

University of Bern, Switzerland
tom.rihm@pqm.unibe.ch and norbert.trautmann@pqm.unibe.ch

Keywords: RCPSP, decomposition, mixed-integer linear programming.

1 Introduction

Resource-allocation problems arise in new product development, engineering, software development, and other projects. In general, such resource-allocation problems pose intractable combinatorial optimization problems. One of the most extensively studied problems in this area is the resource-constrained project scheduling problem RCPSP. This problem consists in scheduling a given set of activities subject to completion-start precedence and renewable-resource constraints such that the project duration is minimized.

In the literature, several mixed-integer linear programming (MILP) formulations of the RCPSP have been presented (cf., e.g., Koné *et al.* 2011). A major advantage of such MILP formulations is the flexibility to account for additional constraints or modified planning objectives. Despite improvements in optimisation software and computer hardware, nowadays such MILP formulations are applicable to small-sized problem instances only. Besides these MILP formulations, a large variety of heuristic methods have been proposed (cf., e.g., Kolisch and Hartmann 2006 for a review); however, these heuristic methods make use of specific schedule-generation schemes, that need to be adapted individually when additional constraints or modified planning objectives must be taken into account.

In this paper, we present a heuristic method that combines mixed-integer linear programming with a network-based decomposition. Under this heuristic, at first the project network is decomposed into subsets of activities. Then, the subsets of activities are iteratively added to the partial schedule, applying an exact MILP Model. Thereby, the activities which have already been scheduled may be delayed, but may not be shifted earlier in time; this reduces the number of variables in the corresponding MILP and allows to insert new activities between activities which have already been scheduled. Our computation results indicate that the heuristic is able to devise optimal solutions to non-trivial problem instances, and outperforms the MILP of Pritsker *et al.* (1969) on medium-sized instances.

To the best of our knowledge, such a decomposition has not been proposed in the literature. Other decomposition heuristics for the RCPSP have been proposed by, e.g., Sprecher (2002) and Debels and Vanhoucke (2007). In contrast to our approach, in these approaches first a feasible schedule for the entire project is computed using a constructive heuristic; then, subsets of activities being in progress according to this schedule within certain time intervals are selected, and the corresponding parts of the schedule are improved by applying a branch-and-bound method or a genetic algorithm to these subsets of activities, respectively.

The remainder of this paper is organized as follows. In Section 2, we present the new heuristic approach. In Section 3, we apply the heuristic to an illustrative example. In Section 4, we report the computational results. In Section 5, we provide some concluding remarks.

2 Heuristic approach

The heuristic consists of two phases. In the first phase, the set of all project activities is partitioned into several subsets. In the second phase, these subsets are scheduled iteratively.

In the first phase, we assign a priority value to each activity. Then we determine a path with maximum priority (i.e. sum of the priority values of all activities of the path) in the activity-on-node network. The nodes of this path form the first subset. Then, we iteratively form further subsets as follows. In the first step, we determine an activity with highest priority in the path that had been determined in the preceding iteration; for the reminder of the decomposition phase, we set the priority value of that activity to zero. In the second step, we again determine a path with maximum priority in the resulting network. All nodes of this path which do not belong to any subset yet form the next subset; if this subset is empty, we delete it from the partition. We repeat these two steps until the partition is complete.

In the second phase, we iteratively schedule the subsets of activities in the order as they have been formed in the first phase. In the first iteration, the start times of the activities of the first subset are determined; in the subsequent iterations, the start times of the activities in the current subset as well as the start times of the activities in all preceding subsets are subject to decision. In order to determine these start times, we apply the MILP of Pritsker *et al.* (1969). After solving the MILP, we perform a left-shift of all activities that have been scheduled. Then, we set the earliest start time of each activity that has been scheduled to its start time in the left-shifted schedule and set the length of the scheduling horizon to the current makespan increased by the sum of the durations of the activities in the next subset and the product of the maximum duration of all activities that have been scheduled multiplied by the cardinality of the next subset. Eventually, we update the earliest and the latest start times of all activities. We continue until all subsets have been scheduled.

3 Illustrative example

For the purpose of illustration, we apply our method to an illustrative example introduced in Sprecher (2002). The project comprises 7 activities and one resource with capacity $R_1 = 6$. The activity-on-node network is shown in Figure 1, where the node labels indicate the durations d_i and the resource requirements $r_{i,1}$ of the activities $i = 1, \ldots, 9$.

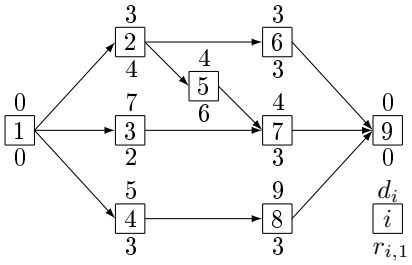


Fig. 1. Example network

In the decomposition phase, we set the priority value of an activity to its duration, i.e. a path with maximum priority correspond to a critical path in the activity-on-node network. We obtain the subsets $\{4, 8\}$, $\{3, 7\}$, $\{2, 5\}$, and $\{6\}$. Figures 2, 3 and 4 show the schedules obtained after the first, the second and the third iteration, respectively. In the third iteration, activities 2 and 5 are inserted in the partial schedule; to this end, activities 4, 7 and 8 are scheduled at a later position than in the second iteration. In the fourth and last iteration, activity 6 is added to the schedule. In the course of this, all the other activities may again be shifted later but not earlier in time. The schedule obtained by our heuristic is shown in Figure 5 and represents an optimal solution to this example.

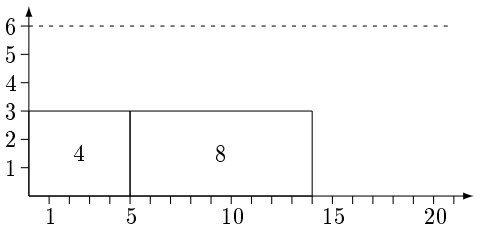


Fig. 2. Schedule after the first iteration

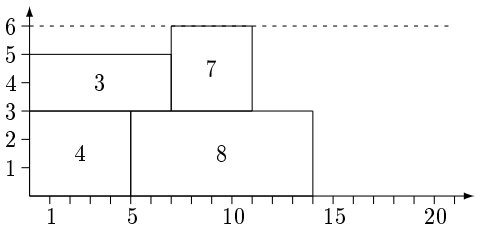


Fig. 3. Schedule after the second iteration

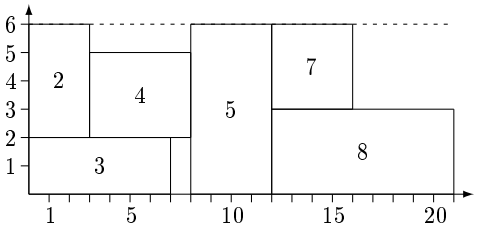


Fig. 4. Schedule after the third iteration

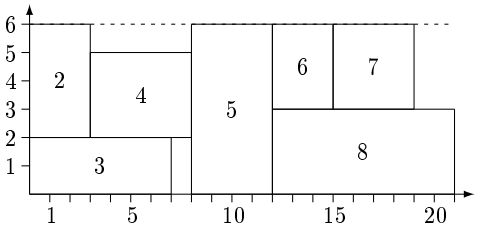


Fig. 5. Final schedule (optimal solution)

4 Computational results

We have implemented the heuristic method presented in Section 2 in AMPL; we have used Gurobi version 5.5 for solving the MILP models. All computations have been performed on a standard PC with a 3.40GHz Intel i7 CPU and 4GB RAM.

A large variety of priority rules have been proposed in the literature (cf., e.g., Klein 2000). In our test, our approach performed best when using the activities' duration as priority values. Tables 1 and 2 list the respective results of our heuristic (RT) for some instances from $j60$ set from the PSPLIB presented in Kolisch *et al.* (1999). The third and fourth column indicate the current best known upper and lower bound. We have used the formulation of Pritsker *et al.* (1969) as a reference (PWW), and we have prescribed a CPU time limit of 1800s. The entry lim indicates that the solution of the reference model has been stopped due to the time limit. Entries in boldface indicate that the makespan obtained by the new heuristic approach is smaller or equal to the makespan obtained with the reference model. Entries marked with $*$ indicate that the solution obtained is optimal.

The results shown in Tables 1 and 2 correspond to a mean makespan of 104.7. With priority rules GRD and GRPW* applied to our heuristic, the mean makespan is 105.7 and 106.0 respectively.

5 Conclusions and outlook

In this paper, we have proposed a novel MILP-based heuristic for the RCPSP. The heuristic decomposes the project network into subsets of activities, which are then scheduled iteratively by applying an MILP formulation. Our computational results indicate that for several instances, an optimal solution is obtained with this heuristic approach, and for instances with a large resource factor, the heuristic approach outperforms an exact MILP model.

The heuristic approach could be extended by a third phase, in which subsets of activities would be unscheduled and rescheduled in order to improve the project schedule; the

Table 1. Results on some instances with 60 activities, $RS = 0.20$ and $NC = 1.50$

Instance j60	RF	Best UB	Best LB	RT MS	RT	RT t^{CPU}	PWW MS	PWW t^{CPU}
1_1	0.25	77	77	**77**	*	2	77	2
1_2	0.25	68	68	**68**	*	3	68	14
1_3	0.25	68	68	**68**	*	3	68	3
1_4	0.25	91	91	**91**	*	2	91	3
1_5	0.25	73	73	**73**	*	3	73	8
1_6	0.25	66	66	**66**	*	3	66	13
1_7	0.25	72	72	74		6	72	59
1_8	0.25	75	75	76		2	75	14
1_9	0.25	85	85	**85**	*	3	85	9
1_10	0.25	80	80	**80**	*	4	80	1
5_1	0.50	76	76	**78**		111	82	lim
5_2	0.50	109	109	**109**	*	48	129	lim
5_3	0.50	80	80	**82**		6	83	lim
5_4	0.50	72	72	80		33	79	lim
5_5	0.50	108	108	**112**		45	136	lim
5_6	0.50	74	74	**76**		11	89	lim
5_7	0.50	75	75	**80**		27	91	lim
5_8	0.50	78	78	82		6	78	1468
5_9	0.50	83	83	84		4	83	37
5_10	0.50	81	81	**83**		14	85	lim
9_1	0.75	87	82	**92**		259	107	lim
9_2	0.75	82	82	**87**		23	100	lim
9_3	0.75	100	99	**105**		511	130	lim
9_4	0.75	87	87	**93**		46	306	lim
9_5	0.75	85	80	**91**		84	113	lim
9_6	0.75	111	105	**116**		609	388	lim
9_7	0.75	109	100	**124**		480	350	lim
9_8	0.75	96	94	**104**		712	131	lim
9_9	0.75	99	98	**105**		100	111	lim
9_10	0.75	93	88	**98**		581	111	lim
13_1	1.00	112	104	**118**		86	333	lim
13_2	1.00	106	101	**115**		799	309	lim
13_3	1.00	88	82	**93**		1082	109	lim
13_4	1.00	103	97	**108**		834	137	lim
13_5	1.00	97	91	**109**		901	308	lim
13_6	1.00	94	90	**99**		207	321	lim
13_7	1.00	87	80	**95**		735	153	lim
13_8	1.00	120	112	**129**		1230	341	lim
13_9	1.00	102	95	**105**		867	348	lim
13_10	1.00	117	112	**127**		1552	354	lim

Table 2. Results on some instances with 60 activities, $RS = 0.20$ and $NC = 2.10$

Instance j60	RF	Best UB	Best LB	RT MS	RT	RT t^{CPU}	PWW MS	PWW t^{CPU}
33_1	0.25	105	105	112		9	105	16
33_2	0.25	100	100	**100**	*	3	100	2
33_3	0.25	79	79	**79**	*	2	79	2
33_4	0.25	81	81	83		2	81	20
33_5	0.25	108	108	109		4	108	71
33_6	0.25	75	75	**75**	*	2	75	30
33_7	0.25	78	78	**78**	*	3	78	31
33_8	0.25	79	79	80		3	79	672
33_9	0.25	108	108	**108**	*	10	108	14
33_10	0.25	84	84	88		3	84	34
37_1	0.50	97	97	**102**		39	113	lim
37_2	0.50	95	95	**100**		29	120	lim
37_3	0.50	139	139	**144**		44	218	lim
37_4	0.50	101	101	**103**		16	329	lim
37_5	0.50	98	98	**107**		44	107	lim
37_6	0.50	102	102	**107**		83	346	lim
37_7	0.50	110	110	**120**		340	347	lim
37_8	0.50	93	93	97		11	93	414
37_9	0.50	96	96	**97**		18	318	lim
37_10	0.50	96	96	104		7	96	453
41_1	0.75	122	122	**129**		364	306	lim
41_2	0.75	113	113	**119**		118	202	lim
41_3	0.75	98	89	**103**		495	304	lim
41_4	0.75	133	133	**143**		79	–	lim
41_5	0.75	115	109	**123**		770	340	lim
41_6	0.75	134	134	**143**		699	339	lim
41_7	0.75	132	132	**142**		735	–	lim
41_8	0.75	135	135	**141**		284	351	lim
41_9	0.75	131	131	**137**		146	–	lim
41_10	0.75	111	105	**116**		415	364	lim
45_1	1.00	96	89	**105**		928	304	lim
45_2	1.00	144	134	**154**		1534	–	lim
45_3	1.00	143	133	**156**		1147	363	lim
45_4	1.00	108	101	**119**		636	–	lim
45_5	1.00	106	99	**119**		865	307	lim
45_6	1.00	144	132	**159**		1351	382	lim
45_7	1.00	122	113	**134**		1542	305	lim
45_8	1.00	129	119	**144**		1140	362	lim
45_9	1.00	123	114	**135**		1407	328	lim
45_10	1.00	114	102	**125**		440	312	lim

selection of these subsets would depend on the structure of the partition resulting from the decomposition phase.

References

Debels D., M. Vanhoucke, 2007, "A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem.", *OPER RES*, Vol. 55, pp. 457-469.

Klein R., 2000, "Bidirectional planning: improving priority rule-based heuristics for scheduling resource-constrained projects.", *EUR J OPER RES*, Vol. 127, pp. 619-638.

Kolisch R., S. Hartmann, 2006, "Experimental investigation of heuristics for resource-constrained project scheduling: An update.", *EUR J OPER RES*, Vol. 174, pp. 23-37.

Kolisch R., C. Schwindt and A. Sprecher, 1999, "Benchmark instances for project scheduling problems.", *in: J. Węglarz (Ed.), Project Scheduling - Recent Models, Algorithms and Applications, Kluwer Academic Publishers, Boston*, pp. 197-212.

Koné O., C. Artigues, P. Lopez and M. Mongeau, 2011, "Event-based MILP models for resource-constrained project scheduling problems.", *COMPUT OPER RES*, Vol. 38, pp. 3-13.

Pritsker A., L. Watters and P. Wolfe, 1969, "Multiproject scheduling with limited resources: A zero-one programming approach.", *MANAGE SCI*, Vol. 16, pp. 93-108.

Sprecher A., 2002, "Network decomposition techniques for resource-constrained project scheduling", *J OPER RES SOC*, Vol. 53, pp. 405-414.

Towards integrating extensions of resource constrained project scheduling problem

Elena Rokou[1], Konstantinos Kirytopoulos[2], Dimosthenis Drivaliaris[3]

[1] School of Mechanical Engineering, National Technical University of Athens, Athens, Greece
erokou@mail.ntua.gr
[2] School of Natural and Built Environments, Barbara Hardy Institute, University of South Australia, Adelaide, Australia
konstantinos.kirytopoulos@unisa.edu.au
[3] Department of Financial and Management Engineering, University of the Aegean, Chios, Greece
d.drivaliaris@fme.aegean.gr

Keywords: RCPSP, project scheduling, mathematical model, evolutionary algorithms.

1 Introduction

The resource-constrained project scheduling problem (RCPSP) is a complex problem even in its simplest form. Due to the difficulties encountered when solving it and much more when handling its various extensions and variations (Barrios, Ballestin & Valls 2011, Deblaere, Demeulemeester & Herroelen 2010, Pan, Hsaio & Chen 2008, Rivera & Celin 2010, Hartmann 2013, Böttcher, Drexl, Kolisch & Salewski 1999, Kolisch & Padman 2001), even today, there is a lack of generic models that integrate all the different facets of a project that should be scheduled and provide a solution process (Hartmann & Briskorn 2010). However, in practice projects often fail to fall precisely in a sole case of those studied in the literature.

To fill in this research gap, a holistic model is proposed in order to provide a way to define most of the desired characteristics, and provide a solution process that will generate project schedules adaptable to different project settings, organisational sizes and strategies and scalable according to the size and criticality of the undergoing project.

2 Problem Definition and Proposed Mathematical Formulation

The objectives to be pursued during project scheduling are defined by translating schedule characteristics, to actual objectives to be optimised during project scheduling.

Project planning entails defining which activity should be executed, when should its execution start and what amount of resources per resource type will be used. An activity can have one or more execution modes, meaning that it can be executed using various resource types and amounts of resources, resulting in different durations. The activities can be splittable or not based on the specificities of the task. Splits can happen in predefined points of time or in any time period. The activities can either require constant amounts of renewable resources, that is, the per-period request for a resource remains unchanged until the activity has been completed or the resource requests can vary with time. However, it is not very probable that all activities will have variable demands so a subset of activities with variable demands should be defined. To start executing an activity all its immediate predecessors should have been finished. This precedence concept in practical situations is extended by allowing start-start, finish-start, start-finish and finish-finish precedence constraints with both minimal and maximal time lags. Three different kinds of resources are considered: renewable, non-renewable and doubly constrained. Resource availabilities

usually are assumed to be constant over time. This assumption is not very close to what actually happens in practical cases where changing availability of workers due to vacations, maternity leaves, sickness or varying equipment capacities due to maintenance or damage is very common.

The proposed variation of the resource constrained project scheduling problem can be mathematically formulated introducing the binary decision variables x_{imqt} which are defined as follows:

$$x_{imqt} = \begin{cases} 1, \text{ if the segment } p_{imq} \text{ of } i \text{ in mode } m \text{ starts at } t \\ 0, \text{otherwise} \end{cases} . \tag{1}$$

The mathematical formulation, shown in Equations (2)-(10), is an extension of the model first presented by Pritsker, Watters & Wolfe (1969) to include preemption, multi-mode activities and generalised constraints. It is also based on the formulations provided by De Reyck & Herroelen (1999) and Hartmann (2013) for the MRCPSP-GPR and the RCPSP/t respectively.

$$\min f(x_{imqt}) , \tag{2}$$

subject to:

$$\sum_{m \in M_i} \sum_{t=0}^{T-1} \sum_{q=0}^{z_{im}} x_{imqt} = 1, \quad \forall i = 0, 1, \ldots, n+1 , \tag{3}$$

$$\left(\sum_{m \in M_i} \sum_{t=0}^{T-1} t x_{im0t} \right) + \delta_{imjn} \leq \sum_{n \in M_j} \sum_{t=0}^{T-1} t x_{jn0t} , \forall (i,j) \in A , \tag{4}$$

$$\sum_{i=1}^{n} \sum_{m \in M_i} \sum_{q=0}^{z_{im}} r^{\rho}_{imk\tau_{im}} x_{imqt} \leq \alpha^{\rho}_{kt} , \forall k \in R^{\rho} , \forall t = 0, 1, \ldots, T-1 , \tag{5}$$

$$\sum_{t=0}^{t_{l(x+1)}-1} \sum_{i=1}^{n} \sum_{m=1}^{m_{i\lambda}} \sum_{q=0}^{z_{im}} r^{\nu}_{iml\tau_{im}} x_{imqt} \leq \alpha^{\nu}_{lI_{lx}} , \forall l \in R^{\nu} , \forall x = 0, \ldots, X_l - 1 , \tag{6}$$

$$x_{0000} = 1 , \tag{7}$$

$$\sum_{t=0}^{T-1} t x_{imz_{im}t} \leq \sum_{t=0}^{T-1} t x_{(n+1)00t} , \forall i = 0, 1, \ldots, n , \tag{8}$$

$$\sum_{t=0}^{T-1} t x_{im(q-1)t} + d_{imq} \leq \sum_{t=0}^{T-1} t x_{imqt} , \forall i = 0, 1, \ldots, n , \forall m \in M_i , \forall q = 1, \ldots, z_{im} , \tag{9}$$

$$x_{imqt} \in \{0, 1\} , \forall i = 0, 1, \ldots, n+1 , \forall m \in M_i , \forall t = 0, 1, \ldots, T-1 . \tag{10}$$

The objective function (2) minimises the selected objective; for example the objective of minimising the makespan can be written as

$$\min \sum_{t=0}^{T-1} t x_{(n+1)00t} .$$

Constraints (3) ensure that each activity is assigned exactly one mode and exactly one start time.

Constraints (4) denote the generalised precedence relations with minimal and maximal time lags, where the actual values for each time lag, whether they originate from a minimal time lag or a maximal time lag are given by δ_{imjn}.

The resource constraints are given in Equations (5) and (6) for renewable and non-renewable resources, respectively.

Equations (7) and (8) ensure that the first activity of the schedule is the dummy source and the last the dummy sink.

Constraints (9) ensure that the splitted activities will be executed in the correct order.

Equation (10) forces the decision variables to assume binary values.

3 Solution Process

The proposed solution algorithm can handle a variety of problems such as the simple RCPSP, the multi mode RCPSP, with or without generalised precedence constraints, having or not variable renewable and/or non renewable resource demands and requirements.

The backbone, of the whole process, is a Genetic Algorithm (GA) that acts as a moderator of the solution process. The proposed process essentially lets the GA decide which algorithm and decoding procedure is promising and work with it. In other words, the evolution is used not only to find a good solution for the problem but also a good algorithm to solve the problem. The genetic algorithm adapts itself to the problem instance actually solved. This way not only the list of activities to be scheduled but also the algorithm itself and the decoding algorithm are subject to genetic optimisation.

Initially, the input data are analysed and transformed to a predefined form for ease of usage and to eliminate redundant data. Then an initial solution set is randomly generated and crossover and mutation operators are used to generate the offspring population.

The moderator GA is responsible for mutation, crossover and selection of the next generation and the auxiliary algorithms (tabu search (TS), simulated annealing (SA), particle swarm optimisation (PSO) and genetic algorithm(GA)) are responsible for the search of a sub-space of the solution space, based on the lists of activities that were given as input, causing this way the update of the given lists of activities in relation to the results found and the calculation of the fitness. As a result, the auxiliary algorithms effectuate a parallel, local or global search, depending on the algorithm, in the solution space of a chromosome or a group of chromosomes belonging to the current generation and exchange the given chromosomes with better ones, having better fitness value.

Finally, the fitness is calculated straightforward using the corresponding objective function on each decoded chromosome.

4 Computational Results

A comparison of the results given by the proposed holistic model and algorithm to each specific problem that was integrated in our model was conducted. The experiment was designed to prove that the proposed method leads to as good results as the best known for each variation and in specific cases it has been seen to give even better results.

In Table 1, a summary of the experimental results, is shown. In all cases the proposed algorithm gives the same optimal or best known results with those that are published in PSPLib and in most cases has a higher accuracy with a percentage of deviation from the best known value lower than 2% for each category of cases. Therefore, the aim to be at least as good as the best known algorithm has been achieved.

[1] The algorithm offered in some cases better results than that appearing in literature. These better results are appearing here as deviations.

Table 1. Comparative results for single-objective instance

Instances	Average Deviation	Max Deviation	Hit of Optimal/UB
RCPSP J30	0.25%	3%	96.7%
RCPSP J120	1.42%	8%	34.46% [1]
PRCPSP J30	0.12%	2.5%	98.7%
PRCPSP J120	1.21%	5%	42.73% [1]
MRCPSP C15	0.23%	1%	98.9%
MRCPSP C21	0.01%	1%	99.9%
MRCPSP J10	0.01%	0%	99.9%
RCPSP-t J30	0.05%	1%	99.7%
RCPSP-t J120	0.22%	1.5%	99.5%
RCPSPmax J30	0.12%	1.8%	90.12%

5 Conclusions

Summarising, this research has the following innovative parts:

- a holistic mathematic model integrating all the known extensions and variations of the resource constrained project scheduling problem, namely, preemption, multiple modes of execution, generalised precedence constraints, variable resources availabilities and requests over time and its binary formulation,
- an adaptive hybrid algorithm that handles the selection of the auxiliary algorithms that will be used for the solution of each problem's instance and the ways that the schedules will be generated (s-GS or p-SGS),
- achievement of good optimisation results as the proposed moderator algorithm leads to as good results as the best known for each variation and in specific cases, although more experiments should take place, it has been seen to give even better results,

The proposed work is a first step towards flexible models of complicated situations, adaptable solution methods and results aiming at supporting the decision makers without giving aphorisms about which solution is the best one and which is not.

The next step on this research is to enhance it by adding a mechanism for automatically dealing with infeasibilities instead of interactively doing it.

References

Barrios, A., Ballestin, F. & Valls, V. (2011), 'A double genetic algorithm for the mrcpsp/max', *Project Management and Scheduling* **38**(1), 33–43.

Böttcher, J., Drexl, A., Kolisch, R. & Salewski, F. (1999), 'Project scheduling under partially renewable resource constraints', *Management Science* **45**(4), 543–559.

De Reyck, B. & Herroelen, W. (1999), 'The multi-mode resource-constrained project scheduling problem with generalized precedence relations', *European Journal of Operational Research* **119**(2), 538–556.

Deblaere, F., Demeulemeester, E. & Herroelen, W. (2010), 'Reactive scheduling in the multi-mode rcpsp', *Computers and Operations Research* **38**(1), 63–74.

Hartmann, S. (2013), 'Project scheduling with resource capacities and requests varying with time: A case study', *Flexible Services and Manufacturing Journal* **25**(1-2), 74–93.

Hartmann, S. & Briskorn, D. (2010), 'A survey of variants and extensions of the resource-constrained project scheduling problem', *European Journal of Operational Research* **207**(1), 1–14.

Kolisch, R. & Padman, R. (2001), 'An integrated survey of deterministic project scheduling', *Omega* **29**(3), 249–272.

Pan, N. H., Hsaio, P. W. & Chen, K. Y. (2008), 'A study of project scheduling optimization using tabu search algorithm', *Engineering Applications of Artificial Intelligence* **21**(7), 1101–1112.

Pritsker, A., Watters, L. & Wolfe, P. M. (1969), 'Multiproject scheduling with limited resources-a zero- one programming approach', *Mgmt Science* **16**(1), 93–108.

Rivera, J. C. & Celin, A. J. (2010), 'Hybrid variable neighborhood and simulated annealing heuristic algorithm to solve rcpsp', *Algoritmo heuristico habrido con multiples vecindarios y recocido simulado para resolver el RCPSP* (56), 255–267.

Appendix 1

Table 2. Basic Notation

Symbol	Definition
$V = \{0, 1, \ldots, n, n+1\}$	the set of activities i
n	number of real activities
$G(V, A)$	directed graph of precedence or temporal constraints
T	the planning horizon, sum of maximal durations of all activities
t	periods, index of T
$[t, t+1)$	time interval corresponding to period t
$Act(t)$	set of all the activities of which a time unit is in progress at t, $t = 0, 1, \ldots, T$
R^{ρ}	set of renewable resources
α^{ρ}_{kt}	variable amount of available units of renewable resource k, $t = 0, \ldots, T-1$
R^{ν}	set of non-renewable resources
t_{lx}	each non renewable resource $l \in R^{\nu}$ is associated to a subset $\{t_{lx} \vert x = 0, \ldots, X_l\}$ of $\{0, 1, \ldots, T\}$ with $0 = t_{l0} < \ldots < t_{lx} < t_{l(x+1)} < \ldots < t_{lX_l} = T$
I_{lx}	subintervals $I_{lx} = [t_{lx}, t_{l(x+1)}]$, $x = 0, \ldots, X_l - 1$ composing a partition of $[0, T)$
$\alpha^{\nu}_{lI_{lx}}$	variable amount of available units of non-renewable resource l
M_i	set of modes (alternative ways of execution) of activity i
$M_i^{p'}$	set of non-preemptive modes of activity i
M_i^{p}	set of preemptive modes of activity i
d_{im}	duration of activity i in mode m
$r^{\rho}_{imk\tau_{im}}$	per period usage of activity i of renewable resource k in mode m
$r^{\nu}_{iml\tau_{im}}$	per period consumption of activity i of non-renewable resource l in mode m
z_{im}	number of splits on activity i in mode m, $z_{im} = 0, ..., d_{im} - 1$
p_{imq}	segment of the preempted activity i with $q = 0, 1, 2, \ldots, z_{im}$
d_{imq}	duration of segment q of activity i in mode m
s_{imq}	start time of segment q of activity i in mode m
f_{imq}	finish time of segment q of activity i in mode m
s_{im0}	start time of activity i
f_{imz_m}	finish time of activity i
s_{000}	start time of project
$f_{(n+1)00}$	finish time of project
$S = (s_{imq})$	schedule, vector of start times of all segments of all activities
$SS^{min}_{imjn}/SS^{max}_{imjn}$	min/max time lag between start of activities i and j in modes m and n
$SF^{min}_{imjn}/SF^{max}_{imjn}$	min/max time lag between start of activity i in mode m and finish of j in mode n
$FS^{min}_{imjn}/FS^{max}_{imjn}$	min/max time lag between finish of activity i in mode m and start of j in mode n
$FF^{min}_{imjn}/FF^{max}_{imjn}$	min/max time lag between finish of activities i and $j$in modes m and n
δ_{imjn}	min/max time lag between start of activities i and j in modes m and n

Power- and energy- related instances in a problem of scheduling computational jobs

Różycki R., Węglarz J.

Poznan University of Technology, Institute of Computing Science
e-mail: rafal.rozycki@cs.put.poznan.pl

Keywords: doubly-constrained resource, discrete-continuous scheduling, nonlinear mathematical programming.

1. Introduction

In this paper we propose methods for improving efficiency of the general approach for solving a machine scheduling problem with an additional, doubly-constrained resource. In this problem preemptable jobs are to be scheduled on parallel identical machines in order to minimize makespan. A general dynamic model of a job execution is used to express the relation between a processing rate of a job and its resource usage. This way one can model e.g. the practical problem of scheduling computational jobs on an advanced multicore processor where available amounts of power and energy should be respected. The problem belongs to the class of laptop problems (Bunde (2006)).

The general approach for solving the problem bases on the work of Węglarz (1981) and has been adopted to machine scheduling case in Różycki, Węglarz (2009). In this approach an optimal schedule is found by solving a nonlinear mathematical programming problem, in which the number of variables and the number of constraints exponentially grows with a size of an instance.

We propose some preprocessing procedures that applied to an instance of the problem allow in some cases to find an optimal solution in a simpler way. The computational complexity of the procedures is linear, thus it is highly recommended to use them before the optimal solution is found using the general approach.

2. Problem formulation

We consider a problem where n independent, preemptable jobs should be processed on m parallel identical machines. The ready time of each job is equal to 0. Each job requires both a machine and an amount of a single doubly-constrained resource for its processing. The processing time of job i is unknown in advance, since its processing rate at time t depends on a temporal usage of the resource - $p_i(t)$. This relation is expressed by an increasing speed function s_i ($s_i(0) = 0$). Moreover, a completion time C_i of job i is also related to its size w_i. As a consequence the considered model of job processing may be expressed in the following form:

$$w_i = \int_0^{C_i} s_i(p_i(t))dt \tag{1}$$

It is worth stressing that each job may be characterized by different speed function in model (1).

The resource is shared among jobs and its doubly-constrained nature is expressed by the following inequalities:

$$\sum_{i=1}^{n} p_i(t) \le P \quad \text{for every } t > 0 \tag{2}$$

$$\sum_{i=1}^{n} \int_0^{C_{\max}} p_i(t)dt \le E \tag{3}$$

where C_{max} is the completion time of a last job in a schedule.

Let us assume that P and E denote respectively: the known limit for available power and the available amount of energy. The problem is to find both: an assignment of jobs to machines and an allocation of the resource to jobs that lead to a schedule with the minimal length.

3. General approach

The general approach is based on the result of Weglarz (1981) which, for given limits of P and E, allows to find a time-optimal resource allocation to n independent jobs if $n \leq m$. In this case, for the considered class of speed functions the shortest schedule is obtained by fully parallel execution of all jobs, where each job is processed using constant amount of power and all jobs are finished at the same moment. Of course, in the case where $n > m$ only m jobs may be performed in parallel. Thus in the general approach, in order to construct an optimal schedule we have to consider all m-element combinations from the set of n jobs. Each combination represents a part of the final schedule, i.e. a single group of jobs that may be performed in parallel on limited number of processors. Since the processors are identical, an order of jobs in such combination may be neglected. Although the jobs are preemptable, we assume that preemption of a job within a combination is forbidden. However, a job is still able to migrate among processors because in consecutive combinations it may be allotted to different processors at no additional cost.

For a sequence of all combinations (an order of combinations in the sequence is arbitrary) a nonlinear mathematical programming (NLP) problem has to be solved to find a schedule with a minimal makespan. A division of energy and job sizes among the combinations leading to a time-optimal schedule is found by solving such NLP problem.

Denote combinations by Z_k, $k = 1, 2, ..., r = \binom{n}{m}$. Denote also by K_i, $i = 1, 2, ..., n$, the set of indices of Z_k's containing job i and by w_{ik} part of job i processed in Z_k. Let T_k and E_k, $k = 1, 2, ..., r$, denote respectively: the length of the part of the schedule corresponding to Z_k and an amount of energy consumed for processing all parts of jobs from Z_k. Notice that for a given level of P, T_k, $k = 1, 2, ..., r$, is a function of w_{ik}, E_k, which are unknown a priori. An exemplary division of job sizes among all combinations is presented in Fig.1.

	$\left[\begin{Bmatrix}1,\\2,\\3\end{Bmatrix}\right.$	$\begin{Bmatrix}1,\\2,\\4\end{Bmatrix}$	$\begin{Bmatrix}1,\\2,\\5\end{Bmatrix}$	$\begin{Bmatrix}1,\\3,\\4\end{Bmatrix}$	$\begin{Bmatrix}1,\\3,\\5\end{Bmatrix}$	$\begin{Bmatrix}1,\\4,\\5\end{Bmatrix}$	$\begin{Bmatrix}2,\\3,\\4\end{Bmatrix}$	$\begin{Bmatrix}2,\\3,\\5\end{Bmatrix}$	$\begin{Bmatrix}2,\\4,\\5\end{Bmatrix}$	$\left.\begin{Bmatrix}3,\\4,\\5\end{Bmatrix}\right]$
	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8	Z_9	Z_{10}
machine 1	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}	w_{16}	w_{27}	w_{28}	w_{29}	$w_{3,10}$
machine 2	w_{21}	w_{22}	w_{23}	w_{34}	w_{35}	w_{46}	w_{37}	w_{38}	w_{49}	$w_{4,10}$
machine 3	w_{31}	w_{42}	w_{53}	w_{44}	w_{55}	w_{56}	w_{47}	w_{58}	w_{59}	$w_{5,10}$
	$\leftarrow T_1 \rightarrow$	$\leftarrow T_2 \rightarrow$	$\leftarrow T_3 \rightarrow$	$\leftarrow T_4 \rightarrow$	$\leftarrow T_5 \rightarrow$	$\leftarrow T_6 \rightarrow$	$\leftarrow T_7 \rightarrow$	$\leftarrow T_8 \rightarrow$	$\leftarrow T_9 \rightarrow$	$\leftarrow T_{10} \rightarrow$

Figure 1. An example of the division of sizes of jobs among all possible combinations in the case of $n = 5$ jobs and $m = 3$ machines

The NLP problem has the following form in the reminded general approach:

Problem NLP1

(minimize) $$T = \sum_{k=1}^{r} T_k^* \left(\{w_{ik}\}_{i \in Z_k}, E_k \right) \tag{4}$$

s.t.: $$\sum_{k=1}^{r} E_k \leq E \tag{5}$$

$$\sum_{k \in K_i} w_{ik} = w_i, \quad i = 1, 2, ..., n \tag{6}$$

$$w_{ik} \geq 0,\ E_k > 0,\ i = 1, 2, ..., n;\ k \in K_i \tag{7}$$

where T_k^* is found as the only positive root of the equation:

$$T_k \sum_{i \in Z_k} s_i^{-1}(w_{ik} / T_k) = E_k \tag{8}$$

if $$\sum_{i \in Z_k} s_i^{-1}(w_{ik} / T_k) \leq P \tag{9}$$

or equation:

$$\sum_{i \in Z_k} s_i^{-1}(w_{ik} / T_k) = P \tag{10}$$

otherwise.

Notice that (8) and (10) are direct consequences of, respectively: (3) and (2). To find the shortest k-th part of the schedule for the given w_{ik}, $i \in Z_k$, and E_k it is better to start with equation (8), since it is usually of simpler form than (10). For example, for practically justified functions:

$$s_i(\cdot) = c_i p_i^{1/\alpha_i}, c_i > 0, \tag{11}$$

and $\alpha_i = \{2, 3, 4\}$, both: (8) and (10) are analytically solvable algebraic equations of an order less than or equal to 4, however (8) is of an order less by 1, and thus is analytically solvable also for $\alpha_i = 5$. If the sum of power allocated to jobs using (8) exceeds available amount P, we have to solve (10), but the information obtained by solving (8) is valuable anyway. Nevertheless, it is worth mentioning that at least equation (8) and inequality (9) has to be solved for each of r parts of the schedule in NLP1. Thus, it is strongly reasonable finding the cases where this approach may be simplified.

4. Special instances of the problem

In this section we show the methods for determining the instances for which one can formulate simpler nonlinear problems than NLP1. The new formulations of NLP problem differ mainly with the way a current value of the criterion function is calculated.

Let us start with an observation that relation between energy E_i consumed by job i and its constant power usage p_i may be represented by the function:

$$E_i(p_i) = w_i \frac{p_i}{s_i(p_i)} \tag{12}$$

Below we present two basic lemmas for which the proofs are obvious.

Lemma 1

Let us assume that job i is processed using a constant amount of power p_i. Then energy E_i consumed during processing of job i is an increasing function of p_i for $p_i \in (0,\infty)$.

Lemma 2

Let us assume that job i is processed using a constant amount of power p_i. If the speed function s_i fulfills the condition:

$$s_i(p_i) = \frac{1}{s_i(1/p_i)}, \quad p_i > 0 \tag{13}$$

then energy E_i consumed during processing of job i is strictly concave function of of p_i for $p_i \in (0,\infty)$.

Of course, Lemma 2 is valid e.g. for processing speed functions (11) and $\alpha_i > 1$. As a consequence of these two lemmas one can propose conditions (without proofs) which allow identify the special instances of the problem.

4.1. Energy related instances

Corollary 1

For instances of the problem where all the speed functions satisfy both: (13) and the following inequality:

$$p_i^E < P, \quad i = 1, 2, ..., n \tag{14}$$

where p_i^E is the (only) positive root of the equation:

$$p_i^E \frac{w_i}{s_i(p_i^E)} = E, \quad i = 1, 2, ..., n \tag{15}$$

in order to find an optimal solution the following nonlinear programming problem has to be solved instead of NLP1:

Problem NLP2
(minimize) (4)
s.t.: (5), (6), (7),
where "=" should be put in (5) and T_k^*, $k = 1, 2,..., r$ is calculated from (8).

4.2. Power related instances

Corollary 2
For instances of the problem where the following inequality is fulfilled:

$$\sum_{i=1}^{n} \frac{w_i P}{s_i(P)} \leq E \tag{16}$$

in order to find an optimal solution the following nonlinear programming problem has to be solved instead of NLP1:

Problem NLP3
(minimize) (4)
s.t.: (6), (7),
where T_k^*, $k = 1, 2,..., r$ is calculated from (10).

5. Summary

Corollaries 1 and 2 may be utilized on a preprocessing step of a procedure of an optimal schedule generating. For a given instance of the problem one can determine if (14) or (16) is satisfied. If one of the two cases happens, the simpler nonlinear programming problem may be solved to reach optimality. If we assume that (15) is analytically solvable (what is the case for the speed functions (11)) such preliminary procedure has the computational complexity O(n).

Of course, Corollaries 1 and 2 do not exhaust all the cases where power or energy constraint is inactive. One can observe that among all formulated NLP problems, NLP2 is the easiest to solve. Therefore, for instances for which (14) and (16) are not satisfied, we suggest the following procedure. Firstly one can try to find an optimal schedule by solving NLP2. If the amount of power used by jobs in each part of schedule in the obtained solution does not exceed P, then the solution is optimal. Otherwise one can try to solve the problem using NLP3. The solution obtained in this way is optimal if the total energy consumption in the schedule is not greater than E.

If neither NLP2 nor NLP3 produce the optimal solution, the only way to solve the problem is to use the described general methodology and solve NLP1.

Acknowledgements

The research was partially supported by Polish National Science Center, Grant no. 2013/08/A/ST6/00296.

References

Bunde D.P., 2006, "Power-aware Scheduling for Makespan and Flow", Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures, Cambridge, Massachusetts, USA, pp.:190-196.

Różycki R., J. Węglarz, 2009, "On job models in power management problems", Bulletin of the Polish Academy of Sciences, Technical Sciences, Vol. 57(2) 2009, pp.147-151.

Węglarz J., 1981, "Project scheduling with continuously-divisible doubly constrained resources", Management Science, Vol. 27(9), pp.:1040-1053.

Scatter search based methods for a distributed flowshop scheduling problem

Rubén Ruiz[1] and Bahman Naderi[2]

[1] Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, 46021, València, Spain.
rruiz@eio.upv.es

[2] Department of Industrial Engineering, Faculty of Engineering, University of Kharazmi, Karaj, Iran.
bahman.naderi@aut.ac.ir

Keywords: Distributed scheduling, Permutation flowshop, Scatter Search.

1 Introduction

The Distributed Permutation Flowshop Problem (DPFSP) was recently introduced by Naderi and Ruiz (2010) as a generalization of the flowshop layout where more than one factory is available to process jobs. Distributed manufacturing is common in large enterprises in today's globalized market. The problem entails two decisions: assigning jobs to factories and scheduling each factory. This short abstract summarizes the main elements of a recently introduced scatter search (SS) method to optimize the maximum makespan among the factories. It is shown that the proposed algorithm produces better results than existing methods by a significant margin.

In the DPFSP the set N of n jobs must be processed by a set G of F factories. Each factory has an identical set M of m machines. The processing times of of the jobs are the same for all factories and are denoted by p_{ij}. Once assigned to a factory, a job has to be completed in that factory by being processed in all machines. The objective is to minimize the maximum makespan among all factories. Naderi and Ruiz (2010) referred to this problem as $DF/prmu/C_{\max}$. It was also demonstrated that no factory should be left empty with no jobs assigned (if $n > F$) for makespan improvement. The total number of possible solutions in the DPFSP is $\binom{n-1}{F-1} n!$ Furthermore, the DPFSP reduces to the regular flowshop if $F = 1$, so it can be concluded that the DPFSP is also an $\mathcal{NP}$-Hard problem. Naderi and Ruiz (2010) proposed six different Mixed Integer Linear Programming models for the DPFSP together with 12 heuristics that resulted from applying two different job to factory assignment rules to six heuristics for the regular PFSP: 1) Assign a given job j to the factory with the lowest current $C_{\max}$, not including job j and 2) Assign job j to the factory which completes it at the earliest time, i.e., the factory resulting in the lowest $C_{\max}$ after assigning job j. The rules are applied each time a job is scheduled. From the six tested heuristics the NEH method with the second job to factory assignment rule of Naderi and Ruiz (2010) (referred to as NEH2) resulted in the best heuristic performance. They also presented a simple Variable Neighborhood Descent (VND) starting with the NEH2 solution with two neighborhoods, referred to as VND(a). As Naderi and Ruiz (2010) pointed out, prior to 2010 there was almost no literature on distributed flowshop scheduling. Later, Lui and Gao (2010) presented a complex electromagnetism metaheuristic (EM). In the computational evaluation, EM was not compared against VND(a). Furthermore, EM's CPU times are significantly larger than those of VND(a). Later, Gao and Chen (2011a) presented a Hybrid Genetic Algorithm with local search (HGA) which reported better solutions than

VND(a) but at the expense of much larger CPU times (about 246 times larger). Gao and Chen (2011b) presented an improvement of the NEH heuristic and the NEH2 of Naderi and Ruiz (2010) referred to as NEHdf. In the computational experiments the method is shown to slightly outperform NEH2 (however, in a provided statistical experiment, NEHdf is not shown to statistically outperform NEH2). More recently, Gao *et. al.* (2012b), presented a genetic algorithm which is shown to slightly outperform the HGA of Gao and Chen (2011a). Gao *et. al.* (2012a) presented a revised VNS method. Gao *et. al.* (2013) have presented a tabu search method. Lin *et. al.* (2013) have presented an iterated greedy method (IG_{VST}). Wang *et. al.* (2013) have proposed an estimation of distribution algorithm (EDA) needing no less than almost 788 times more CPU time than VND(a). As our critical review shows, many of the recently proposed methods have not been compared against each other. Most comparisons are done only against VND(a), which is no more than a heuristic with some local search.

2 Some aspects of the proposed scatter search method

In the proposed algorithm we employ some advanced techniques like a reference set made up of complete and partial solutions along with other features like restarts and local search. The allowed space in this short abstract is not enough to explain the proposed SS in detail so we basically summarize the most important aspects. The solution for the DPFSP is a list of jobs per factory. This jointly indicates the assignment of jobs to factories and the processing order at each factory. In the proposed DPFSP SS procedure we employ two different sets inside the reference set. The first is set H which contains a number b of the best ever found solutions. The second set, denoted as S, contains l factory assignment vectors. The union of these two sets makes the reference set, i.e., $RefSet = H \cup S$ of size $b+l$. Set H contains full solutions and set S only contains factory assignments for jobs. For the initial construction of set H we start with a $Psize$ of 25 random job permutations. 24 of these permutations are used as an initial ordering that is passed to the NEH2 method of Naderi and Ruiz (2010). For the last 25th permutation we use the regular NEH initial ordering instead of random. After, the best b solutions among these 25 are included in set H. Note that this applies to the initial H set construction. Later, at each iteration of the SS procedure, set H contains the best b visited solutions. As for set S, used for diversification, we simply initialize it with random job to factory assignments. At each iteration of the SS algorithm, sets H and S are combined. Therefore, and in order to keep the diversity, set S is randomly regenerated at each iteration of the SS method. The subset generation method is also different from most scatter search applications given the two sets H and S inside $RefSet$. The procedure consists of selecting all possible combinations of solutions in set H with factory assignments in set S. Therefore, at each iteration, $b \cdot l$ pairs are considered. The combination method is an important step and all pairs are combined. We refer to the solution selected from set H as $p1$ and to the factory assignment vector selected from S as $p2$. The new combined solution, referred to as pn is at first identical to $p1$. The combination method has n iterations. At each iteration, a job from pn is randomly selected, without repetition, so at the end all jobs have been selected. We refer to this randomly selected job as h. If a random number uniformly distributed between 0 and 1 ($rand$) is less than a given value p the combination method checks if job h is assigned to different factories in pn and in the job to factory assignment vector $p2$. If this is the case, job h is extracted from its current factory in pn and tested in all possible positions of the factory indicated in $p2$. The final placement of job h is the position resulting in the lowest makespan at the factory indicated in $p2$. If $rand$ is greater or equal than p then the job is not assigned to another factory and left untouched. We apply an improvement procedure at each iteration of the

SS to each solution pn obtained by the solution combination method. The improvement algorithm is a simplification of the VND procedure of Naderi and Ruiz (2010).
After the improvement method is applied to pn we check if the new solution pn is incorporated into the set H of $RefSet$. The inclusion happens if and only if: 1) The makespan of pn is better than the makespan of the worst solution in set H and 2) It is unique, i.e., there are no other identical solutions in set H. Finally, we include a procedure to restart set H after a number of iterations without improvements in the best solution. The procedure is simple; after a iterations without improvements in the best solution, the worst 50% of solutions in set H are discarded and the diversification generation method is employed to generate new solutions. The previous parameters are calibrated by means of the Design of Experiments (DOE) approach and a full factorial design. The details are not given due to space constraints but suffice to say that the parameters b, l and a are set to 10, 10 and 40, respectively.

3 Computational evaluation and conclusions

We briefly detail the results of the experimental evaluation of the proposed SS method. We take the 120 instances of Taillard and augment them with different number of factories $F = \{2, 3, 4, 5, 6, 7\}$. This results in a total of 720 instances available at `http://soa.iti.es`. We compare the following methods from the literature: 1) The EM method of Liu and Gao (2010), 2) HGA of Gao and Chen (2011a), 3) The improved NEH of Gao and Chen (2011b), referred to as NEHdf, 4) VNS(B&B) of Gao *et. al.* (2012a), 5) The TS of Gao *et. al.* (2013), 6) The best iterated greedy (IG) algorithm of Lin *et. al.* (2013). 7-10) The existing methods proposed in Naderi and Ruiz (2010), namely NEH1, NEH2, VND(a) and VND(b). Finally, the 11th method compared is the proposed SS method. Some of the previous cited existing algorithms from the literature were not tested as it was clear that their performance was not competitive. All methods have been carefully coded in C++. The stopping criterion of all metaheuristic methods has been changed so that all algorithms use the same CPU time following the expression $n \times m \times F \times C$ where C has been tested at several values: 20, 40, 60, 80 and 100. Testing all methods with 720 instances and with so many stopping times that range from a few seconds to almost two hours guarantees a full range of results. Moreover, all algorithms have been coded in the same language and are run on the same computers with the same CPU time stopping criterion, resulting in comparable computational campaign. NEH1, NEH2 and NEHdf are heuristics and do not have a stopping criterion and therefore are only tested once with each instance. All other 8 methods are tested with the 720 instances and with the 5 aforementioned different stopping times which means that we have $3 \times 720 = 2,160$ results for the heuristics and $8 \times 5 \times 720 = 28,800$ results for the metaheuristics. The total CPU time needed for the metaheuristic results is almost 165 days. For the computational experiments we have used a cluster with 30 computing blades running at 2.5 GHz. and 16 GBytes of RAM memory. The response variable is the average relative percentage deviation (AVRPD) over the best solution known for all 720 instances and replicates.
In a succinct way, NEH1, NEH2 and NEHdf have AVRPD values of 8.31, 5.15 and 4.91, respectively, while the employed CPU times are 0.006, 0.023 and 0.035 seconds, respectively. In some statistical experiments (not shown) it is demonstrated that NEHdf is not statistically better than NEH2 while being slower. For the metaheuristic methods, Table 1 shows the main results. From the summarized results given, it is clear (and statistically significant) that the proposed SS method produces much better results than the competing methods for all tested CPU times.

Table 1. Average Relative Percentage Deviation (AVRPD) and CPU time used (in seconds) for the tested metaheuristics grouped by CPU time limit C.

C	EM	HGA	IG	SS	TS	VND(a)	VND(b)	VNS(B&B)	CPU Time
20	5.15	3.58	2.60	**1.50**	3.18	3.63	3.78	2.68	164.63
40	5.23	3.52	2.39	**1.36**	3.11	3.63	3.78	2.67	329.25
60	5.26	3.49	2.30	**1.27**	3.04	3.63	3.78	2.68	493.88
80	5.17	3.47	2.24	**1.24**	3.05	3.63	3.78	2.68	658.50
100	5.25	3.43	2.16	**1.19**	3.02	3.63	3.78	2.67	823.13

To conclude, we have presented an Scatter Search method for the Distributed Permutation Flowshop Problem that has shown, through a series of comprehensive computational experiments, to be much better than recent proposed competing methods. The proposed SS method incorporates some high-performing characteristics and constitutes the new state-of-the-art for the considered problem.

Acknowledgements

Rubén Ruiz is partially supported by the Spanish Ministry of Economy and Competitiveness, under the project "RESULT - Realistic Extended Scheduling Using Light Techniques" with reference DPI2012-36243-C02-01 co-financed by the European Union and FEDER funds and by the Universitat Politècnica de València, for the project MRPIV with reference PAID/2012/202.

References

Gao, J., Chen, R., 2011a, "A hybrid genetic algorithm for the distributed permutation flowshop scheduling problem", *International Journal of Computational Intelligence Systems*, Vol. 4(4), pp. 497-508.

Gao, J., Chen, R., 2011b. "An NEH-based heuristic algorithm for distributed permutation flowshop scheduling problems", *Scientific Research and Essays*, Vol. 6(14), pp. 3094-3100.

Gao, J., Chen, R., Deng, W., Liu Y., 2012a. "Solving multi-factory flowshop problems with a novel variable neighbourhood descent algorithm", *Journal of Computational Information Systems*, Vol. 8(5), pp. 2025-2032.

Gao, J., Chen, R., Liu, Y., 2012b, "A knowledge-based genetic algorithm for permutation flowshop scheduling problems with multiple factories", *International Journal of Advancements in Computing Technology*, Vol. 4(7), pp. 121-129.

Gao, J., Chen, R., Deng, W. 2013, "An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem", *International Journal of Production Research*, Vol. 51(3), pp. 641-651.

Lin, S.-W., Ying, K.-C., Huang, C.-Y. 2013, "Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm", *International Journal of Production Research*, Vol. 51(16), pp. 5029-5038.

Liu, H., Gao, L., 2010, "A discrete electromagnetism-like mechanism algorithm for solving distributed permutation flowshop scheduling problem", In *Proceedings - 6th International Conference on Manufacturing Automation, ICMA 2010*, Hong Kong, China, pages 156-163. IEEE Computer Society.

Naderi, B., Ruiz, R., 2010, "The distributed permutation flowshop scheduling problem", *Computers & Operations Research*, Vol. 37(4), pp. 754-768.

Wang, S.-Y., Wang, L., Liu, M., Xu, Y., 2013, "An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem", *International Journal of Production Economics*, Vol. 145(1), pp. 387-396.

Multiagent scheduling problems with minmax-type function and number of tardy jobs

F. Sadi and A. Soukhal

University François Rabelais de Tours, France
LI EA 6300, OC ERL CNRS 6305
faiza.sadi,ameur.soukhal@univ-tours.fr

Keywords: scheduling, multiagent scheduling, dynamic programming, ε-constraint.
Abstract: We consider multiagent scheduling problems in which agents (customers, workers, etc.) have to share the same identical parallel machines to execute their jobs without preemption. Each agent has its own objective function (minimizing the number of tardy jobs) which depends only on the schedule of its jobs. In addition, the obtained scheduling sequence of all jobs should minimize a global objective function (f_{max}) introduced by the owner of the system (production manager or resource manager). An ε-constraint approach is used to determine comprise solutions.

Introduction

Scheduling problems in which agents (customers, production managers, etc.) have to share the same set(s) of machines are at the frontier of combinatorial optimization and cooperative game theory ([1]). The key assumption of our models is that each agent has a disjoint set of jobs to perform and each one has its own objective function which only depends on the schedule of its jobs, except for the resource manager who seeks to determine a schedule that optimizes an objective function which depends on the schedule of all jobs. According to [2], this scheduling problem is called *multiagent scheduling problem with a global objective function*. Such problems may be identified in many manufacturing systems. For example, consider a given company which produces different types of products. One type has a due date to be respected (perishable articles) and the production manager wants to minimize jobs in-progress.

Agnetis et al (2004) [1] introduced the multiagent scheduling problems called *Competing scheduling problems*. They considered two disjoint sets of jobs, each one is associated with one agent and one objective function. The jobs have to be executed on the single machine and the goal is to find the solutions of best compromise between the two agents. When

the ε-constraint approach is used, Agnetis et al. proposed an $O(n^3)$ algorithm for problem $1||\sum U^A, \sum U^B$ and an $O(n \log n)$ algorithm for problem $1||\sum U^A, f^B_{max}$. In [4], the authors studied the $1||\sum U^A_j, \sum C^B_j$ problem, and they presented an NP-hardness proof and a dynamic programming algorithm. Recently, Huynh et al (2011) [2] extended these results to interfering job set scheduling problems. For the case of parallel machines, Sadi et al. [5] identified some polynomial and pseudo-polynomial cases.

1 Problem formulation

Let $\mathcal{J}$ be the set of n jobs to be executed without preemption on m identical parallel machines (m is fixed). Each agent k ($k = 1, \ldots, K$) is associated with one job subset $\mathcal{J}^k \subset \mathcal{J}$, such that $|\mathcal{J}^k| = n_k$ and $\forall k \neq k'$ $\mathcal{J}^k \cap \mathcal{J}^{k'} = \emptyset$. Machines are always available and can only process one job at a time. For each job J_j we have p_j the integer processing time and d_j the due date. We assume that all jobs are available at time zero. Let C_j be the completion time of job J_j. We aim at minimizing some monotonically non-decreasing function $f_{max} = \max_{j=1}^{n}\{f_j(C_j)\}$ ($f_j(t)$ depends on J_j) subject to number of tardy jobs of each agent. To obtain a Pareto solution, ε-constraint approach is used. Hence, for a given vector $\varepsilon = (Q^1, \ldots, Q^k, \ldots, Q^K)$ the number of tardy jobs of each agent k does not exceed Q^k (i.e., $\sum_{j \in \mathcal{J}^k} U^k_j \leq Q^k, k = 1, \ldots, K$). According to multicriteria scheduling notation when the ε-constraint approach is used, the studied problems are: (i) Identical processing times, denoted $Pm|p_j = p, \sum U^k_j \leq Q^k, k = 1, \ldots, K|f_{max}$; (ii) Arbitrary processing times, denoted $Pm|\sum U^k_j \leq Q^k, k = 1 \ldots, K|C_{max}$.

2 Identical processing times

For problem $Pm|p_j = p, \sum U^k_j \leq Q^k, k = 1, \ldots, K|f_{max}$, the objective functions are assumed regular. Hence there is no advantage to keep a machine idle. Because processing times are identical, we can easily prove that the last job is completely executed at $t = \lceil \frac{n}{m} \rceil p$ which corresponds to the optimal makespan value C^*_{max}. Without loss of generality, we consider that the due dates are all larger than p. We also admit that n is a multiple of m. Otherwise, we add dummy jobs J_h, such that $f_h(t) = 0$ and $d_h = np$. So, we can show that there exists an optimal solution where these dummy jobs are processed not earlier than the other jobs. Based on the general setting of Lawler's procedure [3], we propose algorithm 1 to determine an optimal solution.

Theorem 21 *Problem $Pm|p_j = p, \sum U^k_j \leq Q^k, k = 1, \ldots, K|f_{max}$ can be solved by algorithm 1 in $O(n^2)$ time.*

Algorithm 1: For problem $Pm|p_j = p, \sum U_j^k \leq Q^k, k = 1, \ldots, K|f_{max}$

Input: n, m, n_k, p, d_j, Q_k, $j = 1, \ldots, n$, and $k = 1, \ldots, K$

```
t = ⌈n/m⌉p, S = J, q_k = 0, π = ∅; (q_k is the number of tardy jobs of agent k)
while S ≠ ∅ do
    i = 1; t = t − p; (i is a machine index)
    while i ≤ m do
        j = arg min{f_j(t) : J_j ∈ J^k and q_k < Q_k};
        Schedule J_j on M_i at time t;
        π = {J_j}\\π; (a\\b stands for the concatenation of a and b)
        if (t + p > d_j) then
            q_k = q_k + 1; (when J_j ∈ J^k)
        i = i + 1; S = S\{J_j};
Return π;
```

3 Arbitrary processing times and $f_{max} = C_{max}$

Let us consider now $f_{max} = C_{max}$. The problem $Pm|\sum U_j^k \leq Q^k, k = 1, \ldots, K|C_{max}$ is $\mathcal{NP}$-hard ($Pm||C_{max}$ is $\mathcal{NP}$-hard). To obtain an optimal solution, we propose a pseudo-polynomial time dynamic programming algorithm. We assume in the following that the jobs are numbered according to their earliest due date order. Under this numbering, there always exists an optimal schedule that verifies: on each machine, the early jobs are processed before the tardy jobs and in their increasing index order.

Dynamic programming. Let UB be the upper bound for the makespan. For each machine i, we introduce two parameters: P_i is the completion time of the last early job scheduled on machine M_i; and t_i is the completion time of the last job assigned to machine M_i (this last job can be tardy). To illustrate our approach, let us consider the case of two machines ($m = 2$). Function $C_j(P_1, t_1, P_2)$ gives the completion time of the last job assigned to machine M_2 when the first j jobs are already scheduled. Depending on the decision of assigning job J_j to machine that minimizes $C_j(P_1, t_1, P_2)$, the number of tardy jobs of agent $Ag(j)$, denoted $U^{Ag(j)}(P_1, t_1, P_2)$ can be deducted.

Whether if J_j is scheduled early or tardy on M_i, the recursive function is:
$C_0(0, 0, 0) = 0; C_0(P_1, t_1, P_2) = +\infty, \forall (P_1, t_1, P_2) \neq (0, 0, 0); U^k(0, 0, 0) = 0;$

$C_j(P_1, t_1, P_2) = +\infty$, if $P_i < 0$ or $t_1 < 0, U^k < 0$ or $U^k > Q^k$

$$C_j(P_1, t_1, P_2) = \begin{cases} +\infty & \text{if } U^{Ag(j)} > Q^{Ag(j)} \\ \min \begin{cases} \min\left(C_{j-1}(P_1 - p_j, t_1 - p_j, P_2); C_{j-1}(P_1, t_1, P_2 - p_j) - p_j\right) & \text{if } P_i \leq d_j \\ \min\left(C_{j-1}(P_1, t_1 - p_j, P_2); C_{j-1}(P_1, t_1, P_2) - p_j\right) & \text{otherwise} \end{cases} \end{cases}$$

According to the best decision for $C_j(P_1, t_1, P_2)$, the number of tardy jobs of agent $Ag(j)$ is updated. It means that if we decide to schedule job J_j tardy on machine M_1 for example, we have: $U^{Ag(j)}(P_1, t_1, P_2) = U^{Ag(j)}(P_1, t_1 - p_j, P_2) + 1$.

Then, the optimal makespan value is given by: $\min_{\forall t_1 \leq UB, \forall P_i \leq UB, i=1,2} \left\{ \max\left(t_1; C_n(P_1, t_1, P_2)\right) \right\}$

In the case of m machines, notice that the worst-case time (and space) complexity needed to compute $C_j(P_1, t_1, P_2)$ is $O(UB^{m-1})$. And for each agent k, the worst-case space complexity needed to save the number of its tardy jobs is $O(n_k UB^{m-1})$.

Theorem 31 *Problem* $Pm|\sum U_j^k \leq Q^k, k = 1\ldots, K|C_{max}$ *can be solved in* $O(nUB^{m-1})$ *time.*

4 Further research

For further research, it would be interesting to develop strong heuristics for $Pm|\sum U_j^k \leq Q^k, k = 1\ldots, K|f_{max}$. The models considered in this study can also be extended in various directions: analyze other combination of criteria and search of the Pareto front. Some approximation schemes can be constructed as well as exact and approximated heuristic methods.

References

1. A. Agnetis, P.B Mirchandani, Pacciarelli. D, and A. Pacifici. Scheduling problems with two competing agents. *Operations research*, (52)229–242, 2004.
2. N. Huynh-Tuong, A. Soukhal, and J.-C. Billaut. Single-machine multi-agent scheduling problems with a global objective function. *Journal of Scheduling*, 15(3):311–321, 2012.
3. E. L. Lawler. Optimal sequencing of a single machine subject to precedence constraints. *Management Science*, 19(5):544–546, 1973.
4. C.T. Ng, T. C. E. Cheng, and J. J. Yuan. A note on the complexity of the problem of two-agent scheduling on a single machine. *Journal of Combinatorial Optimization*, 12(4):387–394, 2006.
5. F. Sadi, A. Soukhal, and J-C. Billaut. Parallel machines multi-agent scheduling problems with a global objective function. *RAIRO-Operations Research, to appear in 2014.*

A new CP-SAT approach for the multi-mode resource-constrained project scheduling problem

Alexander Schnell and Richard F. Hartl

University of Vienna, Austria
alexander.schnell, richard.hartl@univie.ac.at

Keywords: Multi-mode resource-constrained project scheduling, Constraint Programming, Boolean Satisfiability Solving.

1 Introduction

Our research topic is the analysis of new exact approaches for the multi-mode resource-constrained project scheduling problem (MRCPSP).

For variants of the single-mode RCPSP (SRCPSP) recent exact algorithms combine Branch and Bound (BaB)-based search with principles from Constraint Programming (CP), Mixed-Integer Programming (MIP) and Boolean Satisfiability Solving (SAT). Ohrimenko et al. (2009) introduced the CP-SAT hybrid lazy clause generation (LCG) and successfully applied it to the Open Shop Scheduling Problem. Schutt et al. (2009) were the first to generalize LCG to the SRCPSP with the aim of makespan minimization. Up to now, LCG is the state-of-the-art exact approach for the SRCPSP with standard and generalized precedence relations and the objective of makespan minimization and for the SRCPSP with discounted cash flows (See Schutt et al. (2013a), Schutt et al. (2013b) and Schutt et al. (2012)).

With our work, we aim at generalizing the ideas of LCG for the solution of RCPSP instances with multi-mode jobs as it is an efficient approach for variants of the SRCPSP. Therefore, we use the Constraint Integer Programming (CIP)-framework SCIP (See Achterberg (2009)). To generalize the CP-SAT approaches to the MRCPSP, we implemented two new constraint handlers `precedencemm(...)` and `cumulativemm(...)` for SCIP with which one can model standard precedence relations and renewable resource constraints for multi-mode jobs, respectively. The `precedencemm(...)`- and the `cumulativemm(...)`-constraint capture domain propagation rules for the above problem characteristics. Moreover, to enforce the integration of our constraints in the SCIP-intern SAT mechanisms, we also implemented explanation generation algorithms for the above constraints. Explanation generation algorithms have to deliver clauses consisting of boolean literals which represent the reasons for the domain propagation and cutoffs deduced by our constraints (See Schutt et al. (2013a)). All in all, by modeling and solving the MRCPSP with our new constraints `precedencemm(...)` and `cumulativemm(...)` in SCIP, we can solve the latter problem by a BaB algorithm in combination with CP, SAT and MIP techniques which is provided by SCIP and thus realize an approach similar to LCG for the MRCPSP.

2 Modeling and solution principles

A MRCPSP instance consists of a set of jobs $J = \{0, \ldots, n+1\}$, whereas every job $j \in J$ can be processed in a number m_j of modes $k \in M_j = \{1, \ldots, m_j\}$. Note that 0 and $n+1$ are dummy jobs representing the start and the end of the project. Every job's duration $d_{j,k}$ and nonrenewable (renewable) resource consumption $c^{\nu}_{j,k,r},\ r \in N (c^{\rho}_{j,k,r},\ r \in R)$ is dependent on the selected mode $k \in M_j$. Moreover, every nonrenewable (renewable) resource $r \in N$ ($r \in R$) has a maximal capacity of $C^{\nu}_r (C^{\rho}_r)$ units. In addition, every job $j \in J$ has a set

of successors S_j whereas j cannot end after a job from its successor set S_j has started, i.e. we only consider standard precedence relations. As objective, we consider makespan minimization.

For the modeling of the starting time of job i in mode k and the mode assignment of job i, we use the integer variables $s_{i,k}$ and the binary variables $x_{i,k}$. With the above notation, the MRCPSP can be formulated as follows in SCIP:

$$\begin{aligned}
\min \quad & s_{n+1,1} && (1)\\
s.t. \; & \sum_{k \in M_i} x_{i,k} = 1, \quad \forall i \in J && (2)\\
& s_{i,k} + d_{i,k} \cdot x_{i,k} \leq s_{j,l}, \\
& \forall j \in S_i, \forall i \in J, \; \forall k \in M_i, \; \forall l \in M_j && (3)\\
& \sum_{i \in J} \sum_{k \in M_i} c^{\nu}_{i,k,r} \cdot x_{i,k} \leq C^{\nu}_r, \quad \forall r \in N && (4)\\
& \texttt{precedencemm}(\mathbf{s}, \mathbf{x}, \mathbf{d}, \mathbf{S}), && (5)\\
& \texttt{cumulativemm}(\mathbf{s}, \mathbf{x}, \mathbf{d}, \mathbf{c}^r, C^{\rho}_r), \quad \forall r \in R && (6)\\
& s_{j,k} \in \{lb(s_{j,k}), \ldots, ub(s_{j,k})\}, \\
& \forall j \in J, \; k \in M_j && (7)\\
& x_{j,k} \in \{0,1\}, \quad \forall j \in J, \; k \in M_j && (8)
\end{aligned}$$

where

$$\begin{aligned}
& \mathbf{s} = \mathbf{s}^0 \circ \cdots \circ \mathbf{s}^{n+1}, && \text{where } \mathbf{s}^i_k = s_{i,k}, \; \forall k \in M_i, \; i \in J && (9)\\
& \mathbf{x} = \mathbf{x}^0 \circ \cdots \circ \mathbf{x}^{n+1}, && \text{where } \mathbf{x}^i_k = x_{i,k}, \; \forall k \in M_i, \; i \in J && (10)\\
& \mathbf{d} = \mathbf{d}^0 \circ \cdots \circ \mathbf{d}^{n+1}, && \text{where } \mathbf{d}^i_k = d_{i,k}, \; \forall k \in M_i, \; i \in J && (11)\\
& \mathbf{c}^r = \mathbf{c}^{0,r} \circ \cdots \circ \mathbf{c}^{n+1,r}, && \text{where } \mathbf{c}^{i,r}_k = c^{\rho}_{i,k,r}, \forall k \in M_i, \; i \in J, r \in R && (12)
\end{aligned}$$

With (2) and (4), we formulate the uniqueness of the mode-assignments and the nonrenewable resource constraints, respectively. In (5) and (6), we apply the `precedencemm`$(\ldots)$- and the `cumulativemm`$(\ldots)$-constraint to model the multi-mode precedence constraints and renewable resource constraints, respectively. Thereby, we use the variable vectors $\mathbf{s}$ and $\mathbf{x}$ and the parameter vectors $\mathbf{d}$ and $\mathbf{c}^r$ which are given in (9)-(12). In this context, the operator $\circ$ is defined as the concatenation of two vectors, whereas the vector $\mathbf{c} = \mathbf{a} \circ \mathbf{b}$ is obtained by appending the elements of $\mathbf{b}$ coordinate-wise to $\mathbf{a}$. Note that, (3) are valid inequalities to model standard precedence relations between multi-mode jobs. We add the latter to strengthen our model.

The key principle of the domain propagation algorithms captured in the `precedencemm` and the `cumulativemm`$(\ldots)$-constraint is the computation of a single-mode representation from the multi-mode data in the nodes of the BaB tree. We determine a SRCPSP instance from the multi-mode input based on the domains of the mode-assignment variables $x_{j,k}$. With the single-mode data at hand, our constraint handlers can apply standard propagation algorithms for the SRCPSP. Note that, `cumulativemm`$(\ldots)$ integrates an implementation of timetable propagation (See Baptiste et al. (2001)). The explanation generation algorithms are extensions of the ones used by Schutt et al. (2013b) for the SRCPSP. We additionally include information on the binary variables $x_{j,k}$ to represent the multi-mode data which after transformation lead to the domain updates or cutoffs to be explained.

3 Computational Results and Conclusion

We implemented the constraint handlers `precedencemm(...)` and `cumulativemm(...)` with SCIP 3.0.0 in combination with the programming languages `C/C++`. Moreover, we modeled and solved the formulation of Sect. 2 with the latter version of the framework SCIP. We apply our model in two ways which differ in the generation of the initial domains $D_{j,k} = \{lb(s_{j,k}), \ldots, ub(s_{j,k})\}$:

`SCIPMax` $D_{j,k}$ is evaluated by forward (backward) recursion (See Brucker & Knust (2006)) based on the trivial upper bound (UB) $T_{\max} = \sum_{j \in J} d_j^{\max}$, where $d_j^{\max}$ is the maximal mode duration of job j.

`SCIPBest` The initial domains $D_{j,k}^l$ are generated based on 24 different UBs $T_1, \ldots, T_{24}$ where T_1 equals the best known UB from the literature and $T_l = T_{l-1} + 2$, $\forall l = 2, \ldots, 24$. A model is run on the processor $l = 1, \ldots, 24$ with the initial domain $D_{j,k}^l$ based on T_l. Hence, in this case we apply a parallel approach which applies 24 processors. In the end, we take the best results w.r.t. to all 24 processors.

Our approaches are tested on the 552 feasible MRCPSP-instances with 30 jobs (`J30mm`) from the PSPLIB introduced by Kolisch & Sprecher (1997) and on the new 540 MRCPSP-instances with 50 and 100 jobs (`MMLIB50` and `MMLIB100`) provided by the Operations Research and Scheduling research group of the University of Ghent (2011). We use a time limit of 3600s, 5400s and 7200s for the instance sets `J30mm`, `MMLIB50` and `MMLIB100`, respectively. We compare `SCIPMax` and `SCIPBest` to the state-of-the-art exact approach of Zhu et al. (2006) (`MMBAC`) on the instances `J30mm` (See Table 1). Table 1 shows the number of instances solved to optimality (#opt), the number of best known solutions detected (#best), the average solution time (t_{tot}) and the percentage gap to the critical path lower bound ($G_{LB}(\%)$). The results of our approaches are highly competitive but still inferior to the results obtained by Zhu et al. (2006). However, as their Branch-and-Cut approach is based on a MIP formulation of the MRCPSP, it is highly dependent of a starting solution with a small makespan to reduce the number of binary variables. They use starting solutions computed by a problem specific heuristic whose makespan on average only deviates by 2.18% from the best known makespans of the PSPLIB. Our model still produces solutions of useful quality in comparable average solution times when using the trivial upper bound $T_{\max}$ (`SCIPMax`). Moreover, if we only integrate UBs which have a relative deviation of more than 18 units w.r.t. the best known makespan, we obtain competitive results compared to `SCIPBest` (See column $\texttt{SCIPBest}_{\geq 18}$ in Table 1). This corresponds to an average deviation of more than 57% over all 30-job instances. Thus, our approach can also produce high quality solutions without a good starting solution at hand.

	#opt	#best	t_{tot}	$G_{LB}(\%)$
`SCIPMax`	488	502	465.28	13.52
$\texttt{SCIPBest}_{\geq 18}$	500	513	413.52	12.82
`SCIPBest`	502	517	393.12	12.77
`MMBAC`	506	529	393.13	-

Table 1. 30 Jobs

Tables 2 and 3 show the results for the MRCPSP-instances with 50 and 100 jobs. To our knowledge, we are the first to exactly solve these instances. We are able to prove optimality for $\approx 77\%$ (57%) of the 50(100)-job instances. Moreover, we improved the best known makespans for 70 (93) instances with 50 (100) jobs (See column #Imp).

	#opt	#best	t_{tot}	$G_{LB}(\%)$	#Imp
SCIPMax	400	414	1497.65	41.81	57
SCIPBest	417	444	1329.57	25.96	70

Table 2. 50 Jobs

#opt	#best	t_{tot}	$G_{LB}(\%)$	#Imp
272	301	3776.05	389.71	77
310	355	3280.41	32.83	93

Table 3. 100 Jobs

To conclude, we successfully generalized the state-of-the-art exact solution principles for the SRCPSP to the MRCPSP by introducing two new constraint handlers `precedencemm(...)` and `cumulativemm(...)` for the framework SCIP. In addition, the proposed model can be transformed to variants of the MRCPSP with more general linear constraints like e.g. generalized precedence relations or time-dependent resource profiles and also more general objective functions like e.g. the net present value. However, there is still ample room for an improvement of our approach. At the moment, our implementation of the `cumulativemm(...)`-constraint only contains a timetable propagation algorithm (See Baptiste et al. (2001)) for multi-mode jobs. We plan to integrate stronger propagation algorithms like e.g. edge finding or energetic reasoning in the latter constraint.

Acknowledgements

We would like to thank the SCIP team and especially Stefan Heinz and Jens Schulz for their valuable information about the implementation of constraint handlers.

References

Achterberg, T. (2009), 'Scip: solving constraint integer programs', *Mathematical Programming Computation* **1**(1), 1–41.

Baptiste, P., Le Pape, C. & Nuijten, W. (2001), *Constraint-based scheduling: applying constraint programming to scheduling problems*, Vol. 39, Springer Netherlands.

Brucker, P. & Knust, S. (2006), *Complex scheduling*, Springer Verlag.

Kolisch, R. & Sprecher, A. (1997), 'Psplib-a project scheduling problem library', *European Journal of Operational Research* **96**(1), 205–216.

Ohrimenko, O., Stuckey, P. & Codish, M. (2009), 'Propagation via lazy clause generation', *Constraints* **14**(3), 357–391.

Operations Research and Scheduling research group of the University of Ghent (2011), 'The multi-mode resource-constrained project scheduling problem', `http://www.projectmanagement.ugent.be/?q=research/project_scheduling/multi_mode`.

Schutt, A., Chu, G., Stuckey, P. & Wallace, M. (2012), Maximising the net present value for resource-constrained project scheduling, *in* N. Beldiceanu, N. Jussien & É. Pinson, eds, 'Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems', Vol. 7298 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 362–378.

Schutt, A., Feydy, T. & Stuckey, P. (2013a), Explaining time-table-edge-finding propagation for the cumulative resource constraint, *in* C. Gomes & M. Sellmann, eds, 'Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems', Vol. 7874 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 234–250.

Schutt, A., Feydy, T., Stuckey, P. J. & Wallace, M. G. (2013b), 'Solving rcpsp/max by lazy clause generation', *Journal of scheduling* **16**(3), 273–289.

Schutt, A., Feydy, T., Stuckey, P. & Wallace, M. (2009), Why cumulative decomposition is not as bad as it sounds, *in* I. Gent, ed., 'Principles and Practice of Constraint Programming - CP 2009', Vol. 5732 of *Lecture Notes in Computer Science*.

Zhu, G., Bard, J. & Yu, G. (2006), 'A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem', *INFORMS Journal on Computing* **18**(3), 377–390.

A genetic algorithm for solving the flexible resource constrained project scheduling problem under different objectives

Torben Schramme[1] and Leena Suhl[1]

University of Paderborn, DS&OR Lab, Germany
schramme, suhl@dsor.de

Keywords: Resource constrained project scheduling, flexible resources, genetic algorithm.

1 Introduction and motivation

In the primal Resource Constrained Project Scheduling Problem (RCPSP) each resource allocation for an activity is constant during all periods of its execution. However, in some practical applications, being restricted to these rigid resource profiles is not satisfying, so in the past researchers developed more flexible planning models to incorporate the resource allocation and the activity duration as variables in the decision problem (cf. e.g. (Kolisch *et. al.* 2003), (Fündeling 2001)).

Previous researchers have shown that this flexible RCPSP (FRCPSP) is even harder to solve by Mixed Integer Programming (MIP) than the standard RCPSP, so they finally adapted the very fast and well-known activity lists with schedule generation schemes (SGS) for solving schedules (cf. e.g. (Fündeling and Trautmann 2010), (Zhang and Sun 2011)). That led to reasonable computation times with the drawback of being settled to regular objective functions (e.g. C^{Max}). For nonregular objectives as used for example for robust scheduling or rescheduling, there is no known heuristic for the FRCPSP.

In this work, we combine the for the FRCPSP modified serial SGS from Kolisch *et. al.* (2003) with the genetic algorithm (GA) from Hartmann (1998) to create a competitive solution method for the FRCPSP. In addition, we extend that serial SGS and the representation of the GA to enable this new algorithm to support various time-based objective functions for single and multi-project schedules. To show its effectiveness, we provide the results of two experiments. First, we consider the C^{Max} objective on a single project test set with flexible resource demands. Second, we consider multi project instances with flexible resource demands and solve them under the objectives of minimizing activities tardiness, and minimizing tardiness combined with maximizing robustness, to demonstrate the ability of handling various objectives. Overall, all results are compared with those of a standard MIP approach and (for the first two objectives) with the random sampling approach of Kolisch *et. al.* (2003) and with several priority rules of Fündeling and Trautmann (2010).

2 Problem statement

We consider the flexible resource constrained project scheduling problem (FRCPSP), the decision variables are the finish times $f_i \in \mathbb{N}_0^+$ and the durations $d_i \in \mathbb{N}_0^+$ of each activity i and the resource allocation $r_{ikt} \in \mathbb{R}_0^+$ for each demand for resource k by activity i for all execution periods $t \in \{f_i - d_i, \ldots, f_i - 1\}$. Furthermore we define $s_i = f_i - d_i$ as an auxiliary variable for the starting time.

Activities are non-preemptable and connected by end-to-start precedence relations in an activity on node (AoN) network. The allowed duration is limited by the constraint $\underline{d_i} \leq d_i \leq \overline{d_i}$. A release time t^R and/or a due time t^D may be assigned to selected activities.

The activity network is comprised by two additional dummy activities, one at the start (no inbound arcs) and another at the end (no outbound arcs), both with no resource demands and a duration of 0. If a schedule contains multiple projects, the activities of each project are comprised by additional dummy activities which connect the first resp. last activities of the project with the first resp. last dummy activity of the whole schedule network.

Only renewable, continuous resources are considered with an availability of $a_{kt} \in \mathbb{R}_0^+$ for each resource k and each time period t. Each nondummy activity can have several resource demands. The actual resource allocation r_{ikt} for a demand for resource k by activity i is restricted by $\underline{r_{ik}} \leq r_{ikt} \leq \overline{r_{ik}}$ for $t \in \{s_i, \ldots, f_i - 1\}$. A resource demand may be constant as in the classical RCPSP or flexible. For constant resource demands, $\underline{r_{ik}} = r_{ikt} = \overline{r_{ik}}$ holds so that the amount of allocated resources is equal in all execution periods of the activity. For flexible resource demands, $\underline{r_{ik}} < \overline{r_{ik}}$ applies and the constraint $\sum_{t=s_i}^{f_i-1} r_{ikt} = w_{ik}$ is added to enforce the total requested workload w_{ik}. If an activity i has only constant demands, $\underline{d_i} = \overline{d_i}$ should be set in the input data to make also the activity duration constant.

In this work, this problem will be solved under three different objectives:

O1 is the classical $C^{\max}$ function for minimizing the makespan of a single project schedule.

O2 is the weighted tardiness objective function $\sum \max(0, f_i - t_i^D) * w_i^D$ for minimizing the tardiness of all activities with a due date t_i^D and a weight factor w_i^D.

O3 is a combination of two objectives. For minimizing tardiness, we define $O3_1 = O2$. For robust optimization we define $O3_2 = \sum_{i \in B} b_i$, where b_i denotes the buffer time behind the end of activity i and the begin of its earliest successor, and B denotes a set of selected activities behind whom we want to have buffer time. The final objective is defined by minimizing $O3 = O3_1 - 0.01 * O3_2$ so that minimizing activity lateness is much more important than maximizing the buffer time.

3 Constructing the initial solution set

Due to the fact that a genetic algorithm is for improving existing solutions, the first step of solving the FRCPSP is to generate a set of initial valid solutions. By fixing d_i to a value between $\underline{d_i}$ and $\overline{d_i}$ and fixing $r_{ikt} = \frac{w_{ik}}{d_i}$ for $t \in \{s_i, \ldots, f_i - 1\}$, we obtain a possible constant solution for each flexible demand (if no such fixing exists, there would be no other possible setting for the flexible demand). Assuming that the overall planning horizon is long enough and $r_{ikt} \leq a_{kt}$ always holds, there is a feasible solution with only constant demands and durations so the problem can be solved with standard RCPSP methods. We use randomized priority lists with serial SGS (as described by Demeulemeester and Herroelen (2002)) to generate an initial set of solutions.

4 Modifying the serial SGS for various time-based objectives

The set of all possible placements for an activity i can be defined as

$$P_i = \left\{(s_i, d_i) | s_i \in \{EST_i, \ldots, LST_i\} \wedge d_i \in \left\{\underline{d_i}, \ldots, \overline{d_i}\right\} \wedge s_i + d_i \leq LFT_i\right\}$$

where EST_i, LST_i, LFT_i are the earliest resp. latest possible start times and the latest possible finish time of an activity i according to critical chain calculation. Furthermore, we define an activity fitness function $k_i : (s_i, d_i) \rightarrow \mathbb{R}$ to assign each possible placement a fitness value. The original serial SGS, and also the extended version of Kolisch *et. al.* (2003) for flexible resource demands, which is the foundation in this work, try to schedule an activity as early as possible with respect to its predecessors and the resource availability. We adapted this behavior in the following way: assuming that $k_i(s_i^1, d_i^1) > k_i(s_i^2, d_i^2)$ always

holds when (s_i^1, d_i^1) is a more desirable placement than (s_i^2, d_i^2), and k_i^* denotes the fitness value of the most desirable placement, then the SGS tries to schedule i to the placement (s_i, d_i) with the lowest value of $|k_i^* - k_i(s_i, d_i)|$ which is resource and precedence feasible. To schedule an activity i as early as possible, we define $k_i : (s_i, d_i) \rightarrow -s_i$ and set $k_i^* = -EST_i$. In this modified SGS we assume this as default for all activities.

To support a certain time-based objective function, we now identify the set ϕ of activities whose placements have direct influence on the objective value, and for each $i \in \phi$ we adjust k_i and k_i^*. For objective O1 the only measured activity is the last activity in the network and it shall be placed as early as possible, so because of the defaults no adjustment is needed. For O2, ϕ contains all activities which have a due date, $k_i : (s_i, d_i) \rightarrow -\min(0, f_i - t^D)$ and $k_i^* = 0$. For the robust optimization part of O3, we extend ϕ of O2 by all succeeding activities j of each activity $i \in B$ and set their $k_j : (s_j, d_j) \rightarrow s_j$ and $k_j^* = LST_j$ to schedule them as late as possible.

5 Solution representation for the genetic algorithm

A genetic algorithm is used to improve the initial set of solutions towards the selected objective function. Each individual I_p within the population P represents a possible solution of the decision problem and is defined as $I_p = (V, K^*, m, O)$.

Tuple $V = (v_1, \ldots, v_N)$ assigns each activity i a continuous priority value $v_i \in [0, 1]$. This priority value is used to determine the scheduling order of the activities for the serial SGS. Given a set C containing all candidate activities that could be scheduled next because all of their predecessors have already been scheduled, we take the activity $i \in C$ with $v_i = \max\{v_j | j \in C\}$. Hartmann (1998) tested this random key encoding in his GA but rejected it in his tests in favor of a slightly better list representation. We tested both representations and came to the conclusion that priority values perform better in our case.

Tuple $K^* = (k_1^*, \ldots, k_N^*)$ contains the current desired fitness value (cf. section 4) for each activity. $\forall j \notin \phi : k_j^* = -EST_j$ to schedule them always as early as possible. $\forall j \in \phi$, we first set $k_j^* = k_j(s_j, d_j)$ where (s_j, d_j) is the placement of the activity in the initial solution. In further steps the algorithm will try to improve this value by mutation to achieve perhaps better objective values. $m \in \phi$ denotes the activity whose k_m^* can be subject to mutation in the next iteration. Thus, when an improvement of one k^* results in a better objective value, the information which k^* was modified is inherited to the next generation.

Tuple $O = (d_1^O, \ldots, d_N^O)$ contains offsets for the minimum duration for each activity. That is, the minimum allowed duration for an activity i in the SGS is set to $\underline{d_i} + d_i^O$ where $d_i^O \in \{0, \ldots, \overline{d_i} - \underline{d_i}\}$ to enforce the elongation of the activity and thus lower the greedy resource allocation per time period.

6 Crossover, mutation, and selection operators

In each iteration, the GA first doubles its population P by successively creating two new individuals I_1', I_2' by taking randomly two individuals $I_1, I_2 \in P$ and applying the two-point crossover operator (cf. (Hartmann 1998)) to them. Second, several mutation operators are applied to all new individuals. Last, the modified SGS is used to calculate the fitness of all new individuals and then, the n best individuals out of all old and new individuals are selected for the next generation so that P regains its original size. For mutation, first each $v_i \in V$ is replaced by a random new value with a chance of 5%. Second, each $d_i^O \in O$ is increased or decreased by 1 with a chance of 5%. Third, m is changed to any other activity $\in \phi$ with a chance of 10%. Last, also with a chance of 10%, $k_m^* \in K$ is replaced by $k_m'^* = \min\{k_m(s_m, d_m) | k_m(s_m, d_m) > k_m^*\}$, if such exists.

Table 1. Arithmetic average results over all instances and 3 runs/instance

Algorithm	Objective O1		Objective O2		Objective O3	
	⌀ Gap	# MIP$^{\leq}$	⌀ Gap	# MIP$^{\leq}$	⌀ Gap	# MIP$^{\leq}$
GA (30 seconds)	-0.5%	100/150	11.1%	14/50	6.4%	15/50
Random Sampling (30 seconds)	0.9%	84/150	16.0%	14/50	n/a	n/a
GA (300 seconds)	-3.3%	111/150	- 8.7%	18/50	4.4%	16/50
Random Sampling (300 seconds)	-0.3%	95/150	8.2%	16/50	n/a	n/a
PR: Longest path following	1.4%	89/150	45.1%	5/50	n/a	n/a
PR: Most total successors	5.1%	56/150	52.9%	3/50	n/a	n/a
PR: Most work content remaining	5.2%	61/150	53.7%	3/50	n/a	n/a

7 Computational results

The GA has been tested on 150 single project instances (30-90 activities) for O1, and 50 multi project instances (50-115 activities on 3-5 projects) for O2 and O3. The projects of the instances were taken from PSPLIB instances, a subset of resource demands were flexibilized randomly, resource availabilities were combined with a random factor between 50%-85%, and due dates were set randomly in the range of either 70-100% or 100-140% of the best known project length (depending on whether the due date should be tight or not). Table 1 shows the results of the GA and several other heuristics (for a description of the priority rules, see (Fündeling and Trautmann 2010)). Column "⌀ Gap" denotes the average gap of the heuristic objective value to the best known MIP objective value (solved by Gurobi 5.6 on 4 cores, time limit = 1 hour). Column "# MIP$^{\leq}$" denotes the number of instances whose heuristic objective value is better or equal to the MIP objective value.

8 Conclusion

We could show that the presented GA is superior to all known heuristic solution methods for the FRCPSP with the C^{Max} objective. In addition, for minimizing tardiness or increasing robustness, in average it leads to better results in shorter time than a MIP approach. The next step we recommend is to test other objectives, like moving an activity into a certain time window for rescheduling, or optimizing the net present value of a schedule.

References

Demeulemeester, E. L., Herroelen, W. S., 2002, "Project Scheduling - A Research Handbook", *International Series in Operations Research & Management Science, Kluwer Academic Publishers*, pp. 203-342.

Fündeling, C.-U., 2006, "Ressourcenbeschränkte Projektplanung bei vorgegebenen Arbeitsvolumina", *PhD thesis, Universität Karlsruhe.*

Fündeling, C.-U. and Trautmann, N., 2010, "A priority-rule method for project scheduling with work-content constraints", *European Journal of Operational Research*, Vol. 203, pp. 568-574.

Hartmann, S., 1998, "A competitive genetic algorithm for resource-constrained project scheduling", *Naval Research Logistics (NRL)*, Vol. 45, pp. 733-750.

Kolisch, R., Meyer, K., Mohr, R., Schwindt, C., Urmann, M., 2003, "Ablaufplanung für die Leitstrukturoptimierung in der Pharmaforschung", *Zeitschrift für Betriebswirtschaft*, Vol. 73.2003, pp. 825-848.

Zhang, L., Sun, R., 2011, "An improvement of resource-constrained multi-project scheduling model based on priority-rule based heuristics", *2011 8th International Conference on Service Systems and Service Management (ICSSSM)*, pp. 1-5.

Single Machine Scheduling with Nonmonotonic Piecewise Linear Time Dependent Processing Times

Helmut A. Sedding and Florian Jaehn

University of Augsburg, Germany
helmut.sedding@wiwi.uni-augsburg.de, florian.jaehn@wiwi.uni-augsburg.de

Keywords: Single Machine Scheduling, Time Dependent Processing Times.

1 Introduction

In this work, we introduce a single machine scheduling problem, in which the processing time of a job increases when its start time deviates from a given job dependent ideal time. This problem differs from similar problems found in literature by the fact that most time dependent problems only allow monotonic change in processing time. Its application lies in scheduling of a single worker who successively picks up components and takes them to a moving conveyor belt, for assembling them into a product. We solved the problem by developing a branch and bound procedure.

Consider a conveyor line which moves a product with a constant speed along a straight line. The assembly consists of several independent jobs that need to be scheduled. We focus on a single worker only. In each of his or her jobs, he or she first picks up a component, then builds it into the product. Every part has its own fixed storage location along the conveyor line. Thus, the worker needs to walk from the product to the storage location and back. As this distance depends on the current product position, it is time dependent. Minimum distance occurs at the point in time when the product passes the storage location, to which we will refer as the ideal time. As the distance increases both before and after the ideal time, it is a nonlinear function of time. We assume a constant walking speed that is faster than the conveyor speed. Then, the walking time equals distance divided by speed. This walking time is added to the assembly time of a job, thus the processing time is nonlinearly time dependent.

Scheduling with time dependent processing times has increasingly been studied in the last two decades. Often considered are learning or deterioration effects, sometimes both, e.g. Biskup (2008), Janiak, Krysiak & Trela (2011), as well as controllable processing times with influence on the cost (Shabtay & Steiner 2007). Monotonous functions of time, either increasing or decreasing, are most common (Gawiejnowicz 2008, p. 51), especially linear or piecewise linear cases (Alidaee & Womer 1999, Cheng, Ding & Lin 2004). What has been studied to a lesser extent are convex functions that are both decreasing and increasing over time. These could be used to model our application, because the walking time is minimum at the ideal time, and increases when deviating in both directions.

One of the simplest objective functions in scheduling is to minimize the completion time of all jobs, namely $C_{\max}$. In our application, this leads to a maximization of the number of products assembled per time frame. Therefore, minimizing $C_{\max}$ matches the application case.

2 Problem Description

Given a set of n jobs j, $1 \leq j \leq n$. Each job j has an ideal start time $T_j \in \mathbb{R}^+$ and a base length $l_j \in \mathbb{R}^+$. We define its processing time as $p_j(t) = l_j + a\,|t - T_j|$, a function on the job starting time t. A given penalty factor $0 < a < 1$, common for all jobs,

reveals the processing time increase when t deviates from T_j. A schedule orders the jobs consecutively such that the start time of a job corresponds to the completion time of the preceding job. The objective is to find a schedule that minimizes the completion time for the last job, $C_{\max}$. Note that there are no further restrictions on the order of jobs so that each sequence is feasible. We can denote this problem as $1\big|p_j(t) = l_j + a\,|t - T_j|\big|C_{\max}$ in standard three-field notation (Graham, Lawler, Lenstra & Rinnooy Kan 1979).

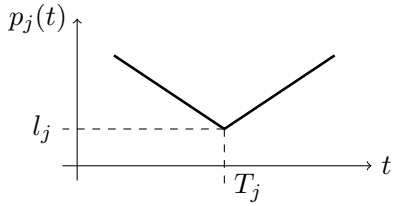


Fig. 1. The processing time $p_j(t)$ of a job j depends on its start time t, with ideal start time T_j.

Looking at the problem, we see that the processing time $p_j(t)$ is a piecewise linear function of time, see also Figure 1. It is also nonmonotonic because before T_j it decreases, and after T_j it increases. Also, it is a convex function, as is the absolute value function.

Coming from the application, the factor a states the relation between the conveyor speed and the worker speed. The factor a must lie in the interval between 0 and 1. Note that with $a = 0$, the problem would become trivial as it is no longer time dependent. Also, this would mean a nonmoving conveyor. With $a \geq 1$, the worker would walk the same speed or even slower than the conveyor moves, which is an unlikely scenario. Also, this would not permit to start a job after its ideal time, because the worker would never be able to reach back to the product. We assume the same factor a on both sides of T_j, because this simplifies formulating the problem, though we are aware that this assumption needs to be dropped in future research as the conveyor moves during walking.

As the time duration increases when scheduling a job not at its ideal time, we observe some similarity of our problem to just in time scheduling, where it is costly to schedule a job not at its due date. Especially the single machine earliness and tardiness minimization problem is similar, as the costs increase linearly when scheduling a job apart its due date (Abdul-Razaq & Potts 1988, Bauman & Józefowska 2006, Billaut & T'kindt 2006). Thus, jobs should be scheduled near their due dates. This is similar to our problem, but instead of billing the deviation as costs, we include it within the processing times, thereby influencing other jobs.

Because the position in the schedule determines a job's deviation from its ideal time, thereby changing the processing time, common proof techniques are hard to use. Especially the pairwise interchange argument for obtaining precedence relations is hard to apply, because when swapping two jobs, other jobs move along and change their processing times. Therefore is it difficult to ensure that the objective function is not worsened, rendering this often used proof technique to be complicated.

That the given problem is very hard to solve is underlined by the fact that a very similar nonlinear problem is also very difficult, namely the problem presented by Kunnathur & Gupta (1990). Here, the processing time is constant until its due date, and then grows by a linear factor. They solve the problem by heuristics and a branch and bound algorithm. Later on, Kononov (1997) showed that this related problem is strongly NP-hard.

3 Branch and Bound Algorithm

To solve this problem exactly, we present a branch and bound algorithm that minimizes the objective $C_{\max}$. To implement it, we need both upper and lower bounds for $C_{\max}$.

3.1 Special Cases of the Problem

To derive bounds, we set the objective to minimize the makespan $C_{\max} - t_{\min}$.

Case (S1): Say we only allow jobs to start after a certain start time t_S, and all the ideal times lie before t_S, meaning $T_j \leq t_S$ for all jobs j. Then, the makespan is minimum when the jobs start at t_S, non-decreasingly ordered by the value of $p_j(t_S)$.

Case (S2): Is like case (S1), but $T_j \geq t_E$ for a certain end time t_E and all jobs j. Then, scheduling the jobs non-increasingly by the value of $p_j(t_E)$, and such that the last job completes at t_E, delivers a minimum makespan.

For a proof of (S1) we refer to e.g. Kunnathur & Gupta (1990, Lemma 1), and as (S2) is symmetric to (S1), the proof is similar.

3.2 Lower Bound

To calculate a lower bound, we relax the problem definition by allowing the parallel execution of jobs. Now, each job can be started at its ideal time, where its processing time will be minimum. Then, the sum of individual processing times leads to a first lower bound.

We improve this lower bound as soon as we are able to state bounds to the time interval where the jobs are scheduled in, denoted by the interval (t_S, t_E). Then we can apply the presented special cases (S1) and (S2) to plan jobs j that have T_j before t_S or after t_E, which results in a much higher lower bound for the processing times.

3.3 Upper Bound

As an upper bound, we devised the following heuristic. First, it generates a random job order. Then, it repeatedly interchanges consecutive jobs until a local optimum is reached. This local search is similar to a heuristic for the earliness and tardiness minimization problem, where it proved to perform quite well (Sourd 2006).

3.4 Branch and Bound Procedure

We utilized our findings to develop a branch and bound procedure. The heuristic initializes the upper bound, this also gives a value for t_E, as there is obviously no possible way for a job to start after the upper bound.

Then, a depth first search runs until it finishes with the optimum $C_{\max}$. For this, jobs are successively appended to the end of the schedule. The variable t_S is initially set to 0 and is afterwards always set to the completion time of the last job of the partial schedule. Should the lower bound exceed the upper bound, a branch is fathomed. A branch completes when all jobs are scheduled, updating the upper bound and t_E to the new best $C_{\max}$.

4 Conclusion

A single machine scheduling problem with time dependent processing times is analyzed for minimizing maximum completion time $C_{\max}$. We present a branch and bound procedure, with a lower bound which uses the fact that the problem is polynomially solvable in two presented special cases. Given the stated practical relevance of the problem, we recommend further development of fast solution algorithms for this problem type.

References

Abdul-Razaq, T. S. & Potts, C. N. (1988), 'Dynamic programming state-space relaxation for single-machine scheduling', *The Journal of the Operational Research Society* **39**(2), 141–152.

Alidaee, B. & Womer, N. K. (1999), 'Scheduling with time dependent processing times: Review and extensions', *The Journal of the Operational Research Society* **50**(7), 711–720.

Bauman, J. & Józefowska J. (2006), 'Minimizing the earliness-tardiness costs on a single machine', *Computers & OR* **33**(11), 3219–3230.

Billaut, J.-C. & T'kindt V. (2006), *Multicriteria Scheduling*, 2nd edn, Springer, Berlin and Heidelberg.

Biskup, D. (2008), 'A state-of-the-art review on scheduling with learning effects', *European Journal of Operational Research* **188**(2), 315–329.

Cheng, T. C. E., Ding Q. & Lin B. M. T. (2004), 'A concise survey of scheduling with time-dependent processing times', *European Journal of Operational Research* **152**(1), 1–13.

Gawiejnowicz, S. (2008), 'Time-dependent scheduling', *Monographs in theoretical computer science, Springer*, Berlin and Heidelberg.

Graham, R. L., Lawler E. L., Lenstra J. K. & Rinnooy Kan, A. H. G. (1979), 'Optimization and approximation in deterministic sequencing and scheduling: a survey', *Annals of Discrete Mathematics* **5**, 287–326.

Janiak, A., Krysiak T. & Trela R. (2011), 'Scheduling problems with learning and ageing effects: a survey', *Decision Making in Manufacturing and Services* **Vol. 5, no. 1-2**, 19–36.

Kononov, A. V. (1997), 'On schedules of a single machine jobs with processing times nonlinear in time', *in* 'Operations Research and Discrete Analysis', *Mathematics and Its Applications* **391**, pp. 109–122.

Kunnathur, A. S. & Gupta S. K. (1990), 'Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem', *European Journal of Operational Research* **47**(1), 56–64.

Shabtay, D. & Steiner G. (2007), 'A survey of scheduling with controllable processing times', *Discrete Applied Mathematics* **155**(13), 1643–1666.

Sourd, F. (2006), 'Dynasearch for the earliness–tardiness scheduling problem with release dates and setup constraints', *Operations Research Letters* **34**(5), 591–598.

Variable Neighborhood Search for a Scheduling Problem with Time Window Penalties

Thevenin Simon[1], Zufferey Nicolas[1]

Geneva School of Economics and Management, University of Geneva, Switzerland.
simon.thevenin@unige.ch n.zufferey@unige.ch

Keywords: metaheuristics, job selection, earliness and tardiness penalties.

1 Introduction

We consider an NP-hard single machine scheduling problem, denoted (P). It takes into account: earliness and tardiness penalties, sequence dependent setup times and costs, and rejection penalties associated with the non selected jobs. Due to the desire of clients to customize the goods, there is a trend in today's companies to adopt a make-to-order production system. This is, for instance, the case in computers, cars, bags, and watches factories (Mansouri *et al.* 2012). The just-in-time paradigm comes then naturally as a way to reduce inventory costs. Depending on the sale politic and on the company's organization, scheduling operations at the workshop level in such environments can be characterized by the need to reject some orders (Oğuz *et al.* 2010), and the use of earliness penalties. An application of such a problem to a numerically controlled machine can be found in Atan and Akturk (2008). Despite the practical importance of such problems, and the abundant literature on scheduling with rejections and on scheduling with earliness and tardiness penalties, only a few papers take into account these two elements (e.g., Atan and Akturk (2008), Shabtay *et al.* (2012)). All these considerations motivate us to propose an efficient metaheuristic to tackle (P), namely a variable neighborhood search (*VNS*).
VNS has been applied to single machine problems in different scheduling environments (e.g., Liao and Cheng (2007)). When all jobs must be scheduled, the neighborhood structures are generated by slightly changing the production sequence (often with moves REINSERT or SWAP as defined in Section 3). In this work, we propose to strategically use the move DROP, consisting in rejecting some jobs. This paper is a sequel of the work presented in Thevenin *et al.* (2013), where the authors propose a tabu search (*TS*) for a problem similar to (P). Readers are referred to Gendreau and Potvin (2010) and Zufferey (2012) for more information on metaheuristics (e.g., tabu search, variable neighborhood search).
Problem (P) is formally described in Section 2. Based on Thevenin *et al.* (2013), Section 3 presents the existing *TS* for (P). Section 4 proposes a *VNS* method for (P). Experiments are discussed in Section 5. Finally, a conclusion ends up the paper.

2 Formal description of problem (P)

Problem (P) is formally stated bellow (Thevenin *et al.* 2013). A set of n jobs is given, and a subset of these jobs have to be selected and scheduled on a single machine, which can handle only one job at a time. For each job j, the following data are given: a processing time p_j, an available date $\bar{r_j}$, a release date r_j, a due date d_j, a deadline $\bar{d_j}$, and a rejection penalty u_j. Let C_j and S_j, respectively, denote the completion time and the starting time of job j. In a feasible solution, each accepted job j satisfies $C_j \leq \bar{d_j}$ and $S_j \geq \bar{r_j}$. Then, for each accepted job j which is not fully performed within time window $[r_j, d_j]$, a penalty must be paid. More precisely, an earliness penalty $E_j(S_j)$ is incurred for each job j started before its release date ($S_j < r_j$), and a tardiness penalty $T_j(C_j)$ has to be paid for each job j which

ends after its due date ($C_j > d_j$). Note that $E_j(\cdot)$ (resp. $T_j(\cdot)$) are non increasing (resp. decreasing) functions. Also, between two consecutive jobs j and j' of different families, a setup time $s_{jj'}$ must be performed, and a setup cost $c_{jj'}$ is incurred. Preemptions are not allowed, and it is possible to insert idle times in the schedule.
A solution s is modeled by a set $\Omega(s)$ of rejected jobs and a sequence $\sigma(s)$ of accepted jobs. To compute the earliness and tardiness penalties associated with s, a starting and an ending time must be assigned to each job of $\sigma(s)$. Due to the possibility to insert idle times in the schedule, this is a complex task accomplished by a so-called *timing procedure* (Thevenin *et al.* 2013). The objective function is given in Equation (1), where $p_s(j)$ represents the predecessor of job j in $\sigma(s)$. Note that the predecessor of the first job is a dummy job, which represents the initial state of the machine.

$$f(s) = \sum_{j \in \sigma(s)} \left[E_j(S_j) + T_j(C_j) + c_{p_s(j)j} \right] + \sum_{j \in \Omega(s)} u_j \tag{1}$$

3 Tabu Search (TS)

Tabu search is a local search metaheuristic. Starting from an initial solution, it generates at each iteration a *neighbor* solution s' from the *current* solution s. The set $N(s)$ of neighbors of s is obtained by performing *moves* on s, which are slight modifications of the solution structure. To avoid cycling, a *tabu list* forbids to perform the reverse of recently performed moves. The best non tabu move is generally performed at each iteration.
In TS proposed for (P) in Thevenin *et al.* (2013), the initial solution is generated by a greedy procedure *GR*. In *GR*, jobs are sorted by increasing slack times ($\bar{d}_j - \bar{r}_j - p_j$), and inserted one by one in the solution at the position which minimizes the costs (but a job is put in $\Omega(s)$ if it is cheaper than to perform it). In TS, four types of moves were proposed for (P): ADD takes a job from $\Omega(s)$ and inserts it in $\sigma(s)$; DROP removes a job from $\sigma(s)$ and puts it in $\Omega(s)$; REINSERT changes the position of one job in $\sigma(s)$; SWAP exchanges the positions of two jobs in $\sigma(s)$. Four different tabu structures were designed. The first (resp. second) forbids to add (resp. drop) a dropped (resp. added) job during τ_1 (resp. τ_2) iterations. The third forbids to move a job which has been added, reinserted or swapped, during τ_3 iterations. The fourth forbids to move a job j between its two previous neighboring jobs during τ_4 iterations, if j has been reinserted or swapped. The τ_i's were tuned as follows: $(\tau_1, \tau_2, \tau_3, \tau_4) = (20, 30, 20, 80)$ for instances with more than 50 jobs, and $(\tau_1, \tau_2, \tau_3, \tau_4) = (20, 20, 15, 25)$ otherwise. At each iteration, only a random proportion $r\%$ (parameter tuned to 30) of the neighbor solutions are evaluated.

4 Variable neighborhood search (VNS)

VNS is a local search method which sequentially uses different neighborhood structures. A generic version of *VNS* is given in Algorithm 1, where $N_1, N_2, \ldots, N_r$ denote a finite set of neighborhoods, and $N_i(s)$ is the set of solutions in the i^{th} neighborhood of solution s. In *VNS* proposed for (P), the way to go from one neighborhood to the next differs from the generic version. The neighborhood $N_i(s)$ consists in dropping $i\%$ of the jobs in $\sigma(s)$, which then enter in $\Omega(s)$. Removing jobs from the accepted sequence deletes a part of the solution structure while keeping other information unchanged. In other words, move DROP is used to diversify the search. The proportion i of deleted jobs is controlled to guide the search away from the incumbent solution when no improvement has been made for a long period. More precisely, i is initialized to i_{min} and takes values in $[i_{min}, i_{max}]$ (where i_{min} and i_{max} are parameters respectively tuned to 5 and 50). In step (3), if s'' is worse than s, then i is set to $1.1 \cdot i$ (but if i is larger than i_{max}, it is set to i_{max}), whereas if s'' is better

than s, then i is set to i_{min}. In this way, the fraction of deleted jobs grows exponentially with the number of iterations without improvement.
In step (1) of our *VNS*, the selected solution is the best among a sample of k (parameter tuned to 50) solutions generated at random in $N_i(s)$. Preliminary tests showed that using such shaking strategies allows to obtain better results than choosing a single solution at random in $N_i(s)$, or dropping the jobs of $\sigma(s)$ one by one by removing the job which leads to the minimum costs at each step.
In step (2), the previously defined *TS* is used without move DROP, and is run during I (parameter tuned to 500) iterations. The initial solution of *VNS* is generated by *GR*.

Algorithm 1 Variable Neighborhood Search

Generate an initial solution s and set $i := 1$

While no stopping criterion is met, **do**

1. *Shaking:* generate a solution s' in $N_i(s)$.
2. *Local search:* apply some local search method with s' as initial solution, and let s'' be the resulting solution.
3. *Move or not:* if s'' is better than s, move there (i.e. set $s := s''$), and continue the search with N_1 (i.e. set $i := 1$); otherwise set $i := i + 1$, but if $i > r$, set $i := r$.

5 Experiments

To generate instances for (P), two critical values are used: the number n of jobs, and a parameter α which controls the interval of time in which release dates and due dates are generated. More precisely, a value *Start* is chosen large enough, and *End* is equal to $Start + \alpha \sum_j p_j$. Then, r_j is randomly chosen in interval $[Start, End]$, and d_j in $[r_j + p_j, End]$. To highlight the ease of adaptation of the proposed methods to different penalty functions, we consider both linear and quadratic penalties in each instance of the experiments. More precisely, the earliness (resp. tardiness) penalties are equal to $w_j(r_j - S_j)^{q_j}$ (resp. $w'_j(C_j - d_j)^{q'_j}$). The weights w_j and w'_j are randomly chosen in $\{1, 2, 3, 4, 5\}$, whereas q_j and q'_j are chosen in $\{1, 2\}$. $\bar{d}_j$ and $\bar{r}_j$ are chosen such that $T_j(\bar{d}_j) = E_j(\bar{r}_j) = u_j$. Value p_j is an integer randomly chosen in interval $[50, 100]$, and $u_j = \beta \cdot p_j$, where β is an integer randomly picked from interval $[50, 200]$. A number of job families is chosen randomly between 10 and 20. Setup times and costs are likely to be related in realistic situations, therefore the setup time $s_{FF'}$ between jobs of families F and F' is chosen randomly in $[50, 200]$, and the setup costs $c_{FF'}$ are set to $\lfloor \gamma \cdot s_{FF'} \rfloor$, where γ is randomly chosen in interval $[0.5, 2]$. Note that the uniform distribution has been used to generate all the data.
To compute the cost of a solution in *TS* and *VNS*, we have adapted the timing procedure proposed in Hendel and Sourd (2007), which is to our knowledge the fastest method for the above defined penalty functions. The solution methods (namely, *GR*, *TS*, and *VNS*) were implemented in C++ and performed on a computer with a processor Intel Quad-core i7 2.93 GHz with 8 GB of DDR3 RAM memory, and a time limit of $n/2$ minutes. Note that *GR* is restarted as long as the time limit is not reached, and provides the best generated solution. Table 1 presents the results. Column *Best* (in \$) indicates the best known result for each instance. Then, for each method the percentage gap between the average result over 10 runs and *Best* is given. The best result for each instance appears in bold face.
It is clear from the results that both local search methods outperform *GR*. The average gap obtained by *GR* is 55.67%, versus 16.78% and 8.88% for *TS* and *VNS*, respectively. The results also show the superiority of *VNS* over *TS*: *VNS* obtains the best results for 11

instances out of 15, versus 5 for *TS*. Obviously, the structured and controlled use of move DROP in *VNS* is efficient to tackle (P).

n	α	*Best* [$]	*GR* [%]	*TS* [%]	*VNS* [%]
25	0.5	46,860	0.40	0.13	**0.05**
	1	35,866	6.50	**0.00**	**0.00**
	2	8,172	21.25	**0.75**	1.33
50	0.5	137,567	6.47	4.26	**2.32**
	1	69,671	44.34	**10.15**	11.26
	2	6,123	166.39	30.91	**19.52**
75	0.5	198,633	19.68	6.52	**6.06**
	1	126,052	33.93	5.15	**0.60**
	2	11,199	246.30	41.58	**32.86**
100	0.5	332,731	21.32	8.36	**6.63**
	1	175,237	50.36	25.65	**4.60**
	2	20,459	124.39	39.34	**17.98**
150	0.5	561,422	23.49	**3.92**	4.80
	1	320,225	53.85	**11.76**	15.60
	2	66,585	16.34	63.20	**9.59**
Average			55.67	16.78	**8.88**

Table 1. Comparison of *GR*, *TS*, and *VNS*

6 Conclusion

We propose in this paper a simple but efficient *VNS* method for a realistic scheduling problem. Experiments show that a well conducted use of move DROP as an exploration tool is a powerful ingredient. A relevant avenue of research would consist in adapting *VNS* for order acceptance and scheduling problems in different scheduling environments (e.g., parallel machines, job shops).

References

Atan, M.O., Selim Akturk, M. , 2008, "Single CNC machine scheduling with controllable processing times and multiple due dates", *International Journal of Production Research*, Vol. 46, pp. 6087-6111.

Gendreau, M. and Potvin, J.-Y., 2010, "Handbook of Metaheuristics", *New York : Springer.*

Hendel Y., Sourd F., 2007, "An improved earliness-tardiness timing algorithm", *Computers & Operations Research*, Vol. 34, pp. 2931-2938.

Liao C.-J. , Cheng C.-C., 2007, "A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date", *Computers & Industrial Engineering*, Vol. 52, pp. 404-413.

Mansouri S. A., Gallear D., Askariazad M. H., 2012, "Decision support for build-to-order supply chain management through multiobjective optimization", *International Journal of Production Economics*, Vol. 135, pp. 24-36.

Oğuz C., Salman S.F., Yalçın B.Z., 2010, "Order acceptance and scheduling decisions in make-to-order systems", *International Journal of Production Economics*, Vol. 125, pp. 200-211.

Pinedo M., 2008, "Scheduling: Theory, Algorithms, and Systems Third Edition", *New York: Springer.*

Shabtay, D., Gaspar, N. and Yedidsion, L., 2013, "A bicriteria approach to scheduling a single machine with job rejection and positional penalties", *Journal of Combinatorial Optimization*, Vol. 23 pp. 395-424.

Thevenin S., Zufferey N. and Widmer M., 2013, "Tabu search for a single machine scheduling problem with discretely controllable release dates", *The 12th International Symposium on Operations Research in Slovenia (SOR'13)*, pp. 1590-1595.

Zufferey N., 2012, "Metaheuristics: Some principles for an efficient design", *Computer Technology and Application*, Vol. 3, pp. 446-462.

A Genetic Algorithm for the Resource-Constrained Project Scheduling Problem with Flexible Resource Profiles

Martin Tritschler, Anulark Naber, and Rainer Kolisch

TUM School of Management, Technische Universität München, Germany
martin.tritschler@tum.de, anulark.naber@tum.de, rainer.kolisch@tum.de

Keywords: resource-constrained project scheduling problem, flexible resource profiles, principal and dependent resources, resource allocation, genetic algorithm.

1 Introduction and Problem Statement

In this work we consider the resource-constrained project scheduling problem (RCPSP) with flexible resource profiles (FRCPSP), in which a set of non-preemptive activities has to be scheduled subject to finish-start precedence relations and resource availability limitations. For each activity only the total required amount of each resource is known. Activity durations, however, are not given and the resource allocation during the processing time of an activity must be determined. As the allocation of resources can be adjusted between time periods, the resulting "resource profile" (Naber and Kolisch 2013) of an activity is flexible and not limited to a rectangular shape over the entire processing time of the activity. The FRCPSP, therefore, consists of determining for each activity the start time, the duration, and the resource allocation per time period in order to minimize the project makespan.

Based on the discrete-time FRCPSP formulation of Naber and Kolisch (2013), the flexible resource profile of an activity for a specific resource must fulfill at least the activity's resource requirement, adhere to lower and upper bounds of resource usage, and maintain a "minimum block length" (Fündeling and Trautmann 2010) of time periods in which resource usage is not changed. Resources are assumed continuous and renewable with time-variant resource capacities and can belong to three categories: principal, dependent and independent. The principal resource of an activity is the main resource, whose allocated amount during the processing time of the activity defines the required amounts of dependent resources through non-decreasing linear functions. The allocated amount of an independent resource is not affected by the quantities of other resources. A project can contain multiple resources of all three kinds, but each activity has just one single principal resource.

The motivation of our research on this NP-hard problem (Kolisch et al. 2003) is to develop a genetic algorithm for the FRCPSP that can solve problem instances of practical size close to optimality and to gain insights on the problem structure in order to further improve heuristic methods.

2 Literature Review

The FRCPSP was initially studied by Kolisch et al. (2003), who defined the problem in the context of a real-world application in pharmaceutical research projects. Whereas the underlying RCPSP has been extensively covered, literature that studies the FRCPSP is relatively scarce.

Nearly all existing approaches for the FRCPSP consider discrete resources and are limited to a single principal resource only. Kuhlmann (2003) applied genetic algorithms to a

problem containing several specific requirements from practice. Ranjbar and Kianfar (2010) employed a genetic algorithm on a generated set of feasible resource profiles, which is, however, limited to specific shape types. Priority rule heuristics were proposed by Kolisch et al. (2003), Fündeling (2006), Fündeling and Trautmann (2010), and Zhang and Sun (2011). Most recently, Naber and Kolisch (2013) provided exact mixed integer programming (MIP) methods for the FRCPSP with continuous resources and multiple resource categories.

Related to the FRCPSP is the discrete time-resource tradeoff problem (DTRTP). In the DTRTP, activity processing time is a function of resource usage, which is assumed constant and integer for the duration of an activity. De Reyck et al. (1998) and Ranjbar et al. (2009) applied metaheuristics to solve the problem by selecting modes, which relate to rectangular-shape resource profiles, from a set of feasible modes. For related RCPSP formulations in which activities can be processed at different continuous processing rates, exact solution methods were proposed by, e.g., Kis (2006).

Our work founds upon the problem formulation of Naber and Kolisch (2013) for which we propose a solution approach inspired by the work of Hartmann (2002).

3 Proposed Solution Approach

As continuous resources lead to an unlimited number of potential resource profiles, the FRCPSP cannot be solved through enumeration of resource profiles. Therefore, our solution approach consists of a genetic algorithm (GA) with self-adaptive parameter control (Hartmann 2002), which determines the sequence of activities, and a schedule generation scheme (SGS), which schedules activity start times and employs heuristic resource allocation methods.

3.1 Genetic Algorithm

Our proposed GA works on an activity list representation (Kolisch and Hartmann 1999) of the FRCPSP. An activity list is any precedence feasible permutation $\lambda = (a_1, ..., a_n)$ of a set of n activities a_i with $i \in \{1, ..., n\}$. The list position $p(a_i) \in \{1, ..., n\}$ of an activity defines its priority for scheduling. In order to transform an activity list into a schedule we have to apply a SGS. Kolisch and Hartmann (1999) distinguish the serial and the parallel SGS, which can be combined with different resource allocation methods in the FRCPSP. As proposed by Hartmann (2002), the parameter which SGS type to use is encoded into solution candidates and undergoes evolution itself. As a result, the GA can self-adapt the SGS type to the respective problem instance. Thus, a solution candidate is represented by a tuple (λ, s), where s denotes the selected SGS type.

Following the approach of Hartmann (2002), we employ a priority rule based biased sampling heuristic to generate activity lists for the initial population of candidates. We incorporate diversity into the initial population by using different priority rules in the sampling process, including the longest path following (LPF) rule, which, according to Fündeling and Trautmann (2010), provided good results for the discrete FRCPSP. Furthermore, we randomly assign SGS types s. To create a new generation of solution candidates, we modify λ and s with the crossover and mutation operators of Hartmann (2002). Candidates for the next generation are selected based on their fitness, which relates to the makespan of a candidate's resulting schedule, as generated by the SGS.

3.2 Schedule Generation Scheme and Resource Allocation Method

Based on the sequence of activities and the selected SGS type, we can schedule activities and allocate resources. Whereas the set of schedules generated by a serial SGS contains

optimal solutions in the RCPSP, this is not necessarily the case in the FRCPSP (Fündeling and Trautmann 2010). A study by Fündeling (2006) showed no clear dominance of the serial or the parallel SGS on all considered problem instances. Hence, we apply both SGS types in order to generate a broad variety of schedules and to not exclude promising solutions.

The serial SGS as proposed by Naber and Kolisch (2013) employs a greedy resource allocation method. The algorithm iterates through activities in λ in the sequence defined by the GA. An unscheduled activity a_i is eligible in a time period if the activity can be started in this period based on precedence relations. In each iteration the algorithm schedules a single eligible activity a_i as early as possible and allocates resources as much as possible. Iterating over time periods, the algorithm allocates the maximum possible resource amount to a_i until the resource requirements of a_i are fulfilled and the activity finishes. If resource constraints are violated, the algorithm returns to the period in which the current block starts and reallocates resources. If resources do not suffice to fulfill the minimal resource usage bounds, the starting period of a_i is revised. The algorithm terminates when all activities have been completed.

We adapted the parallel SGS of Fündeling (2006) for multiple continuous resource types. The parallel SGS increments time periods and considers in each iteration a decision period t. First, it allocates resources to active (i.e., ongoing) activities to guarantee that minimum block lengths and minimal resource usage bounds are met. The remaining resources are then distributed among active activities which qualify for a change in resource allocation and, if sufficient resource capacities remain, eligible activities. To determine the resource amounts allocated to each activity, the sequence of activities in λ, as defined by the GA, and the minimal resource usage bounds are considered. If resource constraints are violated, the algorithm returns to the starting period of the current block and reallocates resources. If resources do not suffice to fulfill the minimal resource usage bounds, an activity is unscheduled and reconsidered for scheduling in later periods. The algorithm terminates when all activities have been completed.

4 Computational Study and Further Research

In our computational study we will evaluate the performance of the proposed method on a dataset of Fündeling and Trautmann (2010), which contains instances of up to 200 activities. We will also compare the quality of our metaheuristic solutions to those of the exact MIP model FP-DT3 of Naber and Kolisch (2013) on instances with up to 40 activities from the dataset of Fündeling and Trautmann (2010).

For further research, local search techniques shall be explored to improve solution quality. Other attempts may include the incorporation of linear programming models to more efficiently allocate continuous resources.

References

De Reyck, B., Demeulemeester, E. and Herroelen, W. (1998), 'Local search methods for the discrete time/resource trade-off problem in project networks', *Naval Research Logistics* **45**(6), 553–578.

Fündeling, C.-U. (2006), *Ressourcenbeschränkte Projektplanung bei vorgegebenen Arbeitsvolumina*, Gabler Edition Wissenschaft Produktion und Logistik, Dt. Univ.-Verl, Wiesbaden.

Fündeling, C.-U. and Trautmann, N. (2010), 'A priority-rule method for project scheduling with work-content constraints', *European Journal of Operational Research* **203**(3), 568–574.

Hartmann, S. (2002), 'A self-adapting genetic algorithm for project scheduling under resource constraints', *Naval Research Logistics* **49**(5), 433–448.

Kis, T. (2006), RCPS with Variable Intensity Activities and Feeding Precedence Constraints, *in* J. Józefowska and J. Weglarz, eds, 'Perspectives in Modern Project Scheduling', Vol. 92 of

International Series in Operations Research & Management Science, Springer US, pp. 105–129.

Kolisch, R. and Hartmann, S. (1999), Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis, *in* J. Węglarz, ed., 'Project Scheduling', Vol. 14 of *International Series in Operations Research & Management Science*, Springer US, pp. 147–178.

Kolisch, R., Meyer, K., Mohr, R., Schwindt, C. and Urmann, M. (2003), 'Ablaufplanung fur die Leitstrukturoptimierung in der Pharmaforschung', *Zeitschrift für Betriebswirtschaft* **73**(8), 825–848.

Kuhlmann, A. (2003), *Entwicklung eines praxisnahen Project-scheduling-Ansatzes auf der Basis von genetischen Algorithmen*, Logos-Verl., Berlin.

Naber, A. and Kolisch, R. (2013), MIP models for Ressource-Constrained Project Scheduling with Flexible Resource Profiles, Working paper, TUM School of Management, Technische Universität München, Germany.

Ranjbar, M., De Reyck, B. and Kianfar, F. (2009), 'A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling', *European Journal of Operational Research* **193**(1), 35–48.

Ranjbar, M. and Kianfar, F. (2010), 'Resource-Constrained Project Scheduling Problem with Flexible Work Profiles: A Genetic Algorithm Approach', *Scientia Iranica* **17**(1), 25–35.

Zhang, L. and Sun, R. (2011), An improvement of resource-constrained multi-project scheduling model based on priority-rule based heuristics, *in* '2011 8th International Conference on Service Systems and Service Management (ICSSSM 2011)', pp. 1–5.

A B&B Approach to Production Scheduling of Tailored Items with Stochastic Release Dates and Processing Times

Urgo M.[1]

Manufacturing and Production Systems Laboratory, Mechanical Engineering Department
Politecnico di Milano, Milano, Italy
marcello.urgo@polimi.it

Keywords: Stochastic Scheduling, Tailored Production.

1 Introduction

In hierarchical production control systems, planning decides on capacities and operations to meet the demand, while scheduling should guarantee the execution of production plans even in face of uncertainties. Aiming at reducing the complexity of the production control system, uncertainties are handled as close to the root of their sources as possible, i.e., at the level of production scheduling. The key idea is trying to stop the propagation of local disruptions to other production phases or departments, let them be in or outside of the boundaries of the enterprise. In particular, we focus on a single resource stochastic scheduling model that captures release and processing time uncertainties of a set of jobs. This framework fits the production of tailored items, where products are specifically designed for a customer and then produced in very small lots. Uncertainty affects the release dates due to the possible delays coming from the raw materials suppliers or from previous production phases, but also the processing time, since the customised design of the product quite often entails the need of fine tuning the process to obtain the desired quality. In order to avoid the propagation of local disruptions downwards in the production chain, the scheduling approach aims at minimising a risk measure of the maximum lateness.

In its deterministic version, this problem is a classical scheduling problem $1|r_i|L_{max}$ and has been demonstrated to be strongly $NP-hard$. In particular, without release dates, the resulting scheduling problem $(1|L_{max})$ is rather simple and can be solved to optimality using the *earliest due date* (EDD) rule. If we consider arbitrarily distributed processing times and deterministic due dates, the EDD rule still minimises the expected maximum lateness (Pinedo 2008) for non-preemptive static list and dynamic policies and for preemptive dynamic policies. Since the EDD schedule provides the optimal maximum lateness for any sample of the processing times, the cumulative distribution of the maximum lateness for the EDD schedule bounds from above all the cumulative distributions of the maximum lateness obtained with any possible schedule. This behaviour can be formalized in terms of stochastic order relations (Ross 1983). Rearrangement inequalities and scheduling problems have been addressed in (Chang and Yao 1993) where a closed form solution for the stochastic counterpart of many classical deterministic scheduling problems is obtained. These results have been rephrased and further exploited in (Zhou and Cai 1997, Cai *et al.* 2007, Wu and Zhou 2008).

The introduction of the release dates (either deterministic or stochastic) makes the problem more difficult to solve. However, with independent generally distributed release dates and processing times and deterministic due dates, the EDD rule still minimises L_{max} but only in the preemptive case and further extension are available but only under additional constraints on the distributions of release dates and processing times (Pinedo

2008). Referring to stochastic objective functions other than the expected value, a trade-off between mean and variance is one of the most simple and used risk measure (De *et al.* 1992). Other common objective functions in the stochastic scheduling are the flow time and the completion time. Moreover, (Sarin *et al.* 2009) provide closed form equations of mean and variance for a large set of scheduling problems. However, no algorithm, neither exact, nor heuristic has been proposed for the maximum lateness single machine scheduling problem to optimise a stochastic objective function different from the expected value.

2 Using a Risk Measure as Objective Function

The financial research has paid particular attention to the definition of risk measures to cope with uncertainty. In particular the study of extreme events, i.e., the tails of the distribution has received due attention. Risk measures as the *Value-at-Risk* are extensively used in the portfolio management and a large amount of literature has been written on their mathematical properties and effectiveness in protecting assets investments.

If we consider a vector of decision variables x and a random vector y governed by a probability measure P on Y independent on x, they univocally determine the value of a performance indicator $z = f(x, y)$ with $f(x, y)$ continuous in x and measurable in y and such as $E\left[|f(x, y)|\right] \leq \infty$. Given x and the performance indicator z, we define the associated distribution function $\psi(x, \cdot)$ on $\mathbb{R}$ as:

$$\Psi(x, \zeta) = P(y|f(x, y) \leq \zeta) \tag{1}$$

The *Value-at-Risk* α ($\alpha - VaR$) of the value of the performance indicator z associated with the decision x is (Rockafellar and Uryasev 2002):

$$\zeta_\alpha(x) = min\{\zeta|\Psi(x, \zeta) \geq \alpha\} \tag{2}$$

A different case refers to discrete distributions as in scenario-based uncertainty models where the uncertainty is modelled through finitely many points $y_k \in Y$, $z = f(x, y)$ is concentrated in finitely many points and $\psi(x, \cdot)$ is a step function. Under these hypotheses, the definition of either the *Value-at-Risk* in (2) must be rephrased (Rockafellar and Uryasev 2002). Given x, if we assume that the different possible values of $z_k = f(x, y_k)$ with $P(z = z_k) = p_k$ can be ordered as $z_i < z_2 < \cdots < z_N$ and given k_α such that

$$\sum_{k=1}^{k_\alpha} p_k \geq \alpha \geq \sum_{k=1}^{k_{\alpha-1}} p_k \tag{3}$$

then the $\alpha - VaR$ is given by $\zeta_\alpha(x) = z_{k_\alpha}$

Being the *Value-at-Risk* a quantile of the objective function distribution, the stochastic dominance between two *cumulative distribution functions (cdf)* also implies a dominance between the respective *Value-at-Risk*, for any given α.

3 A Branch and Bound Approach

We consider a single machine scheduling problem where a set of jobs A containing n jobs that must be processed on a machine M. Let p_j be the processing time of job $j \in A$ and s_j its starting time. Each job is subject to a release date r_j and a due date d_j. The aim is at finding an optimal schedule minimising the $\alpha - VaR$ of the maximum lateness. We restrict the problem to static non-preemptive list policies with unforced idleness allowed. Both the release dates r_j and the processing times p_j of the jobs are independent stochastic variables with general discrete distributions. The objective function is a stochastic variable

itself whose distribution depends on the values of the stochastic variables p_j and r_j and on a set of decisions defining how the jobs are scheduled.

The adopted branching scheme starts from a root node (level 0) where no job has been scheduled. Starting from this node, it is possible to sequence each of the jobs to be scheduled, hence, there are n branches departing from the root node and going down to n nodes (level 1) having different jobs executed first in the schedule. In general, at each node at level $k-1$ in the branching tree, the first $k-1$ jobs in the schedule are already sequenced and $n-k+1$ branches lead to a new node at level k with a different jobs scheduled next. Hence, at level k there are $n!/(n-k)!$ nodes (Pinedo 2008).

Let us consider two jobs $i, j \in A$, with stochastic processing times p_i and p_j described by their cumulative distribution functions $F_i(t)$ and $F_j(t)$. The two jobs are executed in series, first i then j and no release date is considered. Being $*$ the convolution operator, the cumulative distribution functions of the completion times of jobs i and j (c_i and c_j) can be calculated as:

$$F_{c_i}(t) = F_i(t) \ , F_{c_j} = F_{c_i}(t) * F_j(t) = F_{i+j}(t) = F_i(t) * F_j(t)$$

If we consider a stochastic release date for job j, it can be modelled as an additional job k with processing time r_j to be executed before j. Moreover, given the cdf of the completion time of j and its due date d_j, the cdf of the lateness L_j can be calculated as:

$$F_{L_j}(t) = F_{c_j}(t - d_j) \tag{4}$$

Provided the cdf of the lateness for all the considered jobs, the cdf of the maximum lateness is:

$$F_{L_{max}}(t) = \prod_{j \in A} F_{T_j}(t) \tag{5}$$

At each node in the tree, a subset of the jobs ($A^S \in A$) is already scheduled. For these jobs the maximum lateness cdf can be calculated using Equation 5. The execution of the remaining jobs ($A \backslash A^S \in A$) has not been sequenced yet and, hence, the cdf of the maximum lateness of the complete schedule cannot be univocally calculated. Given a not yet scheduled jobs in $j \in A \backslash A^S$, a lower bound for its lateness can be obtained assuming it starts immediately after the already scheduled jobs (A^S) or, if more constraining, after its release date r_j. An upper bound for the lateness L_j of a not jet scheduled job $j \in A \backslash A^S$ can be obtained assuming that it will be sequenced as the last job in the schedule in the following scheme. We compute the maximum cdf between the completion time of the currently scheduled jobs and the release dates of the jobs in $A \backslash A^S$. Then, we calculate the convolution between this cdc and all the cdfs $F_i(t)$ with $i \in A \backslash A^S$. Hence, we compute the maximum cdf between the one just obtained and the release date of job j, convolute it with $F_j(t)$ and consider its value in $t - d_j$ to have the L_j.

4 Results

In total, 160 instances of 10 activities have been generated and solved considering two different risk levels (5% and 25%) for 320 experiments. The algorithm has been coded in C++ using the BoB++ library (Djerrah *et al.* 2006) and executed two Intel X5450 processors. The results in Table 1 show the performance of the algorithm in terms of the time to find the optimal solution and the fraction of nodes of the branching tree visited.

The results show that the algorithm was able to find the optimal solution in an average time of 6.7 seconds, with a variability ranging from a minimum value of 0.547 seconds to a maximum value of 114.781 seconds. Moreover, the average number of nodes visited during

Table 1. Solution time (in seconds).

			Min.	Max.	Mean.	St. dev.
VaR	Risk=5%	Solution time (seconds)	0.810	114.780	7.350	14.030
		% visited nodes	0.017	2.438	0.126	0.266
		% difference vs EDD	0	228.570	6.43	24.37
	Risk=25%	Solution time (seconds)	0.547	97.625	6.042	9.156
		% visited nodes	0.012	1.785	0.111	0.190
		% difference vs EDD	0	98.50	5.09	15.52

the search is about 0.12% of the total number of nodes in the branching tree (6235300). In addition, the results seem to show a slightly increase of the solution time when dealing with a risk level of 5%. This is reasonable since, the more the considered quantile resides in the tail of the distribution, the more the value for different schedules are packed together in a strict range and the effectiveness of the bounding and pruning rules is decreased.

A second class of results reported in Table 1 are the comparison of the optimal solution obtained with the approach against the schedule obtained using the *Earliest Due Date (EDD)* rule. To compare the two solutions, first the EDD rule is used to obtain a schedule. Hence, the schedule is evaluated with the exact approach to calculate the real VaR associated. This value is then compared with the VaR of the optimal schedule. The results shows that the proposed approach perform on average between 5.09% and 6.43% better respect to a simple rule as the EDD. Performing better means that the solution provided by the EDD rule has a VaR different from that associated to the considered risk level.

Acknowledgements

This research has been partially funded by the EU FP7 Project VISIONAIR - Vision and Advanced Infrastructure for Research, Grant no. 262044.

References

Cai, X., Wang, L., Zhou, X., 2007. Single-machine scheduling to stochastically minimize maximum lateness, *Journal of Scheduling*, 10 (4), pp. 293–301.

Chang, C.-S., Yao, D. D., 1993. Rearrangement, majorization and stochastic scheduling, *Mathematics of Operations Research*, 18 (3), pp. 658–684.

De, P., Ghosh, J. B., Wells, C. E., 1992, Expectation-variance analysis of job sequences under processing time uncertainty, *Int. Journal of Production Economics*, 28 (3), pp. 289 – 297.

Djerrah, A., Le Cun, B., Cung, V.-D., Roucairol, C., 2006, Bob++: Framework for solving optimization problems with branch-and-bound methods. In: *15th IEEE International Symposium on High Performance Distributed Computing*, pp. 369 –370.

Pinedo, M. L., 2008, Scheduling - Theory, Algorithms, and Systems. Springer New York.

Rockafellar, R. T., Uryasev, S., 2002, Conditional value-at-risk for general loss distributions, *Journal of Banking & Finance*, 26 (7), pp.1443 – 1471.

Ross, S. M., 1983. Stochastic Processes, 2nd Edition. Holden Day.

Sarin, S., Nagarajan, B., Jain, S., Liao, L., 2009, Analytic evaluation of the expectation and variance of different performance measures of a schedule on a single machine under processing time variability, *Journal of Combinatorial Optimization*, 17 (4), pp. 400–416.

Wu, X., Zhou, X., 2008. Stochastic scheduling to minimize expected maximum lateness, *European Journal of Operational Research*, 190 (1), pp. 103 – 115.

Zhou, X., Cai, X., 1997. General stochastic single-machine scheduling with regular cost functions, *Mathematical and Computer Modelling*, 26 (3), pp. 95 – 108.

A new hard benchmark for the permutation flowshop scheduling problem

Eva Vallada[1], Rubén Ruiz[1] and Jose M. Framinan[2]

[1] Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, 46021, València, Spain.
evallada,rruiz@eio.upv.es
[2] University of Seville, Spain
framinan@us.es

Keywords: benchmark, flowshop, makespan.

1 Introduction

A new benchmark of hard instances for the permutation flowshop scheduling problem with the objective of minimising the makespan is proposed. The new benchmark consists of 240 large instances and 240 small instances with up to 800 jobs and 60 machines. One of the objectives of the work is to generate a benchmark which satisfies the desired characteristics of any benchmark: comprehensive, amenable for statistical analysis and discriminant when several algorithms are compared. An exhaustive experimental procedure is carried out in order to select the hard instances, generating thousands of instances and selecting the hardest ones from the point of view of a GAP computed as the difference between very good upper and lower bounds for each instance. The permutation flowshop scheduling problem (PFSP) consists of determining a processing sequence of n jobs in a set of m machines that are disposed in series. All jobs must be processed sequentially in all machines. Each job j, $j = \{1, \ldots, n\}$ needs a processing time of p_{ij} units at each machine i, $i = \{1, \ldots, m\}$. This processing time is a non-negative, deterministic and known amount. The completion time of a job in the factory is denoted as C_j. The most common objective for the PFSP is the minimisation of the maximum C_j. This is referred to as makespan and denoted as $C_{\max}$. The most widely used benchmark for flowshop scheduling is that of (Taillard 1993) which comprises 120 instances that range from 20 jobs and 5 machines to 500 jobs and 20 machines. At the time of this writing, only 28 instances in the benchmark are "open" meaning that the optimum solution is not known. Several authors have recently been unable to find statistically better performance using Taillard's benchmark and showed that using other randomly generated instances of their own, better performance was observed. In a sense, Taillard's benchmark is reaching exhaustion.

2 New benchmark

The new benchmark consists of 240 large instances and 240 small instances. Small instances are a set of 240 with the following combinations of number of jobs (n) and number of machines (m): $n = \{10, \ldots, 60\}$, $m = \{5, \ldots, 20\}$. For each combination 10 instances are generated, so in total we have $6 \times 4 \times 10 = 240$ small instances. Note that small instances are up to 60 jobs and 20 machines, so we can consider this set to be actually small-medium sized. If we compare with Taillard's, the smallest size is 20 jobs and 5 machines, after 20 jobs, the next size is 50 jobs, so there is an important GAP. Regarding the number of machines, there are also $GAPS$, from 10 to 20 machines in some of the instances. Moreover, Taillard's instances are not equidistant. For example, from

20 jobs and 5 machines to 20 jobs and 10 machines and 20 jobs and 20 machines. The difference between the two first instances is 5 machines and from the second to the third is 10 machines. All these differences make the statistical analysis of the results difficult. In the same way, our benchmarks allow the orthogonal analysis in design of experiments, i.e., all combinations of n and m are present. That way, two-factor interactions between the number of jobs and machines can also be studied. This is not possible with Taillard's, as some $n \times m$ combinations are missing, like 200×5, 500×5 and 500×10. Regarding the large instances, they are also a set of 240 where $n = \{100, \ldots, 800\}$ and $m = \{20, 40, 60\}$. For each combination 10 instances are generated, in total $8 \times 3 \times 10 = 240$ large instances. In this way, two of the three desired characteristics are satisfied: they are exhaustive and amenable for statistical analysis. Regarding the generation, an exhaustive and detailed experimental procedure is carried out. The process is the same for both small and large instances, and consists of generating thousands of instances and to select the hardest ones. Specifically, for small instances, 2,000 instances are generated for each combination. From these 2,000 instances per combination, the hardest 10 are chosen to be part of the new benchmark. For large instances, the procedure is the same, but a 1,000 instances are generated for each combination instead of 2,000. Therefore, a total of 48,000 small instances and 24,000 large instances are generated. From these ones, 240 small and 240 large will be chosen to be part of the new benchmark, those that result as the hardest to solve. The difference between two instances of the same size ($n \times m$) is the matrix of processing times. To test how difficult it is to solve an instance we have on the one hand, two effective algorithms for the permutation flowshop scheduling problem with the objective to minimise the makespan (Ruiz *et al.* 2006) and (Ruiz and Stützle 2007). On the other hand, four lower bounds, one from (Taillard 1993) and the three best polynomial bounds of (Ladhari and Haouari 2005), are computed for each instance. All the generated instances (48,000 small and 24,000 large) are solved by the two effective algorithms, denoted as HGA (Ruiz *et al.* 2006) and IG (Ruiz and Stützle 2007), since they are considered the most effective for this problem. Both algorithms are metaheuristics so each one is run three times on each instance. Regarding the stopping criterion for the methods, a maximum elapsed CPU time is set to $n \cdot (m/2) \cdot 120$ milliseconds (small instances) and $n \cdot (m/2) \cdot 90$ milliseconds (large instances) in order to obtain good upper bounds. These values, 120 and 90, were chosen after checking the algorithms converged. Therefore, for each instance we have six makespan values (three for each run of both algorithms) and four lower bounds. We obtain, for each instance, an upper bound (UB) from the minimum makespan among the six makespan values, and a lower bound (LB) from the maximum value among the four computed lower bounds. The objective is to obtain the GAP between the upper bound and the lower bound for each instance, following the expression $GAP = (UB - LB)/LB$. The higher the GAP value, the harder the instance is. If the upper bound is very close or equal to the lower bound, a GAP near zero will be obtained. In order to obtain the hardest instances per combination, GAP values for the 2,000 instances (small case) or 1,000 instances (large case) are sorted from highest to lowest. The 10 first instances per combination are selected to be part of the new benchmark. The same procedure is applied for both, small and large instances, and as a result of the experimental process, the new benchmark with 240 small instances and 240 large instances is generated.

3 Computational evaluation

Once the new benchmark is generated (denoted as VRF), the objective is to empirically test if it is harder to solve than the most used benchmark for this problem (Taillard's). That is, to satisfy the third desired characteristic: to obtain a discriminant benchmark. In order

to check how hard the new proposed benchmark is, several experiments were carried out based on the comparison of heuristics and metaheuristics methods using both benchmarks (Taillard and VRF). Some of the most important results are remarked in this work (due to space restrictions it is not possible to detail the computational evaluation which in total needed almost 6 years of CPU time). To measure the effectiveness of the methods, the Average Relative Percentage Deviation (RPD) is computed for each instance according to the following expression $RPD = (Method_{sol} - Best_{sol})/Best_{sol}$ where $Best_{sol}$ is the best known solution and $Meth_{sol}$ is the solution obtained by the metaheuristic method.

The NEH heuristic by (Nawaz *et al.* 1983) and NEHD heuristic, which is a modification and improvement of NEH, proposed by (Dong and Ping Chen 2008) were selected since they are considered the most effective ones for the PFSP. In Table 1 we can see the average RPD for both methods and benchmarks. We can observe that, on average, results are very similar for both benchmarks. Heuristic NEHD seems to be more effective than NEH. In order to check if these differences in the RPD values are statistically significant, we apply an analysis of variance (ANOVA). If we focus our attention on the tables, differences, on average, seem to be very similar for both benchmarks. However, the statistical analysis states that these differences are not statistically significant for Taillard's benchmark (intervals are overlapped). Results are very different for VRF (large instances), we can clearly see that there are statistically significant differences between the average RPD values (Figure 1). The result with our independent coding and testing of the NEHD matches that of the original authors as they used their own benchmark since there were no differences using Taillard's. So the conclusion is that we have been able to obtain statistically significant differences with our new benchmark which Taillard's benchmark could not.

Table 1. RPD for the heuristic methods (Taillard and VRF benchmarks)

Heuristic	Taillard	VRF
NEH	3.37	3.35
NEHD	2.91	2.88

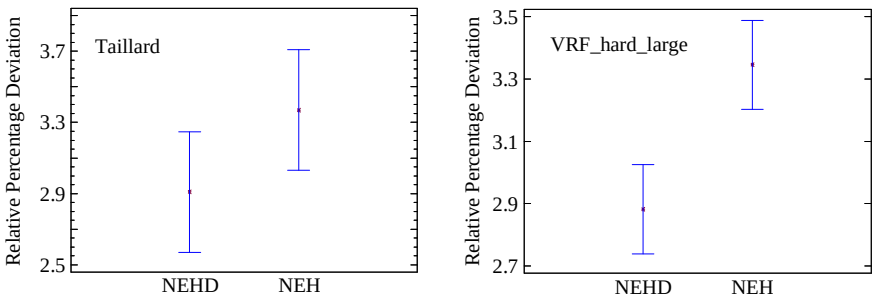


Fig. 1. Means Plot and Tukey HSD intervals at 99% confidence level for the heuristic methods (Taillard's instances left side, VRF (large instances) right side).

For the metaheuristics, the previous HGA and IG methods are selected in order to test the proposed benchmark and to obtain a comparison with Taillard's. Metaheuristic methods are run until a stopping criterion is met: maximum CPU time is set to $n \cdot (m/2) \cdot 60$ milliseconds. The effectiveness of the methods is measured with the RPD but in this case five replicates of each algorithm are run as both methods are stochastic. Results can be

seen in Table 2. The statistical analysis is shown in Figure 2, where we can observe that the differences are much larger than those obtained for Taillard's and they are also statistically significant. We see that with both benchmarks we otain statistically significant differences but with our proposed benchmark, the differences are much larger.

Table 2. RPD for the metaheuristic methods (Taillard and VRF benchmarks)

Method	Taillard	VRF
HGA	0.36	0.94
IG	0.25	0.72

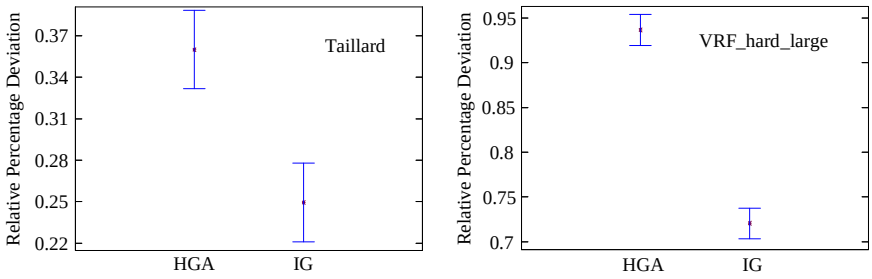


Fig. 2. Means Plot and Tukey HSD intervals at the 99% confidence level for the metaheuristic methods (Taillard's instances left side, VRF (large instances) right side).

Acknowledgements

Eva Vallada and Rubén Ruiz are partially supported by the Spanish Ministry of Economy and Competitiveness, under the project "RESULT - Realistic Extended Scheduling Using Light Techniques" with reference DPI2012-36243-C02-01 co-financed by the European Union and FEDER funds and by the Universitat Politècnica de València, for the project MRPIV with reference PAID/2012/202.

References

Dong, X., Ping Chen, H.H. 2008, "An improved NEH-based heuristic for the permutation flowshop problem", *Computers and Operations Research*, Vol. 35, pp. 3962-3968.

Ladhari, T., Haouari, M., 2005, "A computational study of the permutation flow shop problem based on a tight lower bound", *Computers and Operations Research*, Vol. 32, pp. 1831-1847.

Nawaz, M., Enscore, Jr, E. E., Ham, I., 1983, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem", *OMEGA, The International Journal of Management Science*, Vol. 11, pp. 91-95.

Ruiz, R., Maroto, C., Alcaraz, J., 2006, "Two new robust genetic algorithms for the flowshop scheduling problem", *OMEGA, The International Journal of Management Science*, Vol. 34, pp. 461-476.

Ruiz, R., Stützle, T., 2007, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem", *European Journal of Operational Research*, Vol. 177, pp. 2033-2049.

Taillard, E., 1993, "Benchmarks for basic scheduling problems", *European Journal of Operational Research*, Vol. 64, pp. 278-285.

A Metaheuristic Solution Approach for the Time-constrained Project Scheduling Problem

Van Peteghem Vincent[1], Verbeeck Cédric[2]

[1] EDHEC Business School, France
vincent.vanpeteghem@edhec.edu
[2] Ghent University, Belgium
cedric.verbeeck@ugent.be

Keywords: project scheduling, metaheuristics, time-constrained.

1 Introduction

The aim of project scheduling is the allocation of time intervals to the processing of activities, which can be executed by using a set of scarce resources. In the classical resource-constrained project scheduling problem (RCPSP), this set of available resources is limited and it is not allowed to exceed the available resources. The main focus lies in the minimization of the total duration of the project subject to precedence relations between the activities and the limited renewable resource availabilities. Various exact and (meta-)heuristic procedures for the RCPSP are already proposed in the literature. For an overview of the literature on the RCPSP, see, e.g., Hartmann and Kolisch (2000), Kolisch and Hartmann (2006) and Hartmann and Briskorn (2010).

Another problem in project scheduling, the resource availability cost problem (RACP), focuses on the minimization of the resource cost. In contrast to the 'problem of scarce resources' (Möhring 1984), the resources are not constrained by limited capacities, but a predefined deadline is imposed on the project duration. Mohring (1984) refers to this problem as the 'problem of scarce time'. The aim is to reduce the cost which is associated to the use of resources. The objective is to schedule the activities such that all precedence constraints are observed, the deadline for project termination is met, and the total resource availability cost is minimized.

The Time-Constrained Project Scheduling Problem is a variant on the RCPSP and RACP. Although a fixed set of resources is available, additional resources can be temporarily allocated in certain periods to meet the given deadline. The problem consists of determining a schedule such that the project is completed on time and the total additional cost for the resources is minimized. Since the TCPSP has a nonregular objective function, the existing solution techniques of the RCPSP and the RACP are not suitable for the TCPSP.

Although the problem is of high practical relevance, it has been considered in the literature only rarely. Deckro and Herbert (1989) proposed a mixed-integer formulation for the problem and introduced the concept of project crashing. Kolisch (1995) presented a heuristic for the TCPSP with limited hiring. Furthermore, the priority-rule based heuristics initially proposed for resource levelling by Neumann and Zimmermann (1999) may also be applied for the TCPSP, even considering minimum and maximum time-lags. The same is true for the branch and bound procedure developed by Zimmermann and Engelhardt (1998). Finally, Guldemond *et. al.* (2008) proposed a two stage heuristic for the TCPSP, where it is also allowed to hire additional capacity in overtime. In the first stage of the heuristic, partial schedules are constructed, while in the second stage, an ILP formulation is used to turn a partial schedule into a feasible schedule, and to perform a neighborhoud

search. In a later paper, the authors also presented an algorithm for the problem with adjacent resources (Hurinck *et. al.* 2011).

Nevertheless, the development of an efficient and effective algorithm has a large relevance in project management, since the use of deadlines often occurs in projects in practice. In commercial project management software, the parameters values of TCPSP, e.g. overtime cost, can usually be specified, however, the capabilities of these packages concerning TCPSP are rather restricted. Experimental results show that the proposed solution procedure gives state-of-the-art results for the problem under study. The remainder of the paper is organised as follows: the next section describes the general formulation of the problem, while section 3 describes shortly the solution procedure designed for the TCPSP. In section 4 some preliminary results of the computational experiments are reported.

2 Problem formulation

The TCPSP can be stated as follows. A set of activities N, numbered from a dummy start node 0 to a dummy end node $n + 1$, is to be scheduled without pre-emption on a set R of renewable resource types. Each activity $i \in N$ has a deterministic duration d_i and requires r_{ik} units of resource type $k \in R$. A set of R resources each with an availability equal to a_k, which is assumed constant throughout the project horizon, is available for each resource type k. A project network is represented in an activity-on-the-node format where A is the set of pairs of activities between which a finish-start precedence relationship with time lag 0 exists, and a dummy start node 0 and end node $n + 1$ representing the start and completion of the project. These dummy nodes have zero duration while the other activities have a non-zero duration; the dummies also have zero resource usage. We assume graph $G(N, A)$ to be acyclic.

The precedence relations are given by sets of activities P_j , denoting all direct predecessors of activity j, that have to complete before j starts. S_j denotes the set of all direct successors of activity j which are allowed to start only after activity j is finished.

The project should be finished before the pre-specified deadline δ. Since this deadline can be too tight in order to be realized with the resources initially provided, additional resources can be temporarily be allocated in certain periods resulting in additional cost depending on the respective resource type. The non-negative variable H_{kt} represent the amount of capacity hired of resource type k in time unit t. The variable c_k indicates the cost to hire one resource of resource type k for one time unit. A schedule S is defined by a vector of activity finish times f_i and is said to be feasible if all precedence constraints are satisfied and the project finishes within the predefined deadline δ. The objective of the TCPSP is to find a feasible schedule within the pre-specified project deadline δ such that the total additional cost is minimized. The time-constrained project scheduling problem can be represented as $m, 1|cpm, \delta_n|vrc$ using the classification scheme of Herroelen *et. al.* (1999) or as $PS_m, \infty|d, prec| \sum C_k \max(0, R_k - r_k(S, t)$ following the classification scheme of Brucker *et. al.* (1999).

The TCPSP can be conceptually formulated as follows:

$$\text{Min.} \sum_{k \in R} \sum_{t=0}^{\delta} H_{kt} c_k \tag{1}$$

s.t.

$$f_i + d_j \leq f_j \qquad \forall (i,j) \in A \tag{2}$$

$$\sum_{i \in S(t)} r_{ik} \leq a_k + H_{kt} \qquad \forall k \in R, t = 1, ..., \delta \tag{3}$$

$$f_{n+1} \leq \delta \tag{4}$$

$$f_0 = 0 \tag{5}$$

$$a_k \geq 0 \qquad \forall k \in R \tag{6}$$

$$H_{kt} \geq 0 \qquad \forall k \in R, t = 1, ..., \delta \tag{7}$$

The objective function (1) minimizes the total additional resource cost of the project. Constraint (2) takes the finish-start precedence relations with a time-lag of zero into account. The renewable resource constraints are satisfied thanks to constraint (3), where $S(t)$ represents the set of activities in progress during the time interval $]t-1, t]$. Constraint (4) imposes a pre-specified deadline δ to the project and constraint (5) forces the project to start at time instance zero. Constraints (6) and (7) denote the domain of the variable a_k and H_{kt}.

3 Proposed solution procedure

A solution method is proposed for the time-constrained project scheduling problem, based on the metaheuristic framework of the artificial immune system. The term artificial immune system represents a broad range of algorithms based on the biological processes of the human immune system. According to Gendreau and Potvin (2010) there are 3 categories of artificial immune systems: negative selection algorithms, immune networks and clonal selection algorithms. Only this last category is used to optimize complex combinatorial problems like the TCPSP.

The proposed AIS for TCPSP consists of 4 components: population initiation, selection, hypermutation and a local search. During the population initiation, we generate $top*childs$ initial schedules using a Greedy Random Adaptive Search Procedure (GRASP) in order to obtain a start population. In the second component, all the initial solutions are sorted according to their fitness function (from low to high). Subsequently we select the first *top* elements as our procreation basis. For every top element, *childs* amount of clones are created using a hypermutation process. To prevent the loss of good schedules, an elitism extension was added as mentioned by De Jong (1975). More specifically, the best solution (the first top element) will be added to the new population without modification. After the hypermutation procedure the new schedules are repaired and their fitness value is computed. The fourth component is a local search procedure that tries to reposition deficit activities, which are activities that are scheduled on time periods were extra resources are hired, in order to reduce the total cost of the schedule. The local search procedure is executed on every solution of the new population.

4 Computational results

The quality of the proposed solution procedure is compared to an existing solution procedure of Guldemond *et. al.* (2008). In their study, the authors present computational results for tests on the J30, J60, J90 and J120 set of the PSPLIB dataset. All costs were set equal to 1 and the deadline of each instance was set equal to the optimal (or the best known) project makespan found on the PSPLIB website (http://www.om-db.wi.tum.de/psplib/).

As a consequence, the optimal solution for each of these instances has an objective function of 0.

Table 1. Comparison algorithms

	Guldemond *et. al.* (2008)			**AIS (this paper)**		
Jobs	Average cost	Perc. optimal	Av. CPU-time	Average cost	Perc. optimal	Av. CPU-time
30	3.2	61.25	2.9	0.89	68.54	0.05
60	13.4	57.50	17.2	8.40	68.33	0.18
90	22.6	58.75	53.0	21.77	66.88	0.38
120	72.4	12.50	308.2	92.24	20.33	1.60

These results clearly illustrate that our algorithm performs better in respect to cost as well as percentage optimal solutions found for all datasets except the j120. Finally, our solution method is also drastically faster than the method proposed by Guldemond *et. al.* (2008).

References

Brucker, P., Drexl, A., Mohring, R., Neumann, K., and Pesch, E., 1999, "Resource-constrained project scheduling: notation, classification, models, and methods", *European Journal of Operational Research*, Vol. 112, pp. 3–41.

De Jong, K., 1975, "An analysis of the behavior of a class of genetic adaptive systems", *Ph.D. thesis, University of Michigan, Ann Arbor, Michigan.*

Deckro, R. F. and Herbert, J. E., 1989, "Resource constrained project crashing", *OMEGA International Journal of Management Science*, Vol. 17, pp. 69–79.

Gendreau, M. and Potvin, J.-Y., 2010, "Handbook of Metaheuristics second edition", Springer, New York Dordrecht Heidelberg London.

Guldemond, T., Hurink, J., Paulus, J., and Schutten, J., 2008, "Time-constrained project scheduling", *Journal of Scheduling*, Vol. 11, pp. 137–148.

Hartmann, S. and Briskorn, D., 2010, "A survey of variants and extensions of the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 207, pp. 1–15.

Hartmann, S. and Kolisch, R., 2000, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 127, pp. 394–407.

Herroelen, W., De Reyck, B., and Demeulemeester, E., 1999, "A classification scheme for project scheduling", In J. Weglarz, editor, *Handbook of Recent Advances in Project Scheduling*, pp. 1–26. Kluwer Academic Publishers, Dordrecht.

Hurink, J., Kok, A., Paulus, J., and Schutten, J., 2011, "Time-constrained project scheduling with adjacent resources", *Computer and Operations Research*, Vol. 38, pp. 310–319.

Kolisch, R., 1995, "Project scheduling under resource constraints", Ph.D. thesis, Berlin: Physica.

Kolisch, R., 1999, "Resource allocation capabilities of commercial project management software packages", *Interfaces*, Vol. 29, pp. 19–31.

Kolisch, R. and Hartmann, S., 2006, "Experimental investigation of heuristics for resource-constrained project scheduling: An update", *European Journal of Operational Research*, Vol. 174, pp. 23–37.

Mohring, R., 1984, "Minimizing costs of resource requirements in project networks subject to a fixed completion time", *Operations Research*, Vol. 32(1), pp. 89–120.

Neumann, K. and Zimmermann, J., 1999, "Resource levelling for projects with schedule-dependent time windows", *European Journal of Operational Research*, Vol. 117, pp. 591–605.

Zimmermann, J. and Engelhardt, H., 1998, "Lower bounds and exact algorithms for resource levelling problems", Technical report, University of Karlsruhe.

Expansions for the Resource Renting Problem

Len Vandenheede[1], Mario Vanhoucke[1,2,3] and Broos Maenhout[1]

[1] Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium
len.vandenheede@ugent.be, mario.vanhoucke@ugent.be
[2] Technology and Operations Management Area, Vlerick Business School, Reep 1, 9000 Gent, Belgium
[3] Department of Management Science and Innovation, University College London, Gower Street, London WC1E 6BT, United Kingdom

Keywords: Project Scheduling, Resource Scheduling, Renting costs.

1 Introduction

The resource renting problem (RRP), originally proposed by Nübel (2001), is one of the extensions of the well-known resource-constrained project scheduling problem (RCPSP). The RRP distinguishes itself from the common RCPSP or its other extensions by the addition of time-dependent costs for the use of renewable resources. Time-dependent costs are costs that are encountered each time unit a renewable resource is in the resource set. For instance, the wages of the staff working on a specific activity can be considered as renting costs. Further on, these costs will be called **renting costs**. Renting costs need to be paid every time a resource is in the company portfolio, even though it might not be used at that particular time unit.

Next to renting costs, the problem also takes into account time-independent costs. These costs are made every time a company adds a resource to the existing resource set. From this point on, they shall be referred to as **procurement costs**.

In the RRP the best times to procure and to lose resources are being sought. The objective is to minimize the sum of the total procurement costs and the renting costs. It can be more opportune to keep resources in the company, although, at that moment, they do not participate in any activity of the project. In this case the total amount of renting costs will be smaller than the cost to procure the resources later on again.

A small example might work enlightening. A project consists of three activities, A, B, and C. These activities should be processed one after another, as the activity-on-the-node project network in figure 1 shows. We only consider one resource type. The resource requirement of each activity is indicated above the respective activity node, whereas the activity's duration is indicated below the activity's node. The procurement cost c^p in this example equals 10. If the renting cost c^r is relatively low, for example 2, it is cheaper to keep the earlier procured resources in the portfolio. This situation is depicted as situation (a). However, if the renting cost is relatively high, for example if this cost equals 5, it is cheaper to lose the resources at time $t = 1$ and procure them again at time $t = 4$. The optimal renting policy for this case, is depicted as situation (b). An overview of the total costs of all situations is presented in the table in figure 1.

The remainder of this extended abstract is structured as follows. In section 2 a short overview of the publicized literature is presented. Our approach to solve the RRP is explained in section 3. Next, in section 4, we show the results of an experimental design on which the scatter search procedure is tested. Lastly, a conclusion can be found in section 5.

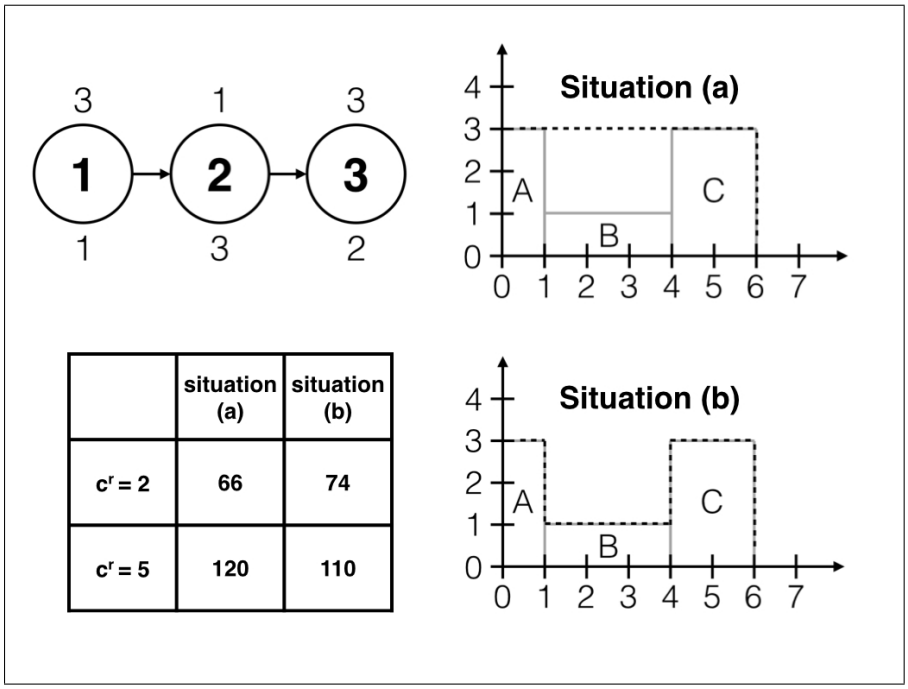


Fig. 1: Example

2 Related literature

In literature, the RRP has rarely been studied. Nübel (2001) introduced the problem and has proposed a branch-and-bound (B&B) technique to solve the RRP, extended with generalized precedence relations. Therefore, the problem Nübel handles is addressed to as RRP/max, where the max stands for the minimum and maximum time lags that are observed between the start and end times of the activities.

Another author that has applied the RRP is Ballestín (Ballestín (2007), Ballestín (2008)). He developed a number of algorithms, based on the well-known genetic algorithm, to solve the RRP. The main difference with Nübel's work is the sequence of the solution approach. Ballestín defines the optimal renting policy for a given schedule, whereas the B&B approach of Nübel determines the optimal policy while constructing the schedule. The added value of Ballestín's work is that his algorithms are able to handle larger instances than the B&B technique as proposed by Nübel.

To the best of our knowledge, there are no other procedures to solve the RRP.

3 Solution approach

As can be read in the previous section, there are only two known methods that can solve the basic RRP, this in large contrast to the numerous methods to solve the common RCPSP or popular expansions, such as the multi-mode resource constrained project scheduling problem (MRCPSP). For an overview of these methods, a good overview is presented in Kolisch and Hartmann (2006).

Because we want to elaborate several expansions on the RRP, we have developed a scatter search procedure that is able to handle both the basic RRP and potential expansions

for the RRP. The scatter search exists of the five traditional building blocks, as proposed by Glover *et al.* (2000), being the diversification generation method, the subset generation method, the solution combination method, the improvement method and the reference set update method. We mainly have focussed on the solution combination method and the improvement method. For these two methods we developed approaches that are related to the minimization of idle resources in the project schedule.

4 Preliminary results

One of the expansions is the addition of setup and setoff costs to the basic RRP. Without going too much into detail, two different kinds of setup costs and two different kinds of setoff costs exist:

- Setup costs if a resource is procured.
- Setup costs to restart an idle resource.
- Setoff costs if a resource is removed from the resource set.
- Setoff costs if a resource is made idle.

All four of them can be related to examples from real-life, which justifies their use in the RRP.

We have adapted the known methods, mentioned in section 2, towards this extended problem and developed a new set of projects through the project generator RanGen2 (Vanhoucke *et al.* (2008)). This set consists of 270 projects in total, all with their own specific characteristics. These 270 projects are tested with the known methods and our own procedure, on numerous assumptions, regarding to the level of the procurement, renting and setup/setoff costs and the imposed deadline.

In order to perform the test experiment and validate the quality of the results, a lower bound based on Nübel (2001) is used. This lower bound basically searches for the maximum amount of resources per time instance that will definitely be required. It is found by accumulating the resource requirements of every near-critical activity i for every $t \in [LS_i, ES_i + duration_i]$ and, if opportune, also incorporating the resource requirements of non-near-critical activities. Since in each schedule there will be a setup and afterwards a setoff of at least this maximum amount of resources, the sum of the costs of the setup and setoff will be added to the lower bound of Nübel. This lower bound is used in all the tests of the computational experiment.

Table 1 shows a few results of the tests we have done. They give the average deviation from the lower bound, for different combinations of procurement, renting and setup/setoff costs, with our own developed scatter search (ScS), Ballestín's genetic algorithm (GA) and Nübel's exact algorithm (B&B), averaged over different deadlines. All algorithms are truncated after 1,000 generated schedules. As can be seen, the scatter search procedure has a smaller deviation from the lower bound, which proves its efficacy.

Table 1: Overview of the results: % average deviation from the lower bound

ScS	GA	B&B
47.84	73.57	71.70

5 Conclusion and future research

In the presentation, the value-adding features of our scatter search procedure will be presented. Further on, additional results will be presented. Questions that will be answered are, amongst others: which project network indicators relate to lower project costs and does the stopping criterium has an influence on the performance of the three procedures? Additionally, other expansions to the basic RRP will be proposed. Lastly, some managerial insights will be presented.

References

Ballestín F., 2007,"A genetic algorithm for the resource renting problem with minimum and maximum time lags", *Lecture Notes in Computer Science*, Vol. 4446, pp. 25Ð35.

Ballestín F., 2008, "Different codifications and metaheuristic algorithms for the resource renting problem with minimum and maximum time lags", in *C. Cotta & J. van Hemert, eds, ÔRecent Advances in Evolutionary Computation for Combinatorial IntelligenceÕ*, Vol. 153 of *Studies in Computational Intelligence, Springer-Verlag Berlin Heidelberg*, Chapter 12, pp. 187Ð202.

Glover F., M. Laguna and R. Marti, 2000, "Fundamentals of scatter search and path relinking", *Control and Cybernetics*, Vol. 29, pp. 653Ð684.

Kolisch R., S. Hartmann, 2006, "Experimental investigation of heuristics for resource- constrained project scheduling: An update", *European Journal of Operational Research*, Vol. 174, pp. 23Ð37.

Nübel H., 2001, "The resource renting problem subject to temporal constraints", *OR Spektrum*, Vol. 23, pp. 359Ð381.

Vanhoucke M., J. Coelho, D. Debels, B. Maenhout and L. Tavares, 2008, "An evaluation of the adequacy of project network generators with systematically sampled networks", *European Journal of Operational Research*, Vol. 187, pp. 511Ð524.

Discrete-Continuous Project Scheduling with Identical Processing Rate Functions of Activities to Maximize the Net Present Value

Waligóra Grzegorz

Poznan University of Technology, Poznan, Poland
e-mail: grzegorz.waligora@cs.put.poznan.pl

Keywords: discrete-continuous project scheduling, discounted cash flows, net present value, metaheuristics.

1. Problem formulation

We consider the *Discrete-Continuous Resource-Constrained Project Scheduling Problem with Discounted Cash Flows* (DCRCPSPDCF), defined as follows (Waligóra 2014). Given is a project consisting of n precedence-related, nonpreemptable activities which simultaneously require for their processing discrete and continuous renewable resources. We assume that R discrete resources are available and r_{il}, $i = 1,2,\ldots,n$; $l = 1,2,\ldots,R$, is the (fixed) discrete resource request of activity A_i for resource l. The total number of units of discrete renewable resource l available in each time period is R_l. Additionally, a single continuous resource occurs which can be allotted to activities in (arbitrary) amounts from the interval [0,1]. The total amount of the resource available at a time is equal to 1. The amount (unknown in advance) of the continuous resource $u_i(t)$, allotted to activity A_i at time t determines the processing rate of activity A_i, as described by Eq. (1):

$$\dot{x}_i(t) = \frac{dx_i(t)}{dt} = f_i[u_i(t)], \quad x_i(0) = 0, \quad x_i(C_i) = \tilde{x}_i \qquad (1)$$

where $x_i(t)$ is the state of activity A_i at time t, f_i is a continuous, increasing function, $f_i(0) = 0$, $u_i(t)$ is the continuous resource amount allotted to activity A_i at time t, C_i is the completion time (unknown in advance) of activity A_i, $\tilde{x}_i$ is the processing demand (final state) of activity A_i. State $x_i(t)$ of activity A_i at time t is the progress in the processing of activity A_i up to time t. A positive cash flow CF_i is associated with the execution of activity A_i. The problem is to find a feasible assignment of discrete resources and, simultaneously, a continuous resource allocation that maximize the *Net Present Value* (NPV) of all cash flows of the project. The way of calculating the NPV depends on the payment model assumed. We consider three models well-known from the literature (see, e.g., Mika et al. 2005): Lump-Sum Payment at the completion of the project (LSP), Payments at Activity Completion times (PAC), and payments at Equal Time Intervals (ETI).

2. General methodology

It has been shown that for concave processing rate functions of activities parallel schedules are optimal, and the general methodology based on so-called feasible sequences has to be applied. Such a case is considered in this work.

A *feasible sequence* is a sequence of combinations of activities executed in parallel in corresponding intervals in a schedule. The ends of the intervals are defined by completions of successive activities. Such a sequence has to be precedence- and resource-feasible (with respect to discrete resources), as well as it has to assure nonpreemptability of activities. It means that each activity appears in exactly one or in consecutive combinations in the sequence. For a given feasible sequence and an assumed payment model, a nonlinear mathematical programming problem can be formulated to find the NPV-optimal continuous resource allocation (Waligóra 2014). Thus, we can find an optimal continuous resource allocation and, in consequence, an NPV-optimal schedule for this particular sequence. Next, a globally optimal schedule can be found by solving the mathematical programming problem for all feasible sequences, and choosing the one with the maximal NPV. Still, the number of all feasible sequences grows exponentially with the number of activities, and therefore local search algorithms can be applied to examine the space of all feasible sequences.

Below we present the nonlinear mathematical programming problems to be solved to find an optimal continuous resource allocation for a given feasible sequence and a chosen payment model:

Lump-Sum Payment:

maximize $$NPV = \left(\sum_{i=1}^{n} CF_i \right) \beta^{C_{\max}} \quad (2)$$

subject to $$\sum_{k \in K_i} \tilde{x}_{ik} = \tilde{x}_i, i = 1,2,...,n \quad (3)$$

$$\tilde{x}_{ik} \geq 0, i = 1,2,...,n; k \in K_i \quad (4)$$

where $$C_{\max} = \sum_{k=1}^{s} M_k^*(\tilde{\mathbf{x}}_k) \quad (5)$$

and $M_k^*(\tilde{\mathbf{x}}_k)$ is the unique positive root of the equation:

$$\sum_{A_i \in Z_k} f_i^{-1}(\tilde{x}_{ik} / M_k) = 1 \quad (6)$$

$M_k^*(\tilde{\mathbf{x}}_k)$ is the minimum length of the part of the schedule generated by the *k*-th combination $Z_k \in S$, K_i is the set of all indices of Z_k's such that $A_i \in Z_k$, and *s* is the number of combinations in the feasible sequence *S*. $C_{\max}$ is the completion time of the project, and β is the discount factor.

Payments at Activity Completion times:

maximize $$NPV = \sum_{i=1}^{n} CF_i \beta^{C_i} \quad (7)$$

subject to (3) and (4)

where $$C_i = \sum_{k=1}^{\max\{K_i\}} M_k^*(\tilde{\mathbf{x}}_k) \quad (8)$$

is the completion time of activity A_i, and $M_k^*(\tilde{\mathbf{x}}_k)$ is the positive root of Eq. (6).

Payments at Equal Time Intervals:

maximize $$NPV = \sum_{h=1}^{H} P_h \beta^{T_h} \quad (9)$$

subject to (3) and (4)

where H is the assumed number of payments, P_h is the payment at time point T_h, $T_h = \begin{cases} h \cdot T & \text{for } h = 1,2,\ldots,H-1 \\ C_{\max} & \text{for } h = H \end{cases}$, $T = \left\lceil \frac{C_{\max}}{H} \right\rceil$,

and $C_{\max}$ is given by Eq. (5) in which $M_k^*(\tilde{\mathbf{x}}_k)$ is the positive root of Eq. (6).

3. Identical processing rate functions

In this work a special case of the DCRCPSPDCF is considered, where the processing rate functions of all activities are identical and concave, i.e. $f_i(u_i) = f(u_i)$, $i = 1,2,\ldots,n$. Such situations can often occur in practice, when activities of a project have the same processing characteristic. Notice that according to Eq. (1) the change of the state of an activity over time is a function of the amount of the continuous resource allotted to the activity at a time. The change of the activity state represents the progress in processing the activity or, in other words, its processing rate. If the processing rate functions of all activities are the same, then each activity will be processed with the same rate dependent on the allotted resource amount only. If we temporarily assume that the continuous resource allocation is close to uniform (i.e. each activity gets approximately the same amount of the resource) in a given interval, then the actual duration of activity A_i depends only on its processing demand (i.e. its final state) $\tilde{x}_i$. This brings an idea that, in order to construct a schedule, continuous resource division may be temporarily neglected, and processing demands of activities can be treated as their actual durations. Next, a feasible schedule can be constructed

satisfying all precedence and discrete resource constraints. In that way, at this stage we obtain a problem similar to the classical discrete RCPSPDCF in which a feasible schedule with the maximal NPV is to be found. As it is known, the common solution representation for the RCPSPDCF is an activity list (AL), for which a feasible schedule can be constructed using the serial Schedule Generation Scheme (SGS) decoding rule. The scheme acts quite simply. It just takes the first yet unscheduled task A_i from the list, and schedules it at the earliest possible starting time that does not violate precedence or resource constraints. The same can be now done for our DCRCPSPDCF problem. Next, based on the constructed schedule, the completion times of consecutive activities define the ends of successive intervals, and for each interval a combination of activities processed in parallel can be constructed. In consequence, a feasible sequence can be obtained as a sequence of such combinations, as discussed in Section 2. Finally, an optimal continuous resource allocation can be found by solving a nonlinear mathematical programming problem for the constructed feasible sequence. Below we present a version of the DCSGS procedure designed for the criterion of maximization the NPV. It is named DCSGS-NPV, and the main difference to the original one is Step 4 where a nonlinear mathematical programming problem related to the assumed payment model has to be solved.

PROCEDURE DCSGS-NPV

Step 1. For each activity A_i, $i = 1,2,\ldots,n$, set d_i equal to $\tilde{x}_i$.

Step 2. Construct a precedence- and discrete resource-feasible schedule for the given instance of the DCRCPSPDCF and the given activity list AL, using the serial SGS decoding rule and assuming d_i as the duration of activity A_i.

Step 3. Construct a feasible sequence S of combinations of activities processed in parallel in successive intervals, where the intervals are defined by completion times of consecutive activities.

Step 4. Find an optimal NPV for feasible sequence S by solving a relevant nonlinear mathematical programming problem for this sequence and the given payment model (see Sect. 2).

4. Local search approach

Solution representation. As already mentioned, we can find an NPV-optimal schedule for a given feasible sequence. Thus, a feasible solution for a local search algorithm should correspond to a feasible sequence. Such a feasible solution is represented by two *n*-element lists. The first one is a precedence-feasible permutation of activities in which each activity has to occur after all its predecessors and before all its successors. This structure is called the *Sequence of Activity Starts* (SAS), and defines the order in which activities are started. The second one is also a precedence-feasible permutation of activities, and defines the order in which activities should be completed in order to fulfil the discrete resource constraints. This list is called the *Sequence of Activity Completions* (SAC). The rule transforming a pair of lists (SAS, SAC) into a feasible sequence S is quite simple. The two-list representation has been previously used for some discrete-continuous project scheduling problems (see, e.g., Waligóra 2008).

Objective function. The objective function for a feasible solution is defined as the optimal NPV for the corresponding feasible sequence, calculated under an optimal continuous resource allocation obtained as a solution of the relevant nonlinear mathematical programming problem.

Starting solution. The initial SAS and SAC lists are generated by setting all activities on both the lists in an ascending order.

Neighbourhood. Neighbours of the current solution are generated by swapping activities on lists SAS and SAC. More precisely, an *activity swap* operator is used which swaps two activities on the list that may be swapped without violating the precedence constraints. This operator can be applied to both the SAS and the SAC list.

Stop criterion. The stop criterion has been defined as an assumed number of visited solutions, i.e. an assumed number of the objective function calculations (equal to 1000).

Metaheuristics. We have implemented *Simulated Annealing* (SA) and *Tabu Search* (TS) metaheuristics to compare their performances for the considered problem. In the SA implementation, the *adaptive cooling scheme*, known as *polynomial-time* (Aarts and Korst 1989), is used to control the cooling process where the only exception is the stop criterion. In the TS algorithm, the *Tabu Navigation Method* (TNM) (Glover and Laguna 1997) is used to manage the tabu list. Also an *aspiration criterion* is applied in order to allow a tabu move if it leads to a solution better than the best found so far. The length of the tabu list has been set at 7.

5. Computational experiment

The experiment is divided in two parts. In the first experiment a comparison to optimal solutions is made for $n = 10$ activities. To this end, each generated instance was solved to optimality by fully enumerating the set of all feasible sequences. The second experiment was performed for $n = 12$, 15, and 20 activities. In this experiment optimal solutions are not known, thus, the relative solution is the best one found by either of the algorithms tested.

The results obtained by both the metaheuristics are very promising, although we can state that SA performs slightly better especially in terms of the number of optimal solutions found. Up to 60% instances solved to optimality (for SA), keeping the average relative deviation from optimum below 0.5% for SA and 0.6% for TS, suggests that both the proposed implementations can be on average quite effective for the considered problems. However, the number of optimal solutions found (especially by TS) for such a small size problem ($n = 10$) is not overwhelming. It shows that when optimal solutions are desired, searching over feasible sequences cannot be replaced by searching over activity lists, even for identical processing rate functions. SA produces better results for smaller number of activities, whereas TS starts to perform better when the problem size grows. It can be concluded that SA is rather predisposed for smaller DCRCPSPDCF problems, whereas TS for larger ones. This, however, should still be confirmed by further experiments with more activities. SA performs better for smaller values of the discount rate, whereas TS for larger values of α. SA is visibly better for the LSP model, for the PAC model both the algorithms are rather comparable. For the periodic payment model ETI, SA is better when the payments are made less often, whereas larger number of payments improve the performance of TS.

6. Conclusions

The conclusions from the experiments are, in general, the following. SA produces better results for smaller problem sizes, and, under an assumed number of activities, for smaller values of the discount rate. TS performs better for larger problems as well as for greater values of the discount rate. Besides, for periodic payments TS gets better when the number of payments grows, whereas SA prefers payments made more rarely, and becomes most effective for one payment at the end (LSP model). Moreover, the comparison to optimal solutions shows that if optimum schedules are required, feasible sequences have to be examined, not activity lists, even for identical processing rate functions. The future research can be carried out in three directions. Firstly, improvements of the proposed metaheuristics are very likely possible and/or implementing other (also hybrid) metaheuristic approaches. Secondly, generalizing the considered problem can be done in several ways, e.g. by incorporating negative cash flows or multiple execution modes. Finally, heuristic procedures for allocating the continuous resource should be developed, in order to shorten the computational times, as well as to analyze larger problem instances.

Acknowledgements

This research was partially supported by the Polish National Science Centre.

References

Aarts E.H.L. and J.H.M. Korst, 1989, "Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing", Wiley, Chichester.

Glover F. and M. Laguna, 1997, "Tabu Search", Kluwer, Norwell.

Mika M., G. Waligóra and J. Węglarz, 2005, "Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models", European Journal of Operational Research, Vol. 164 (3), pp. 639-668.

Waligóra G., 2008, "Discrete-continuous project scheduling with discounted cash flows – a tabu search approach", Computers & Operations Research, Vol. 35 (7), pp. 2141-2153.

Waligóra G., 2014, "Discrete-continuous project scheduling with discounted cash inflows and various payment models – a review of recent results", Annals of Operations Research, Vol. 213 (1), pp. 319-340.

Optimal scheduling policies for k-out-of-n systems with imperfect tests

Wenchao Wei[1], Fabrice Talla Nobibon[2] and Roel Leus[1]

[1] KU Leuven, Belgium
{ Wenchao.Wei; Roel.Leus }@kuleuven.be
[2] Federal Express (FedEx), Brussels, Belgium
tallanob@gmail.com

Keywords: system diagnosis, k-out-of-n, imperfect tests, sequencing and scheduling.

1 Introduction

System health monitoring for complex systems, such as a space shuttle, an aircraft or integrated circuits, is crucial for reducing the likelihood of accidents due to sudden failures, and for improving system availability. It is also imperative that systems be tested before being put into operation, in order to ascertain their functionality. The same inspection procedure may be repeated thousands of times, and so it is important to minimize the total expected costs in the long run. We focus on the k-out-of-n configuration, which requires that, for the overall system to be functional, at least k out of the total n components must be working. This configuration has a wide range of applications in both industrial and engineering systems (Ünlüyurt 2004). The functionality of the system is discovered by testing its components. A solution is an inspection *strategy*, which is a dynamic decision rule that decides in which order to test the components, which respects specific stopping criteria. More specifically, we develop algorithms for finding optimal strategies that minimize the expected testing expenses. Throughout the text, we use the terms 'strategy' and 'policy' interchangeably.

In this paper, we study the case where individual component tests are *imperfect*, which means that a test can observe a component as working when in reality it is down, and vice versa. In a computer's operating system, for example, system check-up may be wrongly alarmed by a fault that does not exist, and the system might operate improperly, possibly leading to extra memory usage. In the policy classes studied below, decisions are made dynamically, meaning that they are conditional on the observations (the outcomes) of the previous tests.

The contributions of our work are fourfold: (1) we describe a general setting for k-out-of-n testing with imperfect tests; (2) we underline the importance of the possibility of fixing error probabilities conditional on the test outcomes rather than on the actual state of the components; (3) we examine different classes of scheduling policies and discuss global optimality of each of the classes; and (4) we present a polynomial-time algorithm to find a globally optimal policy. In the process, we also define and analyze other problem variants of k-out-of-n testing.

2 Definitions and problem statement

We consider a system consisting of n components with component set $N = \{1, 2, \ldots, n\}$. The functionality of the system is discovered by testing sequentially its components on a single test machine. Each component is in one of two *states*: either working (up) or not working (down). The system functions (succeeds) if at least k ($k \leq n$) out of the n components are working and malfunctions (fails) if at least $(n-k+1)$ components are not

working. Each component i is tested at most once, a setting that has been called 'single-pass' testing (Nachlas *et al.* 1990). We refer to this single test of component i as 'test i.' The *outcome* of test i is written as $x_i \in \mathbb{B} = \{0, 1\}$ and is also binary: it is either positive or negative; positive (corresponding with $x_i = 0$) means the test detects a fault, negative ($x_i = 1$) means the opposite. All outcomes can be gathered in an n-dimensional binary vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{B}^n$.

Each component $i \in N$ has a probability p_i to be working, which is known a priori. We study the situation where the measurements (tests) are imperfect: the outcome can be positive while the component is actually working, and vice versa. We assume that each test has a probability α of producing the wrong outcome; we will refer to α as the *predictive error rate*. Such a common error probability α for all component tests may for instance be inherent in the design of a single unreliable test machine. Specifically, for component i, if the test result is positive then we know the actual state of i is down with probability $(1 - \alpha)$ and up with probability α, and mutatis mutandis for a negative outcome. Clearly, we can assume $\alpha < 0.5$, because we can invert the clue of the test otherwise. We define χ_i as the probability that the outcome of test i is negative (x_i equals 1); it holds that $p_i = \chi_i(1 - 2\alpha) + \alpha$ $(i = 1, 2, \ldots, n)$. For ease of notation, we also define $\lambda_i = 1 - \chi_i$ as the probability that test outcome i is positive. Let X_i represent outcome i before testing, which is a Bernoulli random variable with parameter χ_i, and denote by $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ the associated vector of random variables. The realization of each X_i is known only at the end of test i. We assume all variables to be mutually independent.

A *schedule* $\mathbf{s} = (s_1, s_2, \ldots, s_{|\mathbf{s}|})$ is an ordered subset of N; s_t is the index of the test in position t in the schedule $\mathbf{s}$. Let Σ be the set of all schedules, and value c_i represents the cost of test i. We define cost function $f(\mathbf{s})$ for schedule $\mathbf{s}$ as $f(\mathbf{s}) = \sum_{t=1}^{|\mathbf{s}|} c_{s_t}$.

A solution to the sequencing problem under study is a testing policy, which uses diagnosis experience from test outcomes gathered over time. At each stage, the policy prescribes the next step to be taken: either select a new test to conduct, or stop; the stop/continue decision when not all components have been tested yet is based on the accuracy regarding the system's state. We study the optimization problem of designing a test policy Π with minimum expected total diagnostic costs. The inspection procedure stops as soon as a specified threshold value T is reached (e.g., $T = 95\%$) for the probability that the system is either functional or disfunctional. If this stop criterion is never fulfilled then all the n tests are performed. Below, we will use the terms 'guarantee' and 'accuracy' interchangeably to refer to value T; in the context of isolating a single failure source, this value has been referred to as a 'level of confidence.' The value of T should be determined from historical data or by the requirements of the system.

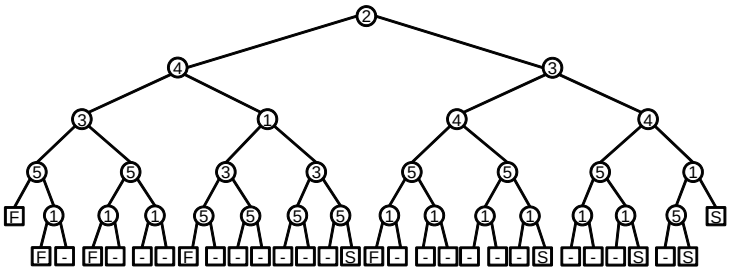


Fig. 1. An illustration of a policy for a k-out-of-5 system

A policy can also be represented by a binary decision tree (BDT). In such a BDT, each non-leaf node is labeled with the index of a component to be tested and has two child nodes. If the component is faulty then the left branch is entered, otherwise the right subtree is taken. Each leaf node either represents a conclusion of a specific system state (successful (S) or failed (F)) with the required accuracy, or indicates that all components have been tested but the state of the system is not identified (labeled by $-$). For a given outcome, the policy generates the schedule stepwise from the root node to a leaf node. Figure 1 depicts a policy for a k-out-of-5 system where four working or failing components are needed to draw a conclusion with the required accuracy; for the intermediate cases, the system state remains indeterminate.

Following the literature on stochastic scheduling (Igelmund and Radermacher 1983), a policy Π can also be seen as a function $\Pi : \mathbb{B}^n \to \Sigma$ that maps outcomes $\mathbf{x}$ to schedules. Note that several outcomes may be mapped to the same schedule. Our objective is to find a policy Π^* within a specific class that minimizes $\mathbb{E}[f(\Pi(\mathbf{X}))]$, with $\mathbb{E}[\cdot]$ the expectation operator with respect to $\mathbf{X}$. Specifically,

$$\mathbb{E}[f(\Pi(\mathbf{X}))] = \sum_{\mathbf{x}\in\mathbb{B}^n} \left(\prod_{i:x_i=1} \chi_i \right) \left(\prod_{i:x_i=0} \lambda_i \right) f(\Pi(\mathbf{x})). \tag{1}$$

We call a policy *globally optimal* if it achieves the minimum expected cost over all possible policies. A decision is *dominant* if by applying it, we do not lose global optimality. Tests are conducted one at a time, so a sequence of (a subset of) jobs indeed defines a schedule; this is a dominant decision for our objective function (but the same is not true for other objectives such as makespan minimization).

3 Optimal policies and complexity results

We define a new testing problem (with perfect tests) called *generalized testing problem*, defined similarly as the classic k-out-of-n testing problem in that an instance is defined by a component set N with parameters c_i and p_i, but instead of parameter k a generalized testing instance takes two parameters k_1 and k_0 and the system diagnosis continues until either k_1 working components or k_0 failing components are found. Clearly, the classic k-out-of-n testing problem is a special case of the generalized problem where $k_1 = k$ and $k_0 = n-k+1$. The *conservative* k-out-of-n testing problem is defined similarly as traditional k-out-of-n testing (with perfect tests), except that we perform tests either until we have observed k tests with negative outcomes, or we have performed all n tests. This problem is a subproblem of generalized testing with $k_1 = k$ and $k_0 = n$.

We define the guarantees $\theta_1(r,t)$ and $\theta_0(r,t)$ as the probability that the system works or fails conditional on the number r of tested components and the number t of observed negative or positive outcomes respectively. Let a $(0 \le a \le n)$ be the smallest integer such that $T \le \theta_1(n,a)$, and integer b $(0 \le b \le n)$ is the lowest integer such that $T \le \theta_0(n,b)$. The time complexity of finding values a and b is $O(n^4)$ (Wei *et al.* 2013). The following two lemmas show how to transform an instance of k-out-of-n testing with imperfect tests into an instance of generalized testing with perfect tests. All proofs are in our working paper Wei *et al.* (2013).

Lemma 1. *For imperfect testing, if the outcome of the current test is the a-th negative one then it is a dominant decision to stop the test procedure immediately.*

Lemma 2. *For imperfect testing, if the outcome of the current test is the b-th positive one then it is a dominant decision to stop the test procedure immediately.*

For the example instance described supra, for instance, depending on the parameters, it might be that $k = 3$ but $a = b = 4$. It also follows that when b does not exist then the conclusion of system failure can never be drawn with the required accuracy. If neither a nor b exist then the testing sequence becomes irrelevant because the diagnosis will conduct all the n tests under all possible outcomes. The setting in which both a and b exist and $a + b \leq n$, can be considered as, in some sense, undesirable.

Lemma 3. *If $a + b \leq n$ then there exist outcomes $\mathbf{x}^*$ and policies Π_1 and Π_2 such that Π_1 under $\mathbf{x}^*$ will first observe a working components while less than b failing components are found (and thus conclude that the system functions with the pre-specified accuracy T), whereas Π_2 under $\mathbf{x}^*$ will first observe b failing components while less than a working components are found (and thus halt the diagnosis with the conclusion of system failure). This phenomenon will never occur when $a + b > n$.*

Below, we examine optimal policies for the following two cases: exactly one of the parameters a and b does not exist, and both a and b exist and $a + b > n$.

A policy is called *dominant* when identical decisions are made at all decision moments for which the decisions made in the past lead to a situation with the same number of components tested and the same number of negative outcomes observed.

Lemma 4. *There always exists a dominant policy that is globally optimal.*

In general the input size of a policy is exponential in the number of jobs. Therefore a more compact description is useful. Inspired by priority policies for standard scheduling problems (Pinedo 2012), we also study *elementary policies*, which are characterized by a total order of the component set N. Such an order can be represented by a list (or permutation) $L = (l_1, l_2, \ldots, l_n)$ of the elements of N. The policy tests the components one by one in the order of the list (so first l_1, then l_2, etc.) until the system's guarantee reaches the threshold value. For a given outcome $\mathbf{x}$ of a k-out-of-n system with imperfect tests, the elementary policy Π characterized by a list L generates a unique schedule $\Pi(\mathbf{x}; L)$ by iterating through the list from left to right. The schedule stops when either θ_1 or θ_0 reaches threshold value T or all the components have been tested.

Proposition 1. *For conservative k-out-of-n testing with perfect tests, the elementary policy represented by the permutation of all n components arranged in nondecreasing order of c_i/χ_i is optimal.*

Proposition 2. *For conservative k-out-of-n testing with perfect tests, an optimal elementary policy is also globally optimal.*

It can also be seen that when either a or b does not exist, then the k-out-of-n testing problem with imperfect tests reduces to conservative $a(b)$-out-of-n testing.

Corollary 1. *When exactly one of the parameters a or b does not exist then the imperfect k-out-of-n testing problem is polynomially solvable.*

When both a and b exist, then the k-out-of-n testing problem with imperfect tests reduces to generalized testing with $a = k_1$ and $b = k_0$. There exist counterexamples showing that the class of elementary policies in this case may 'miss' the global optimum. We can describe another class of compact policies, however, namely the class of interrupted block walking (IBW) policies, for which it can be shown that there always exists a globally optimal IBW policy. More details and illustrations can be found in our working paper Wei *et al.* (2013). This lead to:

Theorem 1. *The generalized testing problem is polynomially solvable when $k_0 + k_1 > n$.*

Corollary 2. *When $a + b > n$ then there always exists a globally optimal IBW policy for the imperfect k-out-of-n testing problem; such a policy can be found in polynomial time.*

References

Hellerstein L., O. Özkan and L. Sellie, 2011, "Max-throughput for (conservative) k-out-of-n testing", *Lecture Notes in Computer Science, Chapter 72*, Vol. 7074, pp. 703-713.

Igelmund G. and Radermacher F., 1983, "Preselective strategies for the optimization of stochastic project networks under resource constraints", Vol. 13, pp. 1-28.

Nachlas J., S. Loney and B. Binney, 1990, "Diagnostic-strategy selection for series systems", *IEEE Transactions on Reliability*, Vol. 39, pp. 273-280.

Pinedo M., 2012, "Scheduling: Theory, Algorithms, and Systems", *Springer*.

Ünlüyurt T., "Sequential testing of complex systems: A review", *Discrete Applied Mathematics*, Vol. 142, pp. 189-205.

Wei W., F. Talla Nobibon and R. Leus, 2013, "Sequential diagnosis of k-out-of-n systems with imperfect tests", *Research report* KBI_1315, Department of Decision Sciences and Information Management, FEB, KULeuven.

Outline of Multi-Objective Model for Training Device Selection (TraDE)

Slawomir Wesolkowski[1] Nevena Francetić[1] and Nathaniel Horvath[2], Stuart C. Grant[3]

[1] Devence Research and Development Canada, Ottawa, ON Canada
s.wesolkowski@ieee.org
[2] University of Waterloo, Waterloo, ON Canada
[3] Defence Research and Development Canada, Toronto, ON Canada

Keywords: training, locations, multi-objective optimization.

1 Introduction

A recurring military problem concerns how training is planned. Since training programs can utilize multiple training devices with varying costs and training capabilities, selecting the types of devices required is a complex trade-off problem. The placement of these devices is critical due to the time and costs involved in travelling to the training device. Here we introduce a scheduling-based model, Training Device Estimation (TraDE), to study the device mixes per location which enable completing training and are the most favourable considering costs and training time.

The paper is organized in the following manner. We give a detailed problem description and literature overview in Section 2. In Section 3, we outline the TraDE model. Section 4 gives an overview of the preliminary results. Section 5 contains some concluding remarks.

2 Problem Description and Literature Overview

Every soldier has to complete mandatory training consisting of a set of *tasks* such as quick aim shooting or safe handling. All tasks must be completed. For each task, there is a list of compatible *devices* that can be used to complete the task. Device types can vary from a rifle at a range to a computer-simulated combat environment. It is assumed that all compatible device types for a task are used to achieve the same level of proficiency in that task. Their training effectiveness is measured in the time duration of the task. Device types also vary in cost incurred for completing a training task.

Soldiers needing military training are stationed at various *locations*. When training is required, the unit (or a part of it) may have to travel to the training location, complete the training, and return to their home base. Since some tasks can take multiple days to train, each soldier incurs costs for staying in the training location until the training task has been completed.

The goal of this study is to determine a *device mix per location*, i.e. the number of devices of each type required at a given location, and to make a device use schedule while minimizing the capital, operating and travel costs, and the total time spent in training.

Given the wide variety of the training tasks and the large number of possible training devices, finding a good device mix is difficult (Cohn 2008). There are many key parameters to take into account. For example, the triangle model (Daly *et. al.* 2009) considers device fidelity, training delivery method, and training content to determine good device mixes. On the other hand, the FAPV approach analyzes the stage of the trainee's learning process to select an effective training environment (Franki *et. al.* 2000). The Army Training (Murty *et. al.* 1995) and the Stochastic Fleet Estimation (SaFE) (Wojtaszek *et. al.* 2013) models approach the problem from a resource allocation standpoint. The problem

of selecting devices is similar to two-dimensional bin or strip packing (Lodi *et. al.* 2002), where bins correspond to training devices. However, in TraDE, device types differ in their capacities, and, hence, bins are not identical. Thus, the usual bin-pakcing techniques are not applicable.

3 TraDE Model

Training device selection is similar to the well-understood fleet mix analysis problem (Wojtaszek *et. al.* 2012). Our solution to training device selection adapts and extends SaFE (Wojtaszek *et. al.* 2013), which uses a multi-objective evolutionary optimization algorithm to obtain an approximation of the Pareto Front for an aircraft fleet mix problem. The main difference between SaFE and TraDE is the necessity in the training problem to also determine training device locations. In SaFE, there is a single base/depot for all aircraft. In addition, a large number of tasks needs to be completed by each aircraft or set of aircraft. In TraDE, soldier capacity of a training device is similar to cargo/passenger capacities in the fleet mix problem for transport aircraft. In the training problem, bins have two dimensions (soldier capacity and training duration) while in the fleet mix problem there is a one-dimensional bin for aircraft sortie (i.e., flight from point A to point B) duration.

In TraDE, the two main input variables are training tasks (such as rifle position and hold, or marksmanship exercise) and training devices, which can vary from a fire range to a simulator or a computer. Each device type is characterized by the following four values: the soldier capacity, $c(i)$, the acquisition cost, the annual operating cost, and possible locations where the device can be placed, limitted due to weather, terrain or other constraints. A device *configuration* is a tuple specifying the number of devices of each type used. Given the soldier capacities for each device type, we determine all possible minimum configurations of devices adequate for a given training task of all soldiers. The set of all device configurations for each task is the *configuration set* and is used to bound our solution space. Considering the number of training tasks and the number of locations where we can place each device in a configuration, we obtain a vast solution space. This space is then searched using an implementation of NSGA-II (Deb 2001).

The members of the population in the algorithm have two-part chromosomes. Each gene corresponds to one training task and has the following form:

$$(n_1, n_2, \ldots, n_D) \times ((l_{11}, l_{12}, \ldots, l_{1n_1}),\ \ (l_{21}, l_{22}, \ldots, l_{2n_2}), \ldots, (l_{n_D1}, l_{n_D2}, \ldots, l_{n_Dn_D}))\,, \quad (1)$$

where D is the number of device types. The first part corresponds to the device configuration and $\sum_{i=1}^{D} n_i \cdot \mathrm{c(i)} \geq$ total number of soldiers. In the second part, eligible locations are assigned to each device in the chosen configuration. n.

The fitness of each population member is evaluated with respect to four objective functions:

- *acquisition cost*: the total cost of obtaining new devices;
- *operating cost*: the sum over all tasks and all soldiers of device-dependent task completion costs;
- *travel cost*: sum over all tasks and all soldiers of all travel costs between the home base and the device location;
- *training time*: the sum over all soldiers of the time spent to complete all training tasks at allocated devices.

For the computation of the operating cost, travel cost and training time, we schedule soldiers for a training task on a device at a particular location giving us a training schedule as a co-product of the objective functions evaluation. Although cost objectives could be

aggregated into one cost objective, this is not done given the different types of budget these costs represent; i.e., most governments usually have distinct budgets for different purposes: capital acquisitions, personnel travel and operating funds. Moreover, we are interested in looking at the trade-off space between all these values without having to set different weights on each objective (determining weights according to decision maker preferences is a very complex problem (Abbass *et. al.* 2009)).

The population is randomly initialized. In addition, the user has the option of introducing "seeds" with some approximate best-fit solutions in order to bootstrap the genetic algorithm. A seeding mechanism has been implemented in order to generate an optimal or nearly optimal solution with respect to each objective separately from all the other objectives.

In each generation, population members are sorted in nondominated fronts. A member in front i has a probability of $\frac{1}{i}$ of being picked as a parent for the next generation. In this way, members that are more likely to produce better solutions are picked with higher probability to be parents, which speeds convergence for a small number of objectives (Wagner *et. al.* 2013). The crossover and mutation operators consider one task at a time. The crossover operator picks with equal probability from one of the two parents the device configuration and locations for the task. Then, the mutation operator with equal probability keeps or replaces the child's device configuration for the task with a randomly chosen configuration from the configuration set. If the device configuration is replaced, new device locations are randomly assigned.

Finally, for every population member, we compute the device mix per location as follows. First, we determine the total time the devices of each type at a given location are used. Given our restrictions on annual device usage, we compute the number of required devices of each type at the given location.

4 Results

Subject matter experts helped in devising the input data set. We implemented NSGA-II and our optimization objectives in MATLAB. We performed 10 experiments on a Windows7 PC with 3GHz 12-core CPU and 18GB RAM. For each experiment, we set the following parameters: a population of 200, 10,000 generations and a mutation rate of 0.25. Each experiment took approximately 12 hours to complete. The results of all 10 experiments were combined and re-sorted to obtain a new nondominated front consisting of 1030 solutions. Comparing the four-tuples of the objective values, the nondominated fronts of individual experiments had on average 51.3% of solutions present in the combined nondominated front, with a standard deviation of 0.71%[1].

Our solution set exhibits certain expected traits. The correlation coefficient between two devices with almost the same training characteristics was -0.7773. Even though these devices have similar functionality, their training time per task and cost per use differ; hence, they cannot completely replace one another.

No solution in the combined nondominated front utilizes all of the existing devices, indicating that some of the existing resources are redundant. Solutions which have capital cost equal to zero (but high operating costs) use only a subset of the existing devices. Solutions with the smallest operating cost (for example, see Fig. 1) favour the use of 'Device B' at almost all locations. The cost per use of 'Device B' in training is on average 2.2% of the cost per use of other compatible devices for the given training tasks, but its capacity is considerably smaller (on average only 9.8% of the capacity of other devices). Figure 2 shows a solutions for the minimum total training time (similar to the minimum

[1] For better convergence, the number of generations should be increased.

travel cost solution). Notice that the majority of devices in these solutions are placed at Locations 1, 4 and 5, which are the home bases for 66% of all soldiers. As expected, this lowers the travel cost and indicates that new devices can be used at these locations for more time efficient training.

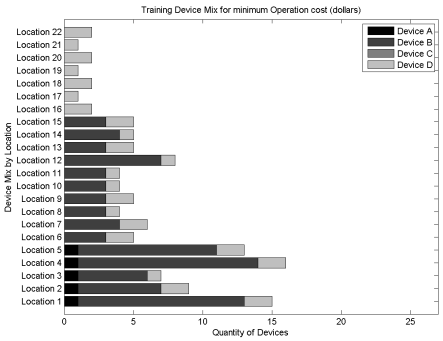


Fig. 1. A device mix with minimum optimal cost objective value

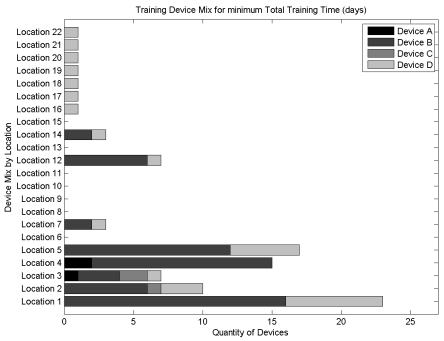


Fig. 2. A device mix with minimum training time objective value

5 Conclusion

Scheduling devices to accomplish training missions is a complex problem, especially when one must consider the optimal location placement of the devices. TraDE provides a set of solutions which indicate trade-offs between multiple objectives. Solutions can also be used to identify existing devices which are redundant and to identify device locations for more efficient training.

References

H. A. Abbass and A. Bender, "The Pareto operating curve for risk minimization in life and robotics," *Artificial Life Robotics*, vol. 14, no. 4, pp. 449–452.

J. Cohn, 2008, "Building Virtual Environment Training Systems For Success," *Virtual Environments for Training and Education: Developments for the Military and Beyond*, Westport, Praeger Security International, pp. 193–207.

M. Daly and D. Thorpe, 2009, "Balancing Simulated and Live Naval Fleet Training," *Interservice/Industry Training Simulation Education Conference*, Orlando.

K. Deb, 2001, " Multi-Objective Optimization using Evolutionary Algorithms," Chichester: John Wiley & Sons.

G. Frank, R. Helms and D. Voor, 2000, "Determining the Right Mix of Live, Virtual and Constructive Training," *Interservice/Industry Training Simulation Education Conference*, Orlando.

A. Lodi, S. Martello, and M. Monaci, "Two-dimensional packing problems: A survey," *European Journal of Operational Research*, vol. 141, no. 2, pp. 241–252.

K. Murty, P. Djang, W. Butler and R. Laferriere, 1995, "The Army Training Mix Model," *Journal of the Operational Research Society*, no. 46, pp. 294–303.

M. Wagner and T. Friedrich, 2013, "Efficient parent selection for Approximation-Guided Evolutionary multi-objective optimization," *IEEE Congress on Evolutionary Computation*, pp. 1846–1853.

D. Wojtaszek and S. Wesolkowski, "Military Fleet Mix Computation and Analysis," *IEEE Computational Intelligence Magazine*, vol. 7, no. 3, pp. 53–61.

D. Wojtaszek and S. Wesolkowski, "Evaluating the Flexibility of Military Air Mobility Fleets," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. Early Access Online.

Scheduling of assessment centers: an application of resource-constrained project scheduling

Adrian Zimmermann and Norbert Trautmann

University of Bern, Switzerland
adrian.zimmermann@pqm.unibe.ch and norbert.trautmann@pqm.unibe.ch

Keywords: Service Management, Assessment Center, RCPSP, Resource Flows

1 Introduction

The planning situation we deal with has been reported to us by a Swiss service provider in the human resources sector. The service provider organizes assessment centers for companies; in such an assessment center, so-called assessors (e.g., psychologists or senior managers) assess the potential and the skills of candidates for a position to be filled.

During an assessment center, each candidate performs a series of exercises. Each exercise consists of three phases. First, the candidate prepares the exercise; for some exercises, this preparation phase is omitted. Second, the candidate performs the exercise and is thereby observed by one or, for some exercises, two assessors; in some exercises (so-called role plays), additionally an actor takes part. Third, the assessor(s) and the actor evaluate the performance of the candidate during the exercise; for some exercises, this evaluation phase is omitted. To prevent time-consuming discussions between the assessors, and to ensure an objective overall assessment, each candidate should be observed by approximately half of all available assessors. In addition, some assessors may not observe certain candidates because they know each other (no-go relationship). Eventually, for each candidate a lunch break has to be scheduled within a prescribed time window. In total, the assessment center may not take longer than one working day. The planning problem at hand consists of determining start times for all exercises and lunch breaks and of assigning the required number of assessors to the candidates for each exercise, such that the duration of the assessment center is minimized.

In Grüter *et. al.* (2013), we have presented a problem-specific formulation of this problem as a mixed-integer linear program (MILP). Apart from that, to the best of our knowledge, this problem has not been treated in the literature.

In this paper, we formulate this problem as a resource-constrained project scheduling problem, and we provide a corresponding MILP formulation based on the resource-flow formulation presented in Koné *et. al.* (2011). We use the resource-flow information to assign the candidates and the exercises to the assessors. We enhance the model performance by adding some symmetry-breaking constraints, valid inequalities, lower bounds, and an integrality constraint. Our computational results indicate that for two real-world problem instances, the model outperforms the model presented in Grüter *et. al.* (2013).

The remainder of this paper is structured as follows. In Section 2, we present our MILP formulation. In Section 3, we report our computational results. In Section 4, we conclude.

2 Model formulation

For each candidate, we interpret the execution of each exercise as an individual activity of the project; furthermore, the lunch break of each candidate also represents an activity. The resources of the project are as follows: (a) each candidate is modeled as a renewable resource with capacity 1; (b) the set of all assessors is modeled as one renewable resource,

the capacity of which equals the number of assessors; (c) the set of all actors is modeled as one renewable resource, the capacity of which equals the number of actors. There are no time lags or precedence constraints between the activities of the project. However, we must take into account that due to the preparation phase and the evaluation phase, the various resource types are not always utilized during the entire duration of the activities.

We use the following notation. The asterisks indicate decision variables.

K	Set of resources
$K^C \subset K$	Set of resources corresponding to candidates
$K^O \subset K$	Set of assessors and actors resources ($K^O = \{k^A, k^R\}$)
A, R	Sets of assessors and actors, respectively
N	Set of candidate-assessor pairs (c, a) with a no-go relationship
I	Set of activities $i = 1, \ldots, n$
I_k	Set of activities that require resource k
I^S	Set of identical activities that belong to an arbitrarily chosen exercise
I^L	Set of lunch breaks
ES^L,LS^L	Earliest (ES^L) and latest (LS^L) start time for the lunch breaks
p_i	Duration of activity i
δ_i^C	Preparation time of activity i for candidates
δ_i^A,δ_i^R	Evaluation time of activity i for assessors and actors, respectively
r_{ik}	Amount of resource k required by activity i
k^A, k^R	Resources representing the sets of all assessors (k^A) and actors (k^R)
M	Sufficiently large number (i.e., length of a working day)
$* D$	Duration of the assessment center
$* F_{cij}^C$	$= 1$, if there is a flow on resource $c \in K^C$ from activity i to j; $= 0$, else
$* F_{kij}^O$	Amount of resource $k \in K^O$ transferred from activity i to activity j
$* S_i$	Start time of activity i
$* X_{cij}^C$	$= 1$, if activity i is executed before act. $j > i$ on resource $c \in K^C$; $= 0$, else
$* X_{kij}^O$	$= 1$, if activity i is executed before activity j on resource $k \in K^O$; $= 0$, else
$* Y_{ca}$	$= 1$, if assessor a is assigned to candidate c at least once; $= 0$, else
$* Z_{ia}$	$= 1$, if assessor a is assigned to activity i; $= 0$, else

To handle the time-varying resource utilization during the course of the activities, we introduce sequencing variables X_{kij}^O for each resource $k \in K^O$. X_{kij}^O must be equal to 1 if one or several units of resource k are transferred from activity i (at the completion) to activity j (at the start). Otherwise the value of X_{kij}^O can be 0, i.e., activities i and j are processed simultaneously, or j finishes some time before the start of i. Since each activity requires exactly one candidate, any flow of resource $c \in K^C$ between two activities will be either 0 or 1. By defining the variable F_{cij}^C as binary for all $c \in K^C$, we can use the variable as a resource flow and as a sequencing variable, respectively. As a sequencing variable, F_{cij}^C equals to 1 if and only if activity j is executed directly after activity i. We additionally introduce sequencing variables X_{cij}^C for formulating a valid inequality. With this notation the model reads as depicted in Figure 1.

The objective (1) is to minimize the duration D of the assessment center; D must be larger than or equal to the largest completion time of an activity of the project. Constraints (3)–(5) enforce a sequence of the activities for the sets of candidates, assessors, and actors, respectively. Constraint (6) links the flow variables to the sequencing variables for all $k \in K^O$. Constraints (7a)–(10) are the resource-flow conservation constraints. For any activities i and j, constraint (11) ensures that either i precedes j, or j precedes i, or i and j are processed in parallel. Constraint (12) ensures that all lunch breaks are executed within the prescribed time window. Constraint (13) implies that to each activity the

Fig. 1. Model formulation

$$
\begin{cases}
\text{Min. } D & (1)\\
\text{s.t. } D \geq S_i + p_i \qquad (i \in I) & (2)\\
S_j - S_i \geq -M + (p_i - \delta_i^A + M) F_{cij}^C \qquad \left(c \in K^C;\ i,j \in I_c\right) & (3)\\
S_j - S_i \geq -M + (p_i - \delta_j^C + M) X_{k^A ij}^O \qquad (i,j \in I_{k^A}) & (4)\\
S_j - S_i \geq -M + (p_i - (\delta_i^A - \delta_i^R) - \delta_j^C + M) X_{k^R ij}^O \qquad (i,j \in I_{k^R}) & (5)\\
F_{kij}^O \leq \min(r_{ik}, r_{jk}) X_{kij}^O \qquad \left(k \in K^O;\ i \in I_k \cup \{0\};\ j \in I_k \cup \{n+1\}\right) & (6)\\
\sum_{j \in I_c \cup \{0,n+1\}} F_{cij}^C = r_{ic} \qquad \left(c \in K^C;\ i \in I_c \cup \{0,n+1\}\right) & (7a)\\
\sum_{j \in I_k \cup \{0,n+1\}} F_{kij}^O = r_{ik} \qquad \left(k \in K^O;\ i \in I_k \cup \{0,n+1\}\right) & (7b)\\
\sum_{i \in I_c \cup \{0,n+1\}} F_{cij}^C = r_{jc} \qquad \left(c \in K^C;\ j \in I_c \cup \{0,n+1\}\right) & (8a)\\
\sum_{i \in I_k \cup \{0,n+1\}} F_{kij}^O = r_{jk} \qquad \left(k \in K^O;\ j \in I_k \cup \{0,n+1\}\right) & (8b)\\
F_{c,n+1,0}^C = 1 \qquad \left(c \in K^C\right) & (9)\\
F_{k,n+1,0}^O = r_{0k} \qquad \left(k \in K^O\right) & (10)\\
X_{kij}^O + X_{kji}^O \leq 1 \qquad \left(k \in K^O;\ i,j \in I_k \cup \{0,n+1\} :\ i < j\right) & (11)\\
ES^L \leq S_i \leq LS^L \qquad \left(i \in I^L\right) & (12)\\
\sum_{a \in A} Z_{ia} = r_{ik^A} \qquad (i \in I_{k^A}) & (13)\\
X_{k^A ij}^O + X_{k^A ji}^O \geq Z_{ia} + Z_{ja} - 1 \qquad (a \in A;\ i,j \in I_{k^A} \cup \{0,n+1\} :\ i < j) & (14)\\
Y_{ca} = 0 \qquad ((c,a) \in N) & (15)\\
\sum_{i \in I_c \setminus I_L} \frac{Z_{ia}}{|I_c \setminus I_L|} \leq Y_{ca} \leq \sum_{i \in I_c \setminus I_L} Z_{ia} \qquad \left(c \in K^C;\ a \in A\right) & (16)\\
\lfloor \tfrac{1}{2} |A| \rfloor \leq \sum_{a \in A} Y_{ca} \leq \lceil \tfrac{1}{2} |A| \rceil + 1 \qquad \left(c \in K^C\right) & (17)\\
S_i \leq S_j \qquad \left(i,j \in I^S : i < j\right) & (18)\\
S_j \geq \sum_{i \in I_c:\ i<j} (p_i - \delta_i^A) X_{cij}^C + \sum_{i \in I_c:\ i>j} (p_i - \delta_i^A)(1 - X_{cji}^C) \qquad \left(c \in K^C;\ j \in I_c\right) & (19)\\
X_{cij}^C \geq F_{cij}^C \qquad \left(c \in K^C;\ i,j \in I_c \cup \{0,n+1\} :\ i < j\right) & (20)\\
X_{cij}^C \geq X_{kij}^O \qquad \left(c \in K^C;\ k \in K^O;\ i,j \in I_c \cap I_k :\ i < j\right) & (21)\\
X_{cij}^C + X_{cjl}^C + (1 - X_{cil}^C) \leq 2 \qquad \left(c \in K^C;\ i,j,l \in I_c :\ i < j < l\right) & (22)\\
D \geq \sum_{i \in I_1} (p_i - \delta_i^A) & (23)\\
D \geq \left\lceil \sum_{i \in I : r_{ik^A} = 1} \frac{(p_i - \delta_i^C)}{|A|} + \sum_{i \in I : r_{ik^A} = 2} \frac{(p_i - \delta_i^C)}{\lfloor |A|/2 \rfloor} \right\rceil & (24)\\
D \in \mathbb{Z}_{\geq 0} & (25)\\
S_i \geq 0 \qquad (i \in I \cup \{0,n+1\}) & (26)\\
F_{cij}^C \in \{0,1\} \qquad \left(c \in K^C;\ i,j \in I_c \cup \{0,n+1\}\right) & (27)\\
F_{kij}^O \geq 0 \qquad \left(k \in K^O;\ i,j \in I_k \cup \{0,n+1\}\right) & (28)\\
X_{cij}^C \in \{0,1\} \qquad \left(c \in K^C;\ i,j \in I_c \cup \{0,n+1\} :\ i < j\right) & (29)\\
X_{kij}^O \in \{0,1\} \qquad \left(k \in K^O;\ i,j \in I_k \cup \{0,n+1\}\right) & (30)\\
Y_{ca}, Z_{ia} \in \{0,1\} \qquad \left(c \in K^C;\ a \in A;\ i \in I_{k^A} \cup \{0,n+1\}\right) & (31)
\end{cases}
$$

required number of assessors is assigned. Constraint (14) means that if an assessor a is assigned to both activities i and j, then there must be a sequence between the two activities. Constraint (15) models the no-go relationships. Constraint (16) determines whether an assessor a has been assigned to a candidate c at least once. Constraint (17) limits the number of different assessors that can be assigned to a candidate. Constraint (18) removes some symmetric solutions from the search space: for an arbitrary chosen exercise, we impose an arbitrary sequence of the corresponding activities; thereby we exclude the activities that belong to candidates for which a no-go relationship exists. Constraint (19) corresponds to the half-cuts proposed by Applegate and Cook (1991), i.e., the constraint represents a valid inequality for all activities of each candidate. Constraints (20)–(22) determine the values of the sequencing variables that are required by the valid inequality. Constraint (23) and (24) represent lower bounds; constraint (23) states that D must be greater than or equal to the total duration of all activities that require the same candidate $c = 1$, and constraint (24) states that D must be greater than or equal to the total duration of all activities (except lunch breaks) distributed among the assessors. Constraint (25) enforces that D must be integer; although all input variables are integer and therefore this constraint is redundant, we observed that this constraint improves the model performance.

3 Computational results

We implemented the model presented in Section 2 in AMPL, and used version 5.5 of the Gurobi Solver. All computations were performed on a workstation with 2 Intel Xeon CPU with 3.1GHz and 128GB RAM. We set the Gurobi Solver option `mipfocus=1`, and we limited the number of threads used for the computations to 8. Based on four real-life instances, we compare our model (TZ14) against the model presented in Grüter *et. al.* (2013) (GTZ13). For all instances and both models, a CPU time limit of 3600sec was prescribed. The characteristics of the instances and the results are shown in Table 1. For instances 1 and 3, the new model outperforms the (GTZ13) model.

Table 1. Results

Inst.	$\|C\|$	$\|A\|$	$\|R\|$	$\|N\|$	**TZ14**		**GTZ13**	
					D	**gap**	D	**gap**
1	7	10	2	0	87	8.0%	88	9.1%
2	11	11	3	0	121	8.3%	119	7.6%
3	9	11	3	1	96	6.3%	102	11.6%
4	6	9	3	0	83	3.6%	82	2.4%

4 Conclusion

This paper deals with an application of resource-constrained project scheduling in service operations management. We have devised an MILP formulation based on the resource-flow model of Koné *et. al.* (2011). Our computational results indicate that for two real-world instances this project-scheduling based MILP outperforms a specific MILP formulation.

In future research, further symmetry-breaking constraints, valid inequalities, and lower bounds should be developed in order to further improve the model performance.

References

Applegate, D., W. Cook, 1991, "A computational study of the job-shop scheduling problem.", *ORSA J COMPUT*, Vol. 3, pp. 149-156.

Grüter J., N. Trautmann and A. Zimmermann, 2013, "An MBLP model for scheduling assessment centers.", *in: NN (eds.), Operations Research Proceedings 2013, Springer, Berlin*, to appear.

Koné O., C. Artigues, P. Lopez and M. Mongeau, 2011, "Event-based MILP models for resource-constrained project scheduling problems.", *COMPUT OPER RES*, Vol. 38, pp. 3-13.

Graph Coloring and Job Scheduling: from Models to Powerful Tabu Search Solution Methods

Nicolas Zufferey[1]

GSEM - University of Geneva, Switzerland, n.zufferey@unige.ch

Keywords: graph coloring, job scheduling, tabu search metaheuristic.

1 Graph coloring as a modeling tool for job scheduling

Consider on the one hand the scheduling problem $SCHED$ where n jobs have to be performed while minimizing the makespan. The following assumptions are made: (A1) the duration of each job is one time period; (A2) all the time periods have the same duration; (A3) the production system consists in $m \geq n$ identical parallel machines; (A4) during each time period, each machine can perform a maximum of one job; (A5) some pairs of jobs are *incompatible* and their processing can thus not overlap in time. On the other hand, consider the famous *graph coloring problem* COL (see (Malaguti and Toth 2010) for a survey). Let $G = (V, E)$ be a graph with vertex set V (with $|V| = n$) and edge set E. COL consists in assigning a color (i.e. an integer between 1 and n) to each vertex such that two adjacent vertices have different colors, while minimizing the number of different colors used. COL can model $SCHED$ as follows: a vertex x represents a job x, an edge $[x, y]$ indicates that jobs x and y are incompatible, and a color t corresponds to a time period t. Therefore, the production terminology (e.g., job, time period, $SCHED$) and the graph coloring terminology (e.g., vertex, color, COL) can be indifferently used. Three variations of $SCHED$ are also considered in this work from the graph coloring perspective: (1) maximize the number of performed jobs if the makespan is upper bounded; (2) consider precedence constraints between some pairs of jobs; (3) allow to perform incompatible jobs during the same time period if incompatibility costs are paid (i.e. relax assumption (A5)).

This paper, which relies on (Zufferey 2013), does not aim to redo the literature review and the experiments for each of the considered production problems (detailed in the next sections), but rather to discuss these problems, with a unified view, under the successful light of the graph coloring model and the tabu search (TS) methodology. For this reason, only a key reference is given for each problem, which provide detailed information on the pseudo-codes of the algorithms, the NP-hard aspects, the literature review, and the presentation of the results. The common point of these key references is that for each considered problem, TS proved to have competitive results according to various criteria (e.g., quality of the obtained solutions, speed, ease of adaptation, ability to take advantage of problem structure).

TS is a powerful local search method where a *neighbor* solution s' is generated at each iteration from a *current* solution s by slightly modifying the latter. Such a modification is called a *move*. In order to avoid *cycling* (i.e. returning to a previously visited solution in the recent past), when a move is performed, its reverse is *tabu* (forbidden) during *tab* (parameter) iterations. At each iteration, TS usually performs the best possible non tabu move. The method can be stopped when a time limit is reached. The reader is referred to (Gendreau and Potvin 2010) for more information on metaheuristics (including TS).

2 Minimizing the makespan

The most efficient metaheuristics for $SCHED$ generally work with a fixed number k of periods, which leads to the k-$SCHED$ problem, consisting in assigning a time period in

$\{1, \ldots, k\}$ to each job, so as not to generate conflicts (a *conflict* occurs if two incompatible jobs are performed at the same time). If a feasible solution is found (also known as a k-scheduling without conflict), the process is restarted with $k-1$ periods, and so on until the used method cannot find a feasible solution (i.e. without conflict). $SCHED$ is therefore tackled by solving a series of k-$SCHED$ problems, beginning for example with $k = n$.
A well-known TS for k-$SCHED$ is *TabuCol* as proposed in (Galinier and Hao 1999). A solution is modeled by $s = (C_1, C_2, \ldots, C_k)$, where each C_t contains the set of jobs performed during period t. Given that a solution s is a partition of all the jobs into k sets (also called *classes*), it may contain conflicts. The objective function f to minimize is therefore the number of conflicts, and the algorithm can stop if a solution s with $f(s) = 0$ is found. A move $(j, C_t, C_{t'})$ consists in assigning a time period t' to a conflicting job j which is currently in C_t. It is then tabu (forbidden) to reallocate period t to job j during *tab* (parameter) iterations, where *tab* depends on the number of conflicts in the current solution. The efficiency of *TabuCol* mainly relies on two ingredients: (1) a powerful incremental computation (it is quick to evaluate the quality of a move $(j, C_t, C_{t'})$ as follows: number of jobs in $C_{t'}$ which are incompatible with j minus number of jobs in C_t which are incompatible with j); (2) at each iteration, the focus is only put on conflicting jobs (the size of the explored neighborhood is therefore drastically reduced).
Another powerful approach to tackle k-$SCHED$ consists in working with solutions modeled with $s = (C_1, C_2, \ldots, C_k; O)$, which is a partition of the n jobs into $k+1$ classes. Each class C_t is conflict-free and contains the jobs performed at period t, and O is the set of unperformed jobs and allows conflicts. The function to minimize is $|O|$: if it reaches zero, a feasible k-scheduling is found. Therefore, there are mainly two efficient solution spaces for k-$SCHED$: (1) the space $\mathcal{E}^{(c)}$ of k-complete schedules whose conflicts have to be minimized; (2) the space $\mathcal{E}^{(p)}$ of k-partial schedules without conflicts, where the number of unperformed jobs has to be minimized. Two algorithms A_1 and A_2 respectively associated with $\mathcal{E}^{(c)}$ and $\mathcal{E}^{(p)}$ can be compared by computing the smallest k for which A_1 finds a k-schedule without conflict, and the smallest k for which A_2 finds a k-schedule without unperformed jobs.

3 Maximizing the number of completed jobs

Consider the k-$SCHED$ problem with assumptions (A1) to (A5). The objective is to maximize the number of performed jobs with a makespan limited to k time periods. Even if the search spaces $\mathcal{E}^{(c)}$ and $\mathcal{E}^{(p)}$ are associated with the same k-$SCHED$ (and therefore $SCHED$) problem, an important distinction can be made from a production standpoint: performing n jobs while minimizing the makespan is very different from minimizing the number of completed jobs for a given upper bound on the makespan. The latter problem is particularly relevant when the number of incompatibilities between jobs is large, so that it is not possible to perform the n jobs. In such a case, the appropriate selection of jobs to perform, as well as the production sequences, are important issues, particularly in the situation where each production action is triggered only by an order from a client. This leads to the literature on the *order acceptance and scheduling problems* (Slotnick 2011).
A powerful TS for k-$SCHED$ is *PartialCol* proposed in (Bloechliger and Zufferey 2008), which works in $\mathcal{E}^{(p)}$. A move (j, C_t) consists in two steps: (1) assign a time period t to a job $j \in O$ (i.e. put job j in class C_t); (2) put in O all the jobs of C_t which are incompatible with j. It is quick to evaluate the value of a move (j, C_t) as the number of jobs in C_t which are incompatible with j (as for *TabuCol*, an efficient incremental computation is used). When a move (j, C_t) is performed, it is tabu for *tab* iterations to reinsert in C_t a job which was just removed in the above step (2) (in order to avoid removing from C_t the job j which has just entered it). The value of *tab* depends on the variability Δf of the objective function $f = |O|$ in the last *cycle* of iterations (a cycle contains several hundred iterations).

As it was the case for *TabuCol*, *PartialCol* has then often been used as an intensification procedure for *k-SCHED* in the best coloring methods.
In (Zufferey *et. al.* 2008) is considered an extension of this problem with different machines (relaxation of assumption (A3)), each job has a specific duration (relaxation of (A1)), and time window constraints are imposed. For each job j are known: the set of machines M_j able to process j, its processing time p_{ij} if it is performed on machine i, and its time window $[r_{ij}, d_{ij}]$ associated with each machine i. The objective is to schedule a maximum number of jobs within their time windows. A solution can be expressed as $s = (C_1, \ldots, C_m; O)$, where C_i contains all the jobs performed on machine i, and O contains the bumped jobs. A conflict occurs between two jobs if there is an intersection between their processing intervals. As in *PartialCol*, a move basically consists in firstly introducing a job $j \in O$ into a class C_i, and then removing from C_i the conflicting jobs (such jobs are introduced in O).

4 Precedence constraints

Consider *SCHED* with precedence constraints. For each job $j \in V$, the set $P(j) \subset V$ of its *immediate predecessors* is known. If $j' \in P(j)$, it indicates that job j' must be completed before job j begins. The objective is to assign a time period t to each job j while minimizing the makespan and satisfying the incompatibility and precedence constraints. Let *SCHED-PREC* denote this problem, whose version with a number of periods fixed to k is expressed as *k-SCHED-PREC*. Similarly to *SCHED*, *SCHED-PREC* can be tackled by solving a series of *k-SCHED-PREC* problems, beginning for example with $k = n$.
SCHED-PREC can be modeled with the *mixed graph* coloring model, denoted *MCOL*. A mixed graph $G = (V, E, A)$ is a graph with a set of vertices V, a set of edges E, and a set of arcs A. By definition, an edge is undirected, and an arc is a directed edge. An edge linking vertices x and y is denoted $[x, y]$, whereas an arc from x to y is denoted (x, y). *MCOL* consists in giving a color to each vertex in order to minimize the number of different colors used, while satisfying incompatibility (if edge $[x, y]$ exists in E, the vertices x and y must get two different colors) and precedence constraints (if arc (x, y) exists in A, the color of x must be strictly smaller than the color of y). One can easily see the equivalence between *MCOL* and *SCHED-PREC*: a vertex represents a job j, an edge $[j, j']$ indicates that the jobs j and j' are incompatible, an arc (j'', j) indicates that the job j can only be initiated when the job j'' is completed, and a color t represents a time period t.
The efficient TS for *k-SCHED-PREC* proposed in (Meuwly *et. al.* 2010) is an extension of *PartialCol*. Let t_j denote the period assigned to job j. There is a conflict between jobs j and j' if (1) or (2) is satisfied: (1) $[j, j'] \in E$ and $t_j = t_{j'}$ (incompatibility constraint violation); (2) $(j, j') \in A$ and $t_j \geq t_{j'}$ (precedence constraint violation). A solution s is modeled by $s = \{C_1, \ldots, C_k; O\}$, where C_t is the set of jobs performed during period t (without any conflict). The function $f = |O|$ must be minimized (all the jobs without a period are in O). The neighborhood structure is the same as for *PartialCol*: assign a color t to an uncolored vertex j, and then remove the color of the vertices of C_t in conflict with j. When such a move is performed, it is then tabu to remove j from C_t during *tab* iterations, which is a number depending on the number of conflicts in the current solution.

5 Incompatibility costs

Consider the problem *k-SCHED-INC* derived from *k-SCHED* by relaxing assumption (A5). If two incompatible jobs j and j' are performed during period t, an *incompatibility cost* $c(j, j') = c(j', j)$ is incurred. If j is performed during t, an *assignment cost* $a(j, t)$ must also be paid. The incompatibility cost $c(j, j')$ represents that the same resource must perform jobs j and j', which implies the mobilization of additional resources to perform j

and j' simultaneously. The assignment cost $a(j,t)$ represents for example the cost of the resources required if one wants to perform job j during period t. The objective consists in assigning a time period to each job in order to minimize the incompatibility and assignment costs. A solution s is modeled by $s = (C_1, \ldots, C_k)$, and an incompatibility graph can be built: an edge $[j, j']$ between two vertices j and j' indicates that if the same color is given to vertices j and j', then the cost $c(j,j') > 0$ is encountered. This model is able to also reasonably account for precedence constraints and forbidden periods for some jobs.
In the efficient TS for k-*SCHED-INC* proposed in (Zufferey *et. al.* 2012), a move $(j, C_t, C_{t'})$ consists in giving period t' instead of period t to job j. However, to avoid evaluating all the possible moves at each iteration, only the most promising moves are examined, that are the ones which contribute the most to the objective function. When a move $(j, C_t, C_{t'})$ is performed, it is tabu to assign period t to job j during *tab* iterations. A refined management of the tabu status is used, based on the following idea: if the diversity of the visited solutions within the last cycle of iterations is below a specific threshold δ (parameter), the value of *tab* is increased for a couple of iterations in order to favor the exploration of new zones of the solution space (diversification phase). On the contrary, if the diversity is greater than δ, the value of *tab* is reduced for a couple of iterations in order to favor a deeper exploration of the search space zone in which the current solution lies (intensification phase).

6 Conclusion

In this work, various NP-hard scheduling problems on parallel machines have been examined under the light of graph coloring models. It was showed that TS can be easily adapted to specific production problems. This success mainly relies on four aspects: (1) an efficient representation of a solution, allowing incremental computation; (2) the use of an auxiliary objective function different from the given objective function associated with the problem; (3) the use of aggressive moves (e.g. which eliminates at least one conflict at each iteration, even it creates other conflicts somewhere else in the solution); (4) an efficient management of the tabu durations. This paper contributes to build bridges between the graph coloring and the production scheduling communities.

References

Bloechliger I., N. Zufferey, 2008, "A graph coloring heuristic using partial solutions and a reactive tabu scheme", *Computers & Operations Research*, Vol. 35, pp. 960-975.

Galinier P., J.-K. Hao, 1999, "Hybrid evolutionary algorithms for graph coloring", *Journal of Combinatorial Optimization*, Vol. 3 (4), pp. 379-397.

Gendreau M., J.-Y. Potvin, 2010, "Handbook of Metaheuristics", *Springer*.

Malaguti E., P. Toth, 2010, "A survey on vertex coloring problems", *International Transactions in Operational Research*, Vol. 17 (1), pp. 1-34.

Meuwly F.-X., B. Ries and N. Zufferey, 2010, "Solution methods for a scheduling problem with incompatibility and precedence constraints", *Algorithmic Operations Research*, Vol. 5 (2), pp. 75-85.

S.A. Slotnick, 2011, "Order acceptance and scheduling: A taxonomy and review", *European Journal of Operational Research*, Vol. 212 (1), pp. 1-11.

Zufferey N., 2013, Chapter "Metaheuristics for Production Scheduling", *Models and Methods in Graph Coloration for Various Production Problems*, B. Jarboui, P. Siarry and J. Teghem (Eds.), Hermès – Lavoisier, France.

Zufferey N., P. Amstutz and P. Giaccari, 2008, "Graph colouring approaches for a satellite range scheduling problem", *Journal of Scheduling*, Vol. 11 (4), pp. 263-277.

Zufferey N., O. Labarthe and D. Schindl, 2012, "Heuristics for a project management problem with incompatibility and assignment costs", *Computational Optimization and Applications*, Vol. 51, pp. 1231-1252.