Metaheuristic Approaches for Resource-Constrained Project Scheduling with Flexible Resource Profiles

Martin Tritschler

TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Wirtschaftswissenschaften
Lehrstuhl für Operations Management

Metaheuristic Approaches for Resource-Constrained Project Scheduling with Flexible Resource Profiles

Dipl.-Wirtsch.-Ing. Martin Tritschler

Vollständiger Abdruck der von der Fakultät für Wirtschaftswissenschaften der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Wirtschaftswissenschaften
(Dr. rer. pol.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Martin Grunow
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. Rainer Kolisch
	2.	Prof. John J. Kanet, PhD University of Dayton, USA

Die Dissertation wurde am 30.9.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Wirtschaftswissenschaften am 15.6.2016 angenommen.

For Ka Yan

Acknowledgments

First of all, I would like to thank Prof. Dr. Rainer Kolisch, who gave me the opportunity to conduct research in the fascinating field of project scheduling and greatly supported me during the whole process of creating this dissertation. I also have to thank Dr. Anulark Naber for providing guidance and feedback. Furthermore, I would like to thank my second examiner Prof. John J. Kanet as well as the chairman of the examination committee Prof. Dr. Martin Grunow. Special thanks go to my colleagues at the Chair of Operations Management for establishing a great work environment with valuable discussions: Claus Brech, Alexander Döge, Dr. Jia-Yan Du, Dr. Thomas Fliedner, Martin Fink, Dr. Markus Frey, Dr. Daniel Gartner, Dr. Ferdinand Kiermaier, Christian Ruf, and Dr. Sebastian Schiffels. Moreover, I am grateful to have good friends who encouraged me to take this challenge. Finally, I would like to express my deepest gratitude to my girlfriend Ka Yan Wu and of course my parents Young-Ae Kang-Tritschler and Jürgen Tritschler—your constant support is truly appreciated.

Munich, September 2015

Abstract

English: The resource-constrained project scheduling problem with flexible resource profiles (FRCPSP) is an optimization problem from the field of project management. It consists of scheduling activities in order to minimize the project makespan. For each activity, we have to determine its duration and a flexible resource profile that fulfills the activity's resource requirements. This flexible resource profile specifies the quantity of resources allocated to the activity in each time period and is not limited to a rectangular shape. In this work, metaheuristic approaches for the FRCPSP are developed and evaluated. First, a hybrid metaheuristic is proposed. It contains a problem-adapted schedule generation scheme, which is embedded into a genetic algorithm. The best solutions are further improved in a variable neighborhood search based on an analysis of resource flows. Second, two model-based metaheuristics are introduced. They implement a mathematical decomposition of the FRCPSP into two parts: a master problem to schedule activities and a subproblem to determine resource profiles. Both model-based metaheuristics use disjunctive arcs to represent solutions to the master problem and apply linear programming techniques to determine optimal resource profiles. The performance of all proposed methods is demonstrated in a computational study with benchmark methods from literature.

German: Das ressourcenbeschränkte Projektplanungsproblem mit flexiblen Ressourcenprofilen (FRCPSP) ist ein Optimierungsproblem aus dem Bereich des Projektmanagements. Hierbei müssen Aktivitäten eingeplant werden mit dem Ziel, die Projektdauer zu minimieren. Für jede Aktivität müssen die Dauer sowie ein Ressourcenprofil, das den Ressourcenbedarf der Aktivität abdeckt, bestimmt werden. Dieses flexible Ressourcenprofil definiert die allokierte Menge an Ressourcen pro Zeitperiode und ist nicht auf eine rechtwinklige Form beschränkt. Im Rahmen dieser Arbeit werden metaheuristische Lösungsverfahren für das FRCPSP entwickelt und evaluiert. Erstens wird eine hybride Metaheuristik vorgestellt. Sie beinhaltet einen problemangepassten Schedule Generation Scheme, der in einen Genetischen Algorithmus eingebettet ist. Die besten gefundenen Lösungen werden mit einer Variable Neighborhood Search basierend auf einer Analyse von Ressourcenflüssen weiter verbessert. Zweitens werden zwei modellbasierte Metaheuristiken vorgestellt. Sie benutzen eine mathematische Dekomposition des FRCPSP in zwei Teile: ein Masterproblem zur Bestimmung des Projektablaufplans und ein Subproblem zur Bestimmung der Ressourcenprofile. Beide modellbasierten Metaheuristiken benutzen disjunktive Vorgangsbeziehungen, um Lösungen des Masterproblems zu repräsentieren, und setzen Techniken der linearen Programmierung zur Bestimmung optimaler Ressourcenprofile ein. Die Leistungsfähigkeit der vorgestellten Verfahren wird in einer numerischen Studie mit bestehenden Benchmark-Methoden demonstriert.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's business world many value adding activities are organized in projects. In general, a project is "a temporary endeavor undertaken to create a unique product, service, or result" (Project Management Institute, 2013, p. 3). This dissertation deals with a hard optimization problem from the field of project management: the resource-constrained project scheduling problem with flexible resource profiles (FRCPSP). The FRCPSP is part of the project planning phase in the overall project life cycle, after the project has been defined but prior to its execution (Lewis, 2006). In the FRCPSP, a single project with a given set of non-preemptive activities has to be scheduled in order to minimize the project duration or makespan. A planning horizon of discrete time periods is considered. The project is constrained by limited resource availability and technological finish-to-start precedence relations with zero time-lags between activities. These precedence relations require that an activity can only start after all of its predecessors have been completed.

The FRCPSP is a generalization of the well-known resource-constrained project scheduling problem (RCPSP), which is categorized as $PS|prec|C_{max}$ in the classification of Brucker et al. (1999). Whereas in the underlying RCPSP, the activity durations are given and the resource-usage of each activity is both known and constant for the entire activity duration, the FRCPSP reflects the common real-world case that only the total resource requirements of each activity are known in advance. The resource requirements are a measure of the effort required to complete the activity, usually expressed in units such as person-days (Demeulemeester and Herroelen, 2002). How to fulfill these resource requirements is part of the planning problem. Hence, besides scheduling the activities, also the activity durations and the continuous quantity of resources allocated to each activity per period have to be determined. As these allocated resource quantities may vary between periods, an activity's "resource profile", as denoted by Naber and Kolisch (2014b), becomes flexible and is not limited to a rectangular shape as in the RCPSP.

Consider for example a project with an activity that requires 12 person-days of human labor as resource in order to be completed. A project manager may determine an activity duration of 6 days and a rectangular-shaped resource profile of 2 employees on each day to fulfill the resource requirement. However, the project manager may as well decide upon an activity duration of 5 days and a flexible resource profile of 3 employees for the first 3 days and 1.5 employees, i.e., 1 employee working full-time and 1 employee working half-time on the activity, for the last 2 days.

The flexible resource profiles allow that the quantity of resources allocated to an activity can change while the activity is processed. Resources released by an activity may be allocated to other activities. The optimal makespan of an FRCPSP project with flexible resource profiles is always at least as good as in the case of solely rectangular-shaped resource profiles (Naber and Kolisch, 2014b). However, the makespan may as well be lower due to better resource utilization and resulting shorter activity durations. The FRCPSP extends the planning scope of classic project scheduling and is suitable for settings in which the assumption of constant resource usage is too restrictive.

Indeed, the FRCPSP has been applied to a wide range of real-world projects. These include pharmaceutical research (Kolisch et al., 2003), software development (Kuhlmann, 2003), construction (Schramme, 2014), and baggage handling at airports (Frey et al., 2014). Since Blazewicz et al. (1983) have shown that already the underlying RCPSP is $\mathcal{NP}$-hard, no methods are known that can solve the FRCPSP to optimality in polynomial time of the problem size. A study by Naber and Kolisch (2014b) shows that for FRCPSP problem instances with as little as 20 activities, a commercial mathematical solver using a mixed integer program (MIP) is unable to always find optimal solutions within a time limit of 2 hours. Motivated by the problem's high practical relevance and justified[1] by the problem complexity, research on metaheuristic approaches for the FRCPSP is of particular interest. Metaheuristics are "solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search" (Gendreau and Potvin, 2010, p. vii). Metaheuristics have been successfully applied to a wide range of hard optimization problems before.

1.1 Scientific Scope and Contributions

During the last decades, numerous metaheuristics have been proposed for the RCPSP, as described in Kolisch and Hartmann (2006). However, methods for the FRCPSP are still relatively scarce, despite the problem's huge potential. The goal of this dissertation is to provide new approaches for solving the FRCPSP to operations management and operations research. From the categorization of scientific research by Mitroff et al. (1974), contributions to modeling and model solving are made. In this work, I develop and evaluate novel metaheuristic approaches for the FRCPSP. The metaheuristics are grouped into two categories.

The idea of the approaches in the first category is to solve the problem by simultaneously scheduling activities and determining resource profiles. For this purpose, the schedule generation scheme (SGS) from the RCPSP is extended and adapted to the FRCPSP. The SGS is a constructive heuristic that, according to Kolisch (1996), generates a feasible schedule by iteratively extending a partial schedule. In the FRCPSP, an SGS also has to allocate resources in order to determine the flexible resource profiles and the variable activity durations. Two metaheuristics that apply SGSs are proposed.

The **hybrid metaheuristic (HM)** introduces the flexible resource profile parallel schedule generation scheme (FSGS). The design of the FSGS incorporates insights on FRCPSP solution characteristics by following the concepts of non-greedy resource allocation and delayed scheduling. Non-greedy resource

[1]Referring to the accepted standard for research on heuristic methods in operations management according to Bertrand and Fransoo (2002)

allocation is to not always allocate the maximum resource quantities to activities. Delayed scheduling is to not always start activities as early as possible. By applying both concepts, the FSGS can generate solutions that a standard SGS is unable to obtain. The FSGS is integrated into a hybrid metaheuristic framework that combines the advantages of two different search methods, a general approach that has shown to yield high performance on numerous hard optimization problems, according to Raidl et al. (2010). The FSGS is embedded into a genetic algorithm (GA) by using the activity list solution representation of Hartmann (1998) in combination with two additional parameters that control the non-greedy resource allocation and the delayed scheduling. The GA explores the vast search space before the search is intensified around promising solutions in a variable neighborhood search (VNS). The VNS locally improves the best solutions from the GA by transferring resource quantities between activities in order to reduce their durations. Specifically, resource quantities are transferred to activities that correspond to the critical path in the resource flow network of Artigues et al. (2003).

To the best of my knowledge, neither a variable neighborhood search nor any other methods that use resource transfers or resource flows have yet been proposed for the FRCPSP.

The **self-adaptive genetic algorithm (SGA)** is the second proposed metaheuristic that uses SGSs. It directly adapts the method of Hartmann (2002) to the FRCPSP. Two general types of SGSs exist, the serial and the parallel SGS. The SGA uses both types to generate a broad variety of schedules and it employs a self-adaptive parameter to select between the types.

All currently existing metaheuristics for the FRCPSP rely on SGSs. However, the heuristic generation of flexible resource profiles—the central distinguishing element of the FRCPSP—bears limitations. Whereas in the RCPSP, the set of schedules generated by the serial SGS always contains an optimal solution (Kolisch, 1996), this is not necessarily the case in the FRCPSP (Fündeling and Trautmann, 2010). The heuristic resource allocation mechanism may be unable to determine the right combination of resource profiles and activity durations. Determining optimal resource profiles for a given schedule with specified activity start periods and durations is already a problem by itself, specifically a continuous linear program. In fact, the FRCPSP consists of two interdependent parts: combinatorial scheduling and continuous resource allocation.

The metaheuristics in the second category exploit this structure by decomposing the problem and determining optimal resource profiles. Using Benders decomposition (Benders, 1962), a master problem (MP) is proposed for the combinatorial scheduling part and a subproblem (SP) is proposed for the continuous part of creating the resource profiles by allocating resources. In classic Benders decomposition, the MP and the SP are repeatedly solved to optimality in an iterative process. However, as the MP constitutes a scheduling problem with high symmetry, numerous iterations would be required. Furthermore, repeatedly solving the MP to optimality becomes time consuming after Benders cuts have been added. Hence, metaheuristics are considered as appropriate to obtain good approximate MP solutions in much faster time (Raidl, 2015). Two model-based metaheuristics are proposed: the **Benders genetic algorithm (BGA)** and the **reduced subproblem genetic algorithm (RGA)**. Other authors also use the equivalent term "matheuristics" for metaheuristics that integrate mathematical programming (Maniezzo et al., 2009).

Both model-based metaheuristics are GAs that gradually improve a population of encoded MP solutions and derive the fitness from the makespan and the resulting resource profiles. A new solution representation using disjunctive arcs (Shaffer et al., 1965) is proposed. The solution representations of existing methods for the FRCPSP cannot be applied in the decomposition, because information on the resource allocation is not yet available at the time the MP solutions are decoded.

Since the MP is not repeatedly solved to optimality anymore, the fitness evaluation becomes the by far most time consuming operation in the GAs. The BGA and the RGA use different approaches to reduce this time. The BGA follows the idea of Sirikum et al. (2007) to approximate the fitness from Benders optimality cuts without determining a resource profile for each encoded MP solution. The RGA, on the other hand, determines resource profiles for all MP solutions by solving the reduced subproblem (RSP) instead of the SP. The RSP is a compact reformulation of the SP. It exploits the structure of resource profiles, which consist of intervals with constant allocated resource quantities. In general, a feasible MP solution may result in an infeasible resource profile that violates the resource constraints. Hence, the BGA and the RGA integrate heuristic improvement operators to locally improve such solutions.

As far as I know, neither a problem decomposition nor a model-based metaheuristic have yet been proposed for the FRCPSP. The disjunctive arc solution representation is also new to the FRCPSP.

I evaluate the performance of the proposed metaheuristics in a computational study and compare them to benchmark methods from literature on problem instances with up to 200 activities. Using statistical analysis, I interpret the results and draw conclusions.

1.2 Overview

The remainder of this dissertation is organized as follows. Chapter 2 provides a description of the FRCPSP. Then the problem is distinguished from related project scheduling problems in Chapter 3. This chapter also provides a review of relevant literature on the topic. The HM is introduced in Chapter 4, which is based on Tritschler et al. (2015a). First, Section 4.1 illustrates important solution characteristics of the FRCPSP. Then Section 4.2 describes the HM and how the insights on the solution characteristics are integrated in the design of the HM's components. The SGA is outlined in Chapter 5, which is based on Tritschler et al. (2014a). In Chapter 6, the model-based metaheuristics are introduced. This chapter is based on Tritschler et al. (2015b). First, the mathematical models are provided in Section 6.1, then the BGA and the RGA are described in Section 6.2. In the computational study presented in Chapter 7, the proposed methods are evaluated and compared to benchmark methods. Chapter 7 is based on Tritschler et al. (2015a,b). The dissertation closes in Chapter 8 with a summary of the main findings, concluding remarks, and an outlook to future research.

Chapter 2

Problem Description

This chapter provides a problem description closely following the FRCPSP definition of Naber and Kolisch (2014b). A formal mathematical model is presented in Chapter 6. The notation is summarized in Appendix B.

A project in the FRCPSP contains a given set $\mathcal{V} = \{1, ..., n\}$ of n non-preemptive activities. The project additionally features the dummy source activity 0 for the project start and the dummy sink activity $n + 1$ for the project end. Using a planning horizon of discrete time periods $t \in \mathcal{T}$, we have to determine for each activity $i \in \mathcal{V}$ a **start period** $s_i \in \mathcal{T}$ and an integer **duration** d_i that leads to a **completion period** $c_i \in \mathcal{T}$ defined as:

$$c_i = s_i + d_i - 1 \tag{2.1}$$

In practice, activity durations are usually measured in discrete work periods of uniform duration, e.g., workdays or workweeks (Project Management Institute, 2013). In accordance with practice and the predominant methodology for the underlying RCPSP, the planning horizon is assumed to be a discrete-time grid. Each activity starts at the beginning of its start period and completes at the end of its completion period. In the remainder of this work, only the entire periods are mentioned in order to maintain simplicity. All activities are subject to finish-to-start **precedence relations** with zero time-lag from the given set $\mathcal{E}_c$. These precedence relations are denoted as conjunctive arcs in order to differentiate them from the additional disjunctive arcs that are introduced in

Chapter 6. A precedence relation $(i \rightarrow j) \in \mathcal{E}_c$ requires that activity j may only start after its predecessor i has been completed: $c_i < s_j$. Precedence relations are assumed to be acyclic. The common activity-on-node network representation for projects is employed. Activities are represented as nodes and precedence relations as arcs. Given that the dummy source activity starts and completes in period 0, the objective of the FRCPSP is to minimize the **makespan** C_{max}:

$$C_{max} = \max_{i \in \mathcal{V}}(c_i) \tag{2.2}$$

The dummy sink activity starts and completes in period $C_{max} + 1$. In addition, a project contains a given set $\mathcal{R}$ of resources. For each activity $i \in \mathcal{V}$, we have to determine a **resource profile** for each required resource $r \in \mathcal{R}_i$. This resource profile specifies the continuous quantity of allocated resources q_{irt} in each period t. Due to the nonpreemption requirement, q_{irt} has to be positive from s_i to c_i. A resource profile consists of one or multiple **blocks** of consecutive periods with a constant quantity of allocated resources. According to Naber and Kolisch (2014b), resource profiles have to adhere to constraints that reflect practical restrictions:

1. The total quantity of resource r allocated to activity i has to fulfill the activity's **resource requirement** w_{ir}: $\sum_{t=s_i}^{c_i} q_{irt} \geq w_{ir}$. Resource quantities are assumed to be linearly additive. Their total quantity may exceed the resource requirement in order to guarantee problem feasibility. Demeulemeester and Herroelen (2002) refer to the term "work content". As resources are not limited to human resources, the broader term "resource requirement" of Naber and Kolisch (2014b) is used.

2. The quantity of resource r allocated to activity i per period has to be within the range of the **lower resource usage bound** $\underline{q}_{ir}$ and the **upper resource usage bound** $\overline{q}_{ir}$. From a technological perspective, a certain minimum number of employees may be required to process an activity. Their number may also be bounded from above, as there is simply not enough space to accommodate them or the activity becomes impractical.

The constraint is also motivated from a project management perspective. Allocating too small resource quantities may extend the duration of an activity beyond a reasonable scale. Both usage bounds may as well be equal, resulting in a resource profile of rectangular shape.

3. The quantity of resource r allocated to activity i has to remain constant for at least a **minimum block length** (Fündeling, 2006) of l_{ir} consecutive periods. Hence, each block needs to have a duration of at least l_{ir} periods. In practice, the minimum block length prevents constant fluctuations in the number of employees assigned to an activity, as this may otherwise cause additional efforts for coordination, travel, and job orientation.

Due to the minimum block length, the decision on when to start blocks plays a central role in the FRCPSP. Consider that an ongoing activity starts a new block with a reduced quantity of allocated resources in period t. In order to directly allocate the released resources to a second parallel activity, the second activity also has to start a new block. However, this is only possible if the second activity's block up to period $t-1$ fulfills the minimum block length. Hence, it is important to synchronize the blocks of different activities in order to ensure a high resource utilization. Furthermore, it can happen that an activity's resource requirement is already fulfilled before its current block ends. If the block has to be continued to meet the minimum block length, the activity duration is extended and an overallocation of resources results.

Each resource r is renewable with a constant **resource availability** of b_r units in each period. Besides machines, tools, and equipment, especially human labor is considered as a renewable resource, according to Demeulemeester and Herroelen (2002): the number of employees available to a project on a single day is limited, but their availability is renewed on each day. All resources are regarded as continuously divisible. As mentioned in Józefowska et al. (2000), typical examples of continuously divisible resources include money and various power sources, for instance, electric, pneumatic, and hydraulic power. Naber and Kolisch (2014b) state that also human resources or machines may be considered as continuously divisible resources if an employee or machine can process multiple activities in parallel. In this case, a fractional resource quantity is allocated to each activity. For example, if 1 employee equally works on 2 activities in parallel for a period of 1 day, 0.5 man-days are allocated to each activity in that period (Naber and Kolisch, 2014b). Józefowska et al. (2000) argue that continuous divisibility can also be assumed if the number of available units of a discrete resource is very high. Similarly, in tactical workforce or resource planning, resources are usually considered as continuously divisible. Naber and Kolisch (2014b) categorize resources into three types:

1. A **principal resource** k is the main resource of an activity and its allocated quantity may define the quantities of other resources. An activity requires at most one principal resource, but a project may contain multiple activity-specific principal resources.

2. A **dependent resource** r of activity i is a resource whose allocated quantity q_{irt} depends on the allocated quantity q_{ikt} of the activity's principal resource k through a linear resource function with coefficient α_{ir} and constant β_{ir}:

   $$q_{irt} \geq \alpha_{ir} \cdot q_{ikt} + \beta_{ir} \tag{2.3}$$

   Dependent resources must be allocated concurrently with their principal resource from the start to the completion period of the activity without interruption. Again, an overallocation of resources is allowed to ensure problem feasibility. An activity may require multiple dependent resources.

3. An **independent resource** of an activity is a resource whose allocated quantity is independent from other resources. An activity may also require multiple independent resources.

Let us proceed with the previous example in which employees are the principal resource. Assume that an employee needs 1 unit of a certain tool as dependent resource in order to perform activities. If the employee is assigned in 1 period to 2 parallel activities 0.5 man-days each, the tool is also shared proportionally between both activities. A linear resource function is assumed, as the marginal requirements for tools and equipment are constant. In this setting, an independent resource may be a facility, such as a production hall, that is required regardless of the number of employees and tools.

To summarize, the FRCPSP is to determine a makespan-minimal solution f that specifies for each activity $i \in \mathcal{V}$ a start period s_i, a duration d_i, and a resource profile for each required resource $r \in \mathcal{R}_i$ that fulfills all constraints.

Additional parameters can be derived for each activity. The **lower bound of duration** $\underline{d}$ is defined as:

$$\underline{d}_i = \max_{r \in R_i} \Big(\max(\Big\lceil \frac{w_{ir}}{\overline{q}_{ir}} \Big\rceil, l_{ir}) \Big) \tag{2.4}$$

Similarly, the **upper bound of duration** $\overline{d}_i$ is calculated by using $\underline{q}_{ir}$ instead of $\overline{q}_{ir}$. Each activity's **earliest start period** $\underline{s}_i$ and its **earliest completion period** $\underline{c}_i$ are determined by precedence-based critical path calculations that use $\underline{d}_i$ as activity duration and ignore the resource constraints. As detailed in Naber and Kolisch (2014b), these calculations are strengthened by applying an extension of the resource-based method of Schrage (1970). It considers the minimum number of periods that is required to fulfill the resource requirements of all predecessors of each activity with respect to the resource constraints. The **latest start period** $\overline{s}_i$ and the **latest completion period** $\overline{c}_i$ are derived by backwards calculations from an upper bound of the makespan with $\underline{d}_i$ as activity duration. The resulting **time window** $\mathcal{T}_i$ of activity i consists of all periods from $\underline{s}_i$ to $\overline{c}_i$. Finally, T_{min} defines a **lower bound** of the makespan, obtained from the earliest completion periods:

$$T_{min} = \max_{i \in V}(\underline{c}_i) \tag{2.5}$$

The FRCPSP is closely related to other project scheduling problems. It contains the RCPSP as a special case featuring only rectangular-shaped resource profiles with equal lower and upper resource usage bounds and a fixed activity duration as well as solely independent resources. Hence, as noted in Kolisch et al. (2003), the FRCPSP is also $\mathcal{NP}$-hard. In the following literate review, the FRCPSP is differentiated from other project scheduling problems.

Chapter 3

Literature Review

The FRCPSP was initially studied by Kolisch et al. (2003), who propose the problem in the context of real-world pharmaceutical research projects. They provide an MIP formulation as well as a priority rule heuristic that uses a schedule generation scheme (SGS). According to Kolisch (1996), the SGS is a constructive heuristic that generates a feasible schedule by iteratively extending a partial schedule. Kolisch (1996) distinguishes the serial SGS, proposed by Kelley (1963), and the parallel SGS, originating from Kelley (1963) and Bedworth and Bailey (1982). In both methods, a priority rule is employed to determine the sequence in which activities are scheduled. The method of Kolisch et al. (2003) can use both SGS types. Their employed SGSs implement the greedy principles of scheduling activities as early as possible and allocating the largest possible resource quantities. An SGS that adheres to these principles is denoted as a **standard SGS** in the remainder of this dissertation.

The coming sections provide a review of literature on the FRCPSP. The FRCPSP with continuous resource quantities is covered in Section 3.1 and the variant with discrete resource quantities in Section 3.2. Related project scheduling problems are considered in Section 3.3.

3.1 Continuous Resources

For the FRCPSP with continuously divisible resource quantities, as presented in Chapter 2, four MIP models are proposed by Naber and Kolisch (2014b). The authors also apply a priority rule heuristic with a standard serial SGS to compute an upper bound of the makespan. Naber and Kolisch (2014a) consider the generalization to a continuous time axis. Schramme (2014) deals with a problem variant that features neither minimum block lengths nor dependent resources. Besides an MIP model, he provides a GA with a non-greedy serial SGS. The FSGS proposed in Chapter 4 of this dissertation differs from the SGS of Schramme (2014), as the FSGS specifically addresses the minimum block length and determines activity durations as a result of the resource allocation, whereas Schramme (2014) considers activity durations as input parameters. The SGS of Schramme (2014) schedules activities as early as possible for the objective of makespan minimization. For other non-regular objectives, activity starts may as well be delayed.

3.2 Discrete Resources

The FRCPSP with discrete resource quantities differs from its continuous counterpart. The works of Fündeling (2006), Fündeling and Trautmann (2010), and Baumann et al. (2015) deal with a problem variant in which resource quantities are discrete and the resource requirements have to be exactly fulfilled. Thus, the set of resource profiles is finite and the problem becomes purely combinatorial. Naber and Kolisch (2015) point out that the exact requirements for discrete resources in combination with the minimum block length can lead to longer activity durations. Due to the resulting higher optimal makespans, methods for this discrete problem variant cannot be directly compared to methods for the FRCPSP with continuous resources.

An MIP model is proposed by Baumann et al. (2015), who also consider a relaxation of the integrality requirements. Fündeling (2006) provides priority rule heuristics and proposes standard serial and parallel SGSs, as well as a parallel SGS that allocates resources to the maximum number of activities by first fulfilling their lower resource usage bounds. He also describes a branch-and-bound method. The work of Fündeling and Trautmann (2010) is based on the standard serial SGS of Fündeling (2006). Both works assume that all activities in a project require the same principal resource. For projects entirely limited to a single resource, Ranjbar and Kianfar (2010) employ a GA with a serial SGS that uses a priori generated resource profiles. Resource profiles are limited to rectangular, non-increasing, non-decreasing, and triangular shapes and are generated by an enumeration procedure. For multi-project scheduling limited to a single resource, Zhang and Sun (2011) briefly outline a priority rule heuristic that uses a parallel SGS. For a related problem with several additional constraints from a practical application in software development, Kuhlmann (2003) proposes multiple GAs. Meyer (2003) covers a problem application in pharmaceutical research and proposes a priority rule heuristic.

3.3 Related Problems

The underlying RCPSP has been broadly covered in literature. For literature reviews refer to Brucker et al. (1999), Hartmann and Briskorn (2010), and Węglarz et al. (2011). Heuristics and metaheuristics are compared in Hartmann and Kolisch (2000) and Kolisch and Hartmann (2006).

The concept of allowing multiple processing modes for activities was introduced by Elmaghraby (1977). In this context, Węglarz et al. (2011) distinguish two basic activity processing models: the processing time vs. resource amount model and the processing rate vs. resource amount model.

In the first model, activity processing time is a nonincreasing function of the quantity of allocated resources, which is assumed constant throughout the entire processing time (Węglarz et al., 2011). In this category, the multi-mode resource-constrained project scheduling problem (MRCPSP), introduced by Talbot (1982), is similar to the FRCPSP. Each activity can also be processed in multiple different modes. However, the MRCPSP only features a predetermined set of modes with rectangular-shaped resource profiles from which one mode has to be selected. As resources in the FRCPSP are continuously divisible, the problem at hand cannot be simply converted to the MRCPSP by exhaustively generating modes (Naber and Kolisch, 2014b). Metaheuristics for the MRCPSP are compared in van Peteghem and Vanhoucke (2014).

The FRCPSP also differs from the discrete time-resource tradeoff problem (DTRTP), first proposed by De Reyck et al. (1998). In the DTRTP, activity durations are a function of resource usage. As this resource usage is assumed constant and integer for the entire duration of an activity, resource profiles are limited to rectangular shapes. A branch-and-bound approach for the problem is provided by Demeulemeester et al. (2000), while metaheuristics are proposed by De Reyck et al. (1998) and Ranjbar and Kianfar (2007). Ranjbar et al. (2009) consider the DTRTP with multiple resources, which constitutes a special case of the MRCPSP. They solve the problem by using an SGS-based metaheuristic that selects modes from a set of feasible modes.

In the second activity processing model, the activity processing rate is an increasing function of the quantity of allocated resources (Węglarz et al., 2011). Specifically, the allocated resource quantities are continuous and may vary during the processing time. Early works in this category are, e.g., Węglarz (1981) and Leachman et al. (1990). For RCPSP variants with variable activity intensities, MIP models are proposed by Kis (2006), Haït and Baydoun (2012), and Bianco and Caramia (2013). A discrete–continuous project scheduling problem that contains one continuous and multiple discrete resources is analyzed by Józefowska et al. (2000).

According to Naber and Kolisch (2014b), concepts of the FRCPSP are also applied in aggregate project planning on the tactical level. In rough-cut capacity planning, considered by Hans (2001) and Baydoun et al. (2014), multiple single operations are grouped into aggregate activities with variable duration but given resource requirements. Similarities to the FRCPSP can also be found in the context of project crashing. By allocating additional resource quantities, Deckro and Hebert (1989) reduce the duration of activities.

Chapter 4

The Hybrid Metaheuristic

In this chapter, the hybrid metaheuristic (HM) for the FRCPSP is presented. Characteristics of optimal solutions are analyzed in Section 4.1 and the gained insights are then applied in the design of the HM in Section 4.2.

4.1 Solution Characteristics

Approaches beyond the greedy principles mentioned in Chapter 3 are required to generate optimal solutions for the FRCPSP. The examples provided in this section apply to both the standard serial and the standard parallel SGS. To maintain simplicity, only one resource is considered in the examples and the resource index is omitted from the notation.

4.1.1 Non-greedy Resource Allocation

The greedy principle of always allocating the maximum resource quantity does not necessarily lead to the minimum makespan, as noted by Fündeling and Trautmann (2010). It may prevent the start of other activities or increase their durations. An optimal solution may feature periods in which the resource allocation to an activity is reduced, whereas in other periods it is increased in order to utilize the available resources.

For the project illustrated in Figure 4.1, Figure 4.2a illustrates a solution resulting from greedy resource allocation. Activity 1 starts with its maximum

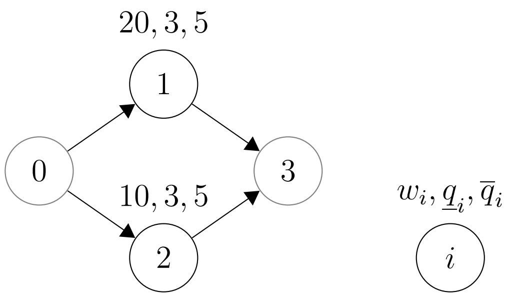

Figure 4.1: Project featuring a single resource with availability of $b = 7$ and a minimum block length of $l_i = 2$ for all activities

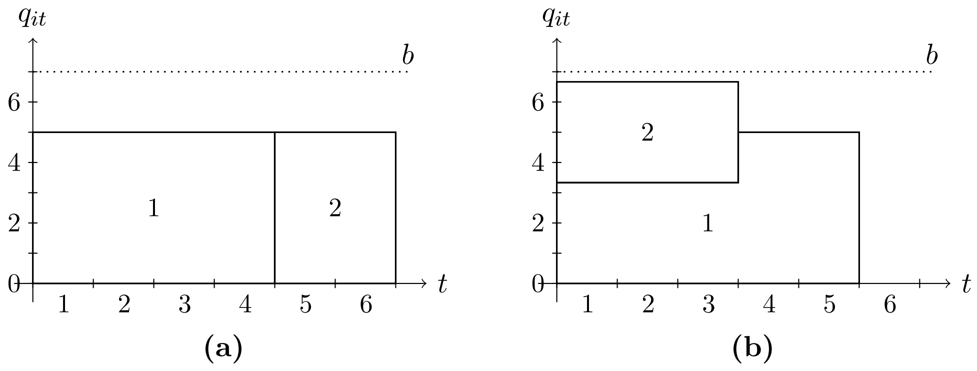

Figure 4.2: Examples of solutions for the project given in Figure 4.1

resource allocation of 5 units. Hence, activity 2 cannot start until period 5 and the resulting makespan is 6 periods. Figure 4.2b shows a solution resulting from a non-greedy resource allocation of only $3\frac{1}{3}$ resource units to both activities in periods 1 to 3. In periods 4 and 5, the resource quantity allocated to activity 1 is increased to the maximum of 5 units in order to utilize the available resources. The resulting makespan of 5 periods is optimal.

4.1.2 Delayed Scheduling

Scheduling activities as early as possible does not necessarily lead to the minimum makespan either. By delaying the start of an activity to a later period, the activity may be able to exploit a higher resource availability, which otherwise could not be utilized. This effect is caused by the minimum block length, as shown in Tritschler et al. (2014b) and Baumann et al. (2015).

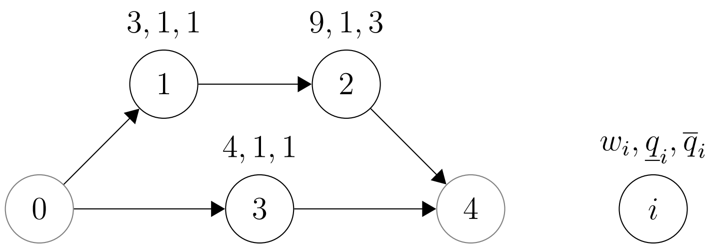

Figure 4.3: Project featuring a single resource with availability of $b = 3$ and a minimum block length of $l_i = 3$ for all activities

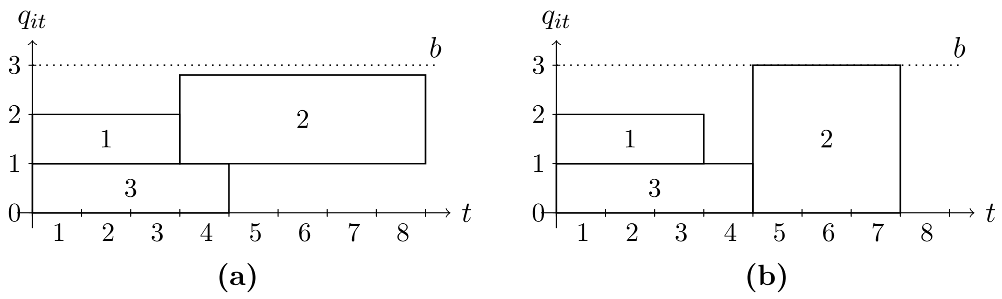

Figure 4.4: Examples of solutions for the project given in Figure 4.3

For the project given in Figure 4.3, Figure 4.4a depicts a solution in which all activities start as early as possible. Activity 2 starts in period 4, where due to the processing of activity 3 only 2 resource units are available. Respecting the minimum block length of 3 periods, 1.8 resource units are allocated for 5 periods to activity 2, resulting in a makespan of 8. Allocating additional resource quantities to activity 2 in period 7 would only further increase the makespan due to the minimum block length. By delaying the start of activity 2 to period 5, where 3 resource units are available, an optimal solution with the minimum makespan of 7 is obtained, as shown in Figure 4.4b.

4.2 Hybrid Metaheuristic

An adequate combination of different algorithmic approaches that exploits synergies between the methods is an important step towards achieving high performance on hard optimization problems (Raidl et al., 2010). The proposed HM combines three components. Section 4.2.1 introduces the flexible resource profile parallel schedule generation scheme (FSGS) and explains how it applies the concepts from Section 4.1. The FSGS is integrated into the genetic algorithm (GA) shown in Section 4.2.2. Whereas the GA explores the large search space, its best solutions are further locally improved with the variable neighborhood search (VNS) presented in Section 4.2.3.

The general assumption is that a feasible solution exists. The availability of each resource has to fulfill the lower bound of resource usage of each activity.

4.2.1 Flexible Resource Profile Parallel Schedule Generation Scheme

The FSGS implements non-greedy resource allocation and delayed scheduling. Just as depicted in the example from Figure 4.2b, the FSGS can both realize periods in which the resource allocation to an activity is limited and periods in which it is maximized again. The set of generated solutions contains only feasible solutions. However, due to the heuristic creation of resource profiles, it may not always contain the optimal solution. The coming sections first describe the FSGS input parameters, then explain the algorithm, and finally illustrate it with an example.

4.2.1.1 Input Parameters

The FSGS uses three input parameters of length n:

- **Activity list** λ: The sequence of activities for resource allocation is defined by the activity list λ of Hartmann (1998). λ is any precedence feasible permutation of the activities from set $\mathcal{V}$.

- **Resource allocation limit list** ρ: To facilitate non-greedy resource allocation, list $\rho = (\rho_1, \dots, \rho_n)$ contains for each activity $i \in \mathcal{V}$ an integer $\rho_i \in \{0, 1, \dots, \overline{\rho}_i\}$ that defines the limit $w_{ir}/(\underline{d}_i + \rho_i)$ for the allocated resource quantity q_{irt}. Instead of directly encoding the numerous continuous resource quantities for each resource and period, just one integer value is required per activity. The resulting activity duration is not necessarily $\underline{d}_i + \rho_i$ but is determined by the actual quantity of resources allocated per period in the FSGS. The upper bound $\overline{\rho}_i$ of ρ_i prevents too low limits and resulting excessive prolongations of activities:

$$\overline{\rho}_i = \min(\overline{d}_i - \underline{d}_i, \overline{c}_i - \underline{s}_i) \tag{4.1}$$

- **Start delay list** σ: To delay the start of activities, list $\sigma = (\sigma_1, \dots, \sigma_n)$ contains for each activity $i \in \mathcal{V}$ an integer $\sigma_i \in \{0, 1, \dots, \overline{\sigma}_i\}$ so that the start of activity i is delayed by σ_i periods. The delay is relative to the period in which the FSGS would otherwise schedule the activity. This is not necessarily the earliest precedence-feasible and resource-feasible period. The upper bound $\overline{\sigma}_i$ of σ_i is calculated from the earliest and latest start periods and, thus, allows free start periods within the activity's time window:

$$\overline{\sigma}_i = \overline{s}_i - \underline{s}_i \tag{4.2}$$

4.2.1.2 Algorithm

The FSGS extends the period-based approach of the parallel SGS to flexible resource profiles, minimum block lengths, and dependent resources. The FSGS increments time periods and considers in each iteration a period t. Resources are allocated to an activity in the order of principal, dependent, and independent resources. An activity is completed as soon as the resource requirements and the minimum block lengths of all of its required resources are met. The duration d_i of activity i results from the allocated resource quantities per period. For simplicity, let us assume for now that resource r of activity i is either principal or independent, i.e, $r \in \mathcal{R}_i^{pi}$.

Algorithm 1 Steps of the FSGS in period t

```
1. for $i \in \mathcal{V}_\mathcal{A}$ in sequence of $\lambda$, $r \in \mathcal{R}_i^{pi}$ do
        if $l_{ir(t-1)} \geq l_{ir}$ then
            $q_{irt} = \underline{q}_{ir}$
        else
            $q_{irt} = q_{ir(t-1)}$

2. for $i \in \mathcal{V}_\mathcal{A} \cup \mathcal{V}_\mathcal{E}$ in sequence of $\lambda$, $r \in \mathcal{R}_i^{pi} : \varphi_r > 0$ do
   (a)  if $i \in E$ and $\forall r' \in \mathcal{R}_i : \varphi_{r'} \geq \underline{q}_{ir'}$ then
            if $delay_i \geq \sigma_i$ then
                $q_{irt} = \min(\varphi_r, \frac{w_{ir}}{\underline{d}_i + \rho_i})$
            else
                $delay_i = delay_i + 1$
   (b)  if $i \in \mathcal{V}_\mathcal{A}$ and $l_{ir(t-1)} \geq l_{ir}$ then
            if $q_{irt} + \varphi_r \geq q_{ir(t-1)}$ then
                $q'_{irt} = \max(\underline{q}_{ir}, \min(q_{irt} + \varphi_r, \frac{\xi_{ir}}{l_{ir}}, \frac{w_{ir}}{\underline{d}_i + \rho_i}))$
                if $\lceil \frac{\xi_{ir}}{q'_{irt}} \rceil \geq l_{ir}$ and $(\lceil \frac{\xi_{ir}}{q'_{irt}} \rceil < \lceil \frac{\xi_{ir}}{q_{ir(t-1)}} \rceil$ or $q_{ir(t-1)} > \frac{w_{ir}}{\underline{d}_i + \rho_i})$
                    $q_{irt} = q'_{irt}$
                else
                    $q_{irt} = q_{ir(t-1)}$
            else
                $q_{irt} = q_{irt} + \varphi_r$
            if $\lceil \frac{\xi_{ir}}{q_{irt}} \rceil < 2 \cdot l_{ir}$ then
                $q_{irt} = \max(\underline{q}_{ir}, \xi_{ir} / \lceil \frac{\xi_{ir}}{q_{irt}} \rceil)$

3. for $i \in \mathcal{V}_\mathcal{A}$ in sequence of $\lambda$, $r \in \mathcal{R}_i^{pi} : \varphi_r > 0$ do
        Repeat Step 2b with $q'_{irt} = \max(\underline{q}_{ir}, \min(q_{irt} + \varphi_r, \frac{\xi_{ir}}{l_{ir}}, \overline{q}_{ir}))$
```

For each period t, the FSGS performs the steps given in Algorithm 1. λ provides the sequence of activities for resource allocation in all steps. ρ limits the resource allocation in Steps 2a and 2b. The start of activities is delayed by σ in Step 2a. The FSGS distinguishes two sets of activities. The activities of set $\mathcal{V}_{\mathcal{A}}$ are **active** in period t, that means they are currently processed. The activities of set $\mathcal{V}_{\mathcal{E}}$ are **eligible** for scheduling in period t, since all their predecessors have been completed up to $t-1$. The block up to period $t-1$ is denoted as the "current block", whereas the block starting in period t is the "new block". Algorithm 1 uses additional variables: φ_r is the current leftover availability of resource r, ξ_{ir} is the current remaining requirement of activity i for resource r, $delay_i$ is the number of periods by which the start of activity i has been delayed for, and l_{irt} is the length of the block of activity i and resource r until period t. All sets and variables are constantly updated but these update operations are not stated in the algorithm to maintain brevity.

Step 1 **Continue active activities to ensure nonpreemption:** For each resource r of activity $i \in \mathcal{V}_{\mathcal{A}}$ for which the minimum block length l_{ir} has been met, a quantity equal to the lower resource usage bound $q_{irt} = \underline{q}_{ir}$ is allocated. If l_{ir} has not been met, the current block is continued by allocating the quantity of the previous period $q_{irt} = q_{ir(t-1)}$.

Step 2 **Allocate remaining resources:** Resources are distributed among active and eligible activities in a non-greedy manner according to sequence λ. Two disjunct cases apply:

Step 2a **Start eligible activities:** Eligible activities are started based on the delays from σ. Activity $i \in \mathcal{V}_{\mathcal{E}}$ starts if the lower usage bound of each required resource is met and the start of the activity has already been delayed for at least σ_i periods. ρ_i limits the allocated resource quantity to $q_{irt} = \min(\varphi_r, w_{ir}/(\underline{d}_i + \rho_i))$.

Step 2b **Modify resource allocation of active activities:** This step only applies to resources required by active activities $i \in \mathcal{V}_{\mathcal{A}}$ for which the minimum block length has been met. For such resources, a quantity equal to the lower usage bound has already been allocated in Step 1. Now, additional resource quantities are allocated. Three different cases apply: (1) A new block has to be started due to insufficient resources. (2) The current block is continued. (3) A new block is started in order to change the resource allocation. The operations of each case are described below:

(1) If the current leftover quantity φ_r of resource r does not suffice to continue the current block, $q_{irt} = q_{irt} + \varphi_r$ is set and consequently a new block starts.

(2) If φ_r suffices to continue the current block, the algorithm checks whether to allocate the same quantity as in the previous period, i.e., $q_{irt} = q_{ir(t-1)}$, or to start a new block in case (3).

(3) A new block with $q'_{irt} = \max(\underline{q}_{ir}, \min(q_{irt} + \varphi_r, \xi_{ir}/l_{ir}, w_{ir}/(\underline{d}_i + \rho_i)))$ is only started by setting $q_{irt} = q'_{irt}$ if the following condition (I) and at least one out of conditions (II) or (III) apply: (I) The remaining resource requirement ξ_{ir} is sufficient to accommodate at least one minimum block length: $\lceil \xi_{ir}/q'_{irt} \rceil \geq l_{ir}$. (II) The new block resulting from an increased resource quantity q'_{irt} is shorter than the continued current block: $\lceil \xi_{ir}/q'_{irt} \rceil < \lceil \xi_{ir}/q_{ir(t-1)} \rceil$. (III) The allocated quantity in the current block is larger than the limit defined by ρ_i: $q_{ir(t-1)} > w_{ir}/(\underline{d}_i + \rho_i)$. Condition (III) can only apply if Step 3 has been performed at the beginning of the current block. Resulting from condition (III), the new block with a decreased resource quantity q'_{irt} again adheres to the limit defined by ρ_i. Conditions (I) and (II) are taken from Naber and Kolisch (2014b).

Next, if less than two minimum block lengths remain to complete the activity, i.e., $\lceil \xi_{ir}/q_{irt} \rceil < 2 \cdot l_{ri}$, the resource allocation is kept constant until the activity completes. Starting a new block in the next periods would otherwise increase the duration of the activity. To prevent an overallocation of resources, $q_{irt} = \max(\underline{q}_{ir}, \xi_{ir}/\lceil \xi_{ir}/q_{irt} \rceil)$ is set.

Step 3 **Utilize slacks:** After completing Steps 1 and 2, a feasible partial schedule up to period t has been generated. However, due to the resource allocation limits, the available resources may not be fully utilized in period t. These leftover resource quantities (slacks) are now exploited by temporarily exceeding the allocation limit. The resources are allocated to active activities in the sequence of λ if the minimum block length permits it. For this purpose, Step 2b is repeated with the modification $q'_{irt} = \max(\underline{q}_{ir}, \min(\varphi_r + q_{irt}, \xi_{ir}/l_{ir}, \overline{q}_{ir}))$, which allows allocating resources up to $\overline{q}_{ir}$.

Let us now consider dependent resources. The allocated quantity of dependent resource r is calculated from the allocated quantity q_{ikt} of principal resource k by $q_{irt} = \alpha \cdot q_{ikt} + \beta$. If φ_r is lower than q_{irt}, then $q_{irt} = \max(\underline{q}_{ir}, \varphi_r)$ is set and, as a consequence, the allocated quantity of the principal resource is updated to $q_{ikt} = (q_{irt} - \beta)/\alpha$. This modification of the principal resource in return requires to recalculate the quantities of other dependent resources. The process is repeated until q_{ikt} adheres to the availability and the resource usage bounds of all dependent resources.

4.2.1.3 Example

Consider the project illustrated in Figure 4.5. As the project only contains a single resource, the resource index is omitted in the notation. For $\lambda = (3, 1, 4, 2)$, $\rho = (1, 0, 0, 0)$, and $\sigma = (0, 0, 0, 0)$, the FSGS generates the solution f shown in Figure 4.6. In all periods not mentioned in the explanations below, Step 1 of Algorithm 1 applies.

In period 1, resources are first allocated to activity 3 due to its position in λ. As $\underline{d}_3 = \max(\lceil 12/4 \rceil, 2) = 3$ and $\rho_3 = 1$, a resource quantity of $\min(5, 12/(3+1)) = 3$ is allocated in Step 2a. The remaining 2 resource units are allocated to activity 1 in Step 2a. Activity 3 completes in period 4. In period 5, first 2 resource units are allocated to activity 1 in Step 1. This is sufficient for activity 1 to complete. Then, the remaining 3 resource units are allocated to start activity 4 in Step 2a. In period 6, activity 4 continues with the same allocated quantity as in period 5 due to Step 1. Hence, activity 2 can only start with an allocated quantity of 2 units from Step 2a. Activity 4 completes in period 9. In period 10, activity 2 requires additional $\xi_2 = 5$ resource units to complete. First, a resource quantity $q_{2,10} = 2$ that is equal to the lower usage bound is allocated in Step 1. As the length of the

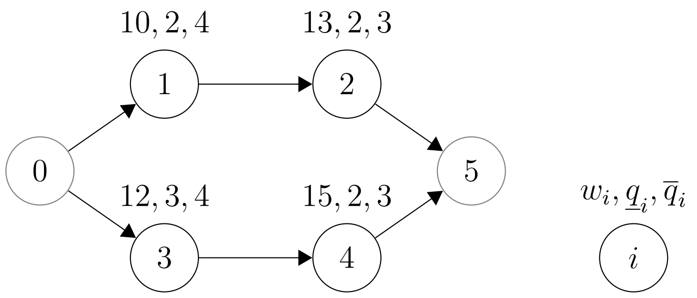

Figure 4.5: Project featuring a single resource with availability of $b = 5$ and a minimum block length of $l_i = 2$ for all activities

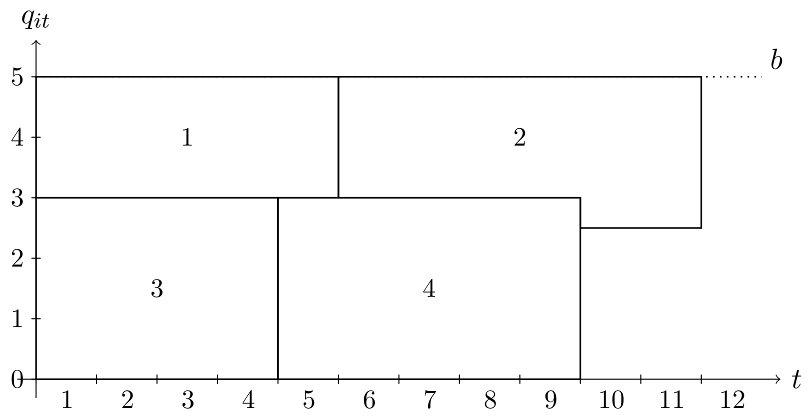

Figure 4.6: Solution f generated by the FSGS for the project from Figure 4.5

current block of 4 is greater than the minimum block length $l_2 = 2$, Step 2b increases the quantity to $q'_{2,10} = \max(\underline{q}_2, \min(q_{2,10} + \varphi, \xi_2/l_2, w_2/(\underline{d}_2 + \rho_2))) = \max(2, \min(2 + 3, 5/2, 15/(5 + 0)) = 2.5$. The new resulting block has a length of 2, which equals the minimum block length and is 1 period shorter than the current block if it were continued. As the new block does not overallocate resources, the resource quantity is not changed by operation $\max(2, 2.5/\lceil 2.5/2.5 \rceil) = 2.5$. The activity completes in the next period, resulting in a makespan of 11.

4.2.2 Genetic Algorithm

A GA is a population-based metaheuristic inspired by the principles of natural evolution. Since their first proposal by Holland (1975), GAs have been successfully applied to a wide range of hard optimization problems including the RCPSP, as reported in Reeves (2010) and Kolisch and Hartmann (2006).

Within the HM, the FSGS is embedded into a GA. This proposed GA uses the FSGS's three input parameter lists as solution representation. An encoded solution (λ, ρ, σ) is decoded by the FSGS into a feasible FRCPSP solution f. The resulting makespan is the fitness value.

The GA first creates an initial population of solutions as described in Section 4.2.2.1. In each generation, the elite solution with the lowest makespan is inserted unmodified into the next generation. Using the elite solution's makespan, the GA updates the upper bounds $\overline{s}_i$, $\overline{c}_i$, $\overline{\rho}_i$, and $\overline{\sigma}_i$. Thus, the bounds are tightened with each makespan improvement. Then the GA selects a set of solutions for the next generation by stochastic universal sampling as of Baker (1987). This fitness-proportionate selection procedure overcomes several limitations of simple roulette wheel selection (Reeves, 2010). The selected solutions are modified by the operators described in Section 4.2.2.2 and constitute the next generation. The whole process is repeated until the given maximum number of solutions F_{GA} has been generated. The GA returns as result the list $\vec{f}_{gen}$. It contains the encoded elite solution (λ, ρ, σ) of each GA generation as well as each corresponding decoded elite solution f. The list entries are unique regarding the encoded solutions. Hence, if two generations feature the identical encoded elite solution, it is only stored once in $\vec{f}_{gen}$. The list is sorted in ascending order of generation count and, as a consequence, also in non-increasing order of makespan. $\vec{f}_{gen}$ is used after the GA's completion as input for the VNS.

The GA is chosen as the metaheuristic framework for the FSGS because of two reasons. First, crossover as the GA's main operation is rather coarse-grained and can lead to huge modifications in the encoded solutions. This behavior is desired because $\vec{f}_{gen}$ shall contain different encoded solutions, although their makespans may be similar or identical. A fine-grained local improvement of these solutions is later obtained by the VNS. Second, efficient permutation-based crossover operators are present in literature and can easily be applied to all three input parameters of the FSGS.

4.2.2.1 Initial Population

In the initial population, the activity list λ is constructed by iteratively adding one activity whose predecessors are already contained in λ. The dummy source activity is implicitly assumed to be always present in the list. Diversity is introduced into the population by selecting the next activity randomly as well as by the priority rules employed in Fündeling (2006), namely longest path following (LPF), most work remaining (MWR), and most total successors (MTS). For 75% of the population, ρ_i is set with a probability of 10% to a uniform random integer value within the bounds $\underline{\rho}_i$ and $\overline{\rho}_i$. For the rest of the population $\rho_i = 0$ applies. All σ_i are set to 0 in order to prevent delayed activity starts in the initial population.

4.2.2.2 Operators

The proposed GA uses the partially-matched two-point crossover of Hartmann (1998) to recombine solutions. The operator maintains the precedence order of λ. It randomly selects two crossover points in lists λ, ρ, and σ of two parent solutions and generates two new feasible child solutions. Then the GA applies three mutation operators. The mutation probabilities are listed in the implementation details given in Section 7.3. To mutate λ, the operator of Hartmann (1998) is used. It exchanges an activity in λ with the one at the next position with a certain probability if the exchange is precedence-feasible. Each ρ_i is mutated with a certain probability by either replacing it by a uniform random integer within its constantly updated bounds or increasing (decreasing) it by one. Both cases are selected with equal probability. Finally, each σ_i is replaced with a certain probability by a uniform random integer within its constantly updated bounds.

4.2.3 Variable Neighborhood Search

The VNS, introduced by Mladenović and Hansen (1997), is a metaheuristic that combines local search with systematic change of neighborhoods. In the HM, the VNS is used to further locally improve the GA's best solutions by transferring resource quantities between selected pairs of activities (i, j). Given that potential to reduce the makespan exists, resource quantities are removed from the resource profiles of activity i and added to the resource profiles of activity j.

The VNS is an appropriate metaheuristic for this purpose, as its concept of nested neighborhoods represents combinations of multiple pairwise resource transfers. Whereas in the original method of Mladenović and Hansen (1997), a random perturbation is used to escape local minima, the proposed VNS escapes local minima by moving to the next solution from the GA's result list $\vec{f}_{gen}$. Thus, the VNS intensifies the search along the GA's search trajectory on solutions of already high quality.

Section 4.2.3.1 describes the heuristic selection of activity pairs, Section 4.2.3.2 explains the resource transfer between activities, and Section 4.2.3.3 presents the overall VNS framework along with a definition of neighborhoods. The VNS is illustrated in an example in Section 4.2.3.4.

4.2.3.1 Activity Selection

For most activity pairs a resource transfer is either infeasible or does not reduce the makespan. Hence, the VNS only operates on a subset of heuristically identified activity pairs. These activity pairs $(i, j) \in \mathcal{V}^{pair}$ are selected such that resource quantities are transferred from "non-critical" activity i to "critical" activity j in order to reduce the duration of j. Critical activities, as defined below, correspond to a longest path in the resource flow network of Artigues et al. (2003). This longest path is interpreted as the solution's critical path (Kelley, 1963). Therefore, reducing the durations of critical activities may also reduce the makespan. Set $\mathcal{V}^{pair}$ for solution f is constructed in five steps:

1. **Prevent cycles:** Due to the flexible resource profiles, the resource quantity allocated to an activity over time may first decrease and then increase again. In this case, there is a resource flow from the activity via other activities to itself. Hence, the resulting resource flow network contains cycles and the calculation of a longest path becomes $\mathcal{NP}$-hard (Garey and Johnson, 1979, p. 213). To derive an acyclic resource flow network, solution f is transformed into solution f^{rec} in which all activities $i^{rec} \in \mathcal{V}^{rec}$ feature resource profiles of rectangular shape for all required resources. Resource profiles that contain more than one block are split into multiple new activities that each represent one single block. In each period in which an activity from solution f starts a new block, a new corresponding activity $i^{rec} \in \mathcal{V}^{rec}$ starts in solution f^{rec}.

2. **Generate resource flow network:** An acyclic resource flow network is derived from f^{rec}. A modified version of the algorithm of Artigues et al. (2008, p. 34) is used to calculate the resource flows. An arc from activity i^{rec} to activity j^{rec}, with $i^{rec}, j^{rec} \in \mathcal{V}^{rec}$, is set if there is a positive resource flow from i^{rec} to j^{rec}. Different from Artigues et al. (2008), the algorithm first selects i^{rec} depending on whether it corresponds to the same original activity from $\mathcal{V}$ as j^{rec} and then depending on the earliest completion period. The weight of the arc is set to $\min(s_{j^{rec}} - s_{i^{rec}}, d_{i^{rec}})$.

3. **Identify critical activities:** A given maximum number of longest paths in the resource flow network is calculated by depth-first search. For each resulting longest path, an activity $j \in \mathcal{V}$ of solution f is critical if it corresponds to an activity $j^{rec} \in \mathcal{V}^{rec}$ that is on this longest path. Set $\mathcal{V}_{\mathcal{C}}$ contains all critical activities $j \in V$ related to the longest path.

4. **Select activity pairs:** Set $\mathcal{V}^{pair}$ contains activity pairs (i, j) in which activity i is non-critical, activity j is critical and has a duration above its minimum duration, and set $\mathcal{T}_{ij}^{pair}$ of periods for resource transfer is non-empty. $\mathcal{T}_{ij}^{pair}$ are the periods in which activity i has an allocated quantity above its lower usage bound, while activity j has a quantity below its upper usage bound for shared principal or independent resource r:

$$\mathcal{V}^{pair} = \left\{(i,j) \,\middle|\, i \notin \mathcal{V}_\mathcal{C} \wedge j \in \mathcal{V}_\mathcal{C} \wedge d_j > \underline{d}_j \wedge \mathcal{T}^{pair}_{ij} \neq \emptyset\right\} \tag{4.3}$$

$$\mathcal{T}^{pair}_{ij} = \left\{t \in \mathcal{T} \,\middle|\, \exists r \in \mathcal{R}^{pi}_i \cap \mathcal{R}^{pi}_j : (q_{irt} > \underline{q}_{ir} \wedge q_{jrt} < \overline{q}_{jr})\right\} \tag{4.4}$$

5. **Break ties:** In case of multiple longest paths, multiple sets $\mathcal{V}^{pair}$ may occur. To break ties, select the set with the highest makespan reduction potential:

$$\sum_{(i,j)\in\mathcal{V}^{pair}} (d_j - \underline{d}_j) \tag{4.5}$$

4.2.3.2 Resource Transfer

For $(i,j) \in \mathcal{V}^{pair}, r \in \mathcal{R}^{pi}_i \cap \mathcal{R}^{pi}_j$ and $t \in \mathcal{T}^{pair}_{ij}$, the maximum transfer quantity χ_{ijrt} is defined as:

$$\chi_{ijrt} = \min(q_{irt} - \underline{q}_{ir}, \overline{q}_{jr} - q_{jrt}) \tag{4.6}$$

χ_{ijrt} is the minimum of the maximum sendable and receivable resource quantities. The quantity of the principal resource determines the required quantities of dependent resources.

A new feasible solution f' is generated from the existing solution f by performing v resource transfers for v activity pairs $(i,j) \in \mathcal{V}^{pair}$. The FSGS generates solution f' from the representation (λ, ρ, σ) of solution f. However, in each period $t \in \mathcal{T}^{pair}_{ij}$, the allocated resource quantity in Steps 2a and 2b of Algorithm 1 is modified to $q_{irt} = \max(\underline{q}_{ir}, \min(\overline{q}_{ir}, q_{irt} - \chi_{ijrt}))$ for sending activity i and to $q_{jrt} = \max(\underline{q}_{jr}, \min(\overline{q}_{jr}, q_{jrt} + \chi_{ijrt}))$ for receiving activity j. In Step 2b of Algorithm 1, a new block is already started if the remaining resource requirement ξ_{ir} suffices to fulfill at least one minimum block length. The resource transfer is only performed if both activities meet the minimum block length. This requirement is not considered in the activity selection, as the block lengths resulting from multiple resource transfers are unknown in advance. Information on the performed resource transfers is saved such that the resulting solution f' can be used as input for additional resource transfers.

4.2.3.3 Solution Improvement

In the VNS, neighborhood number v of feasible solution f consists of all feasible solutions that can be generated from f by v resource transfers for activity pairs $(i, j) \in \mathcal{V}^{pair}$. The VNS integrates two components: (1) the first-improvement variable neighborhood descent of Hansen and Mladenović (2005), which is a local search method employing a purely deterministic neighborhood change mechanism, and (2) a perturbation method to escape local minima by using $\vec{f}_{gen}$ from the GA:

1. The VNS starts in neighborhood $v = 1$ of the last solution f from $\vec{f}_{gen}$. This solution has the best makespan found by the GA. Each local search step in neighborhood v is to generate a new solution f' from the current incumbent f by v resource transfers with activity pairs from $\mathcal{V}^{pair}$. If the makespan is not improved within the given maximum number of generated solutions per neighborhood, the VNS proceeds to the next neighborhood $v = v + 1$. If the makespan of f' is reduced, the VNS moves from f to f' and continues the search in neighborhood $v = 1$ of solution f'.

2. If f is not improved within the given maximum number of non-improving solutions or if the maximum allowed neighborhood has been reached, the VNS escapes the local optimum f by selecting the next unique solution f' from $\vec{f}_{gen}$ and moving to f'. Thus, the VNS exploits the results of the GA and operates on solutions of already high quality instead of using randomly generated solutions.

The VNS terminates and returns the overall best-found solution if $\vec{f}_{gen}$ has been fully processed or the given maximum number of solutions F_{VNS} has been generated.

4.2.3.4 Example

Assume that solution f from Figure 4.6 is processed by the VNS. In the activity selection, solution f is transformed into solution f^{rec} with rectangular-shaped

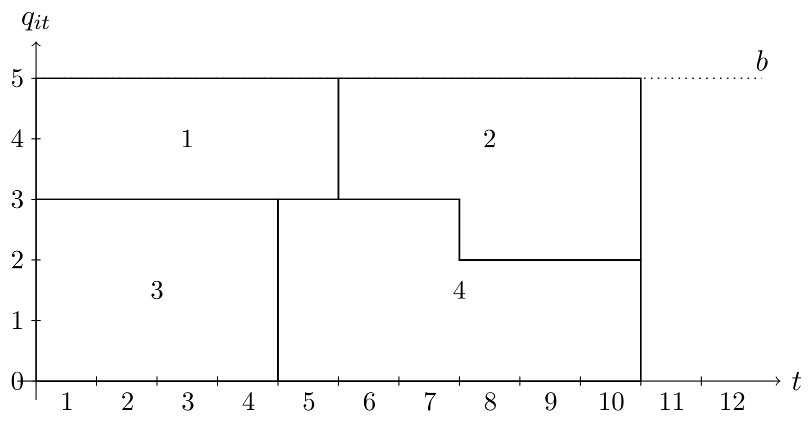

Figure 4.7: Improved solution f' resulting from solution f of Figure 4.6

resource profiles. The resource profile of activity 2 in f consists of 2 blocks. Hence, it corresponds to 2 activities with rectangular shaped resource profiles in f^{rec}: activity $2_a^{rec} \in \mathcal{V}^{rec}$ from periods 6 to 9 and activity $2_b^{rec} \in \mathcal{V}^{rec}$ from periods 10 to 11.

The resulting resource flow network contains two longest paths of length 11: $(1, 2_a^{rec}, 2_b^{rec})$ and $(3, 4, 2_b^{rec})$. For the first longest path with critical activities $\mathcal{V}_\mathcal{C} = \{1, 2\}$, activities 1 and 2 are parallel to activity 4. Hence, transfer periods $\mathcal{T}_{4,1}^{pair} = \{5\}$ and $\mathcal{T}_{4,2}^{pair} = \{6, 7, 8, 9\}$ result. This leads to activity pairs $\mathcal{V}^{pair} = \{(4, 1), (4, 2)\}$ with a reduction potential of $(d_1 - \underline{d}_1) + (d_2 - \underline{d}_2) = (5 - 3) + (6 - 5) = 3$ periods. The second longest path $(3, 4, 2_b^{rec})$ with critical activities 3, 4, and 2 has no reduction potential. For all critical activities, the only parallel activity which is not critical itself is activity 1. As the allocated resource quantity of activity 1 is equal to its lower usage bound in all periods, no resource transfer is possible.

Using $\mathcal{V}^{pair}$, the VNS starts in neighborhood $v = 1$ of f. This neighborhood contains all solutions resulting from $v = 1$ resource transfer. The first resource transfer for activity pair $(4, 1)$ with $\mathcal{T}_{4,1}^{pair} = \{5\}$ in period 5 is not performed, as activity 1's remaining resource requirement of 2 units does not suffice to accommodate the minimum block length. For the resource transfer of activity pair $(4, 2)$ with $\mathcal{T}_{4,2}^{pair} = \{6, 7, 8, 9\}$, the minimum block lengths of activities 4 and 2 prevent a transfer in periods 6 and 7. In period 8, a transfer[1] of $\chi_{4,2,8} = \min(3-2, 3-2) = 1$ units is feasible, resulting in 3 resource units allocated to activity 2 and 2 units allocated to activity 4, as shown in solution f' of Figure 4.7. Due to the minimum block length, both activities have the same resource allocation in period 9 and complete in period 10. As a result, the makespan is reduced by 1 period to the optimum of 10.

[1]The resource index is omitted in $\chi_{4,2,8}$.

Chapter 5

The Self-adaptive Genetic Algorithm

The self-adapting genetic algorithm (SGA) of Tritschler et al. (2014a) is briefly presented in this section. Since it is a metaheuristic that also integrates SGSs, it is employed to directly compare the HM's performance. Section 5.1 outlines its overall design before details on the applied SGSs are given in Sections 5.2 and 5.3.

5.1 Overall Design

The SGA adapts the approach of Hartmann (2002) to the FRCPSP. It uses an activity list solution representation and SGSs to decode the representation into feasible solutions. As a study by Fündeling (2006) on the FRCPSP showed no clear dominance of the serial or the parallel SGS on all considered problem instances, the SGA uses both SGS types. Both types are standard SGSs, as they schedule activities as early as possible and allocate the maximum resource quantities. The parameter that determines the applied SGS type is part of the solution representation and undergoes evolution itself. As a result, the SGA can self-adapt the SGS type to the problem instance. To summarize, a solution is represented by λ and the SGS type.

In the initial population, λ is generated as described in Section 4.2.2.1 by using the priority rules LPF, MWR, and MTS as well as random activity selection. The SGS type is randomly assigned with equal distribution. In each generation, solutions are selected depending on their fitness, which equals to the makespan. Elitism is used as done in the HM's GA from Section 4.2.2. The crossover and mutation operators of Hartmann (2002) are applied on the selected solutions in order to create the next generation.

5.2 Serial Schedule Generation Scheme

The SGA uses the serial SGS of Naber and Kolisch (2014b). Following the sequence of λ, the algorithm schedules in each iteration a single activity i as early as possible and allocates resources as much as possible. In each period, the algorithm allocates the maximum possible resource quantity to activity i until its resource requirements and the minimum block length are fulfilled and the activity completes. 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 minimum resource usage bounds, the starting period of i is revised. The algorithm terminates when all activities have been completed.

5.3 Parallel Schedule Generation Scheme

The parallel SGS increments time periods and considers in each iteration a decision period t. First, it allocates resources to active activities to guarantee their nonpreemption and to ensure that minimum block lengths are met. The remaining resources are then distributed among active activities that qualify for a change in resource allocation and eligible activities. To determine the resource quantities allocated to each activity, the sequence of activities in λ and the lower resource usage bounds are considered. The algorithm terminates when all activities have been completed.

Chapter 6

Model-based Metaheuristics

This chapter introduces two model-based metaheuristics for the FRCPSP. The methods integrate concepts from Benders decomposition (Benders, 1962) into GAs.

To the best of my knowledge, I am not aware of any applications of Benders decomposition to the FRCPSP or the underlying RCPSP. For the related MRCPSP, Boschetti and Maniezzo (2009) propose a Benders decomposition heuristic in which both the mode assignment master problem and the scheduling subproblem are solved by heuristics. Li (2015) combines mathematical programming and constraint programming in a hybrid Benders decomposition for project scheduling with multi-purpose resources. He solves the scheduling subproblem, which is a pure feasibility problem, by constraint programming techniques and infers “logic-based” cuts to the master assignment problem to exclude infeasible assignments. For other optimization problems, Raidl (2015) provides a most recent literature review of metaheuristics that integrate Benders decomposition or other mathematical decomposition techniques.

Before the proposed model-based metaheuristics are presented in Section 6.2, the applied mathematical models are introduced in Section 6.1. Table 6.1 summarizes the notation of the models.

Table 6.1: Notation used in the models

Indices	
i, j	Activity
r, k	Resource, specifically the principal resource k
t	Time period
Sets	
$\mathcal{E}_\mathcal{C}$	Precedence relations: Conjunctive arcs
$\mathcal{H}$	Solved subproblems
$\mathcal{R}$	Resources
$\mathcal{R}_i$	Required resources of activity i
$\mathcal{T}$	Discrete time horizon
$\mathcal{T}_i$	Time window for activity i
$\mathcal{T}_r^\delta$	Interval start periods for resource r
$\mathcal{T}_{ir}^\delta$	Interval start periods for activity i and resource r
$\mathcal{V}$	Activities
Ω	Tuples of activity i, principal k, and dependent resource r
Parameters	
b_r	Availability of resource r in each period
$\underline{c}_i, \overline{c}_i$	Earliest and latest completion period of activity i
$\underline{d}_i, \overline{d}_i$	Lower and upper bounds of duration of activity i
e_{irt}	Start of the interval of act. i and res. r that contains per. t
g_{rt}	Duration of the interval of res. r that contains per. t
g_{irt}	Duration of the interval of act. i and res. r that contains per. t
l_{ir}	Minimum block length of activity i and resource r
$\underline{q}_{ir}, \overline{q}_{ir}$	Lower and upper usage bounds of activity i for resource r
$\underline{s}_i, \overline{s}_i$	Earliest and latest start period of activity i
T_{min}	Lower bound of the makespan
w_{ir}	Requirement of activity i for resource r
α_{ir}, β_{ir}	Coefficient and constant of linear resource function
π_o, π_p	Penalties for outsourcing and allocated resource quantity
$\gamma_{rt}^h, \zeta_{irt}^h, \theta_{irt}^h, \iota_{ikrt}^h, \kappa_{ir}^h, \mu_{irt}^h, \nu_{irt}^h$	Dual values associated to SP constraints (6.17) to (6.23)
Binary variables	
z_{it}	1 if activity i is active in period t, 0 otherwise
δ_{irt}	1 if activity i is allowed to change its allocated quantity of resource r from period $t-1$ to t, 0 otherwise
Integer variables	
C_{max}	Project makespan
Continuous vars.	
o_{rt}	Outsourcing of resource r in period t
q_{irt}	Quantity of resource r allocated to activity i in period t
η	MP objective function value

6.1 Decomposed Models

Using the reformulation of Benders (1962), the FRCPSP is decomposed into a master problem (MP) and a subproblem (SP). In the MP, activities are scheduled by determining their start periods and durations and additionally the minimum block length is enforced. In the subproblem (SP), the resource profiles are created by allocating resources. The idea of Benders decomposition is that the two separate problems are easier to solve than the original problem. Constraints in the form of Benders cuts are only gradually added to the MP in an iterative process. By fixing the "complicating" (Benders, 1962) decision variables of a solution to the MP, the resulting SP is a linear program (LP). When the SP is solved to optimality, the optimal dual values associated to each SP constraint can consequently be derived. From these dual values, a Benders cut is generated and added to the MP. The collective set of Benders cuts guides the MP in successive iterations towards improved MP solutions. Ultimately the procedure is expected to converge towards an optimal complete solution.

The MP is introduced in Section 6.1.1 and the SP in Section 6.1.2. Section 6.1.3 provides the reduced subproblem (RSP). The RSP is a reformulation of the SP with a reduced number of constraints and variables.

6.1.1 Master Problem

The purpose of the MP is to determine for each activity a start period and a duration as well as intervals of periods that have to feature a constant resource allocation in order to fulfill the minimum block length. Depending on these decisions, resources are allocated by solving the SP, from which a Benders optimality cut is derived and added to the MP. If the allocated resource quantities per period exceed the resource availability in the SP, the cut imposes a penalty for the resource constraint violation into the MP objective function value. The MP is modeled as follows:

$$\text{Minimize} \quad \eta \tag{6.1}$$

subject to

$$\sum_{t\in\mathcal{T}_i} z_{it} \geq \underline{d}_i \qquad \forall i \in \mathcal{V} \tag{6.2}$$

$$\sum_{t\in\mathcal{T}_i} z_{it} \leq \overline{d}_i \qquad \forall i \in \mathcal{V} \tag{6.3}$$

$$\sum_{i\in\mathcal{V}} \underline{q}_{ir} \cdot z_{it} \leq b_r \qquad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{6.4}$$

$$z_{i(\underline{s}_i-1)} = z_{i(\overline{c}_i+1)} = 0 \qquad \forall i \in \mathcal{V} \tag{6.5}$$

$$\delta_{irt} \geq -z_{i(t-1)} + z_{it} \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i, t \in \mathcal{T}_i \cup \{\overline{c}_i+1\} \tag{6.6}$$

$$\delta_{irt} \geq z_{i(t-1)} - z_{it} \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i, t \in \mathcal{T}_i \cup \{\overline{c}_i+1\} \tag{6.7}$$

$$\delta_{irt} \leq z_{i(t-1)} + z_{it} \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i, t \in \mathcal{T}_i \cup \{\overline{c}_i+1\} \tag{6.8}$$

$$C_{max} \geq t \cdot z_{it} \qquad \forall i \in \mathcal{V}, t \in \mathcal{T}_i \tag{6.9}$$

$$\sum_{\tau\leq t} z_{j\tau} \leq \overline{d}_j(1-z_{it}) \qquad \forall (i,j) \in \mathcal{E}_c, t \in \mathcal{T}_i \cap \mathcal{T}_j \tag{6.10}$$

$$\sum_{\tau\geq t+1} z_{i\tau} + \overline{d}_i(z_{it} - z_{i(t+1)}) \leq \overline{d}_i \qquad \forall i \in \mathcal{V}, t \in \mathcal{T}_i \tag{6.11}$$

$$\sum_{\tau=t}^{t+l_{ir}-1} \delta_{irt} \leq 1 \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i, t \in \mathcal{T}_i \tag{6.12}$$

$$\begin{aligned}
\eta \geq\ & C_{max} \qquad \forall h \in \mathcal{H} \\
& - \sum_{r\in\mathcal{R}} \sum_{t\in\mathcal{T}} \gamma^h_{rt} \cdot b_r \\
& + \sum_{i\in\mathcal{V}} \sum_{r\in\mathcal{R}_i} \sum_{t\in\mathcal{T}_i} \zeta^h_{irt} \cdot \underline{q}_{ir} \cdot z_{it} \\
& - \sum_{i\in\mathcal{V}} \sum_{r\in\mathcal{R}_i} \sum_{t\in\mathcal{T}_i} \theta^h_{irt} \cdot \overline{q}_{ir} \cdot z_{it} \\
& + \sum_{(i,k,r)\in\Omega} \sum_{t\in\mathcal{T}_i} \iota^h_{ikrt} \cdot \beta_{ir} \cdot z_{it} \\
& + \sum_{i\in\mathcal{V}} \sum_{r\in\mathcal{R}_i} \kappa^h_{ir} \cdot w_{ir} \\
& - \sum_{i\in\mathcal{V}} \sum_{r\in\mathcal{R}_i} \sum_{\substack{t\in\mathcal{T}_i\cup \\ \{\overline{c}_i+1\}}} \mu^h_{irt} \cdot \overline{q}_{ir} \cdot \delta_{irt} \\
& - \sum_{i\in\mathcal{V}} \sum_{r\in\mathcal{R}_i} \sum_{\substack{t\in\mathcal{T}_i\cup \\ \{\overline{c}_i+1\}}} \nu^h_{irt} \cdot \overline{q}_{ir} \cdot \delta_{irt}
\end{aligned} \tag{6.13}$$

$$z_{it} \in \{0,1\} \qquad \forall i \in \mathcal{V}, t \in \mathcal{T}_i \tag{6.14}$$

$$\delta_{irt} \in \{0,1\} \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i, t \in \mathcal{T}_i \cup \{\overline{c}_i+1\} \tag{6.15}$$

The MP is an MIP with the objective (6.1) of minimizing the continuous variable η. The model features two binary decision variables. z_{it} is 1, if activity i is active in period t, or 0, otherwise. Consequently, s_i is the earliest period with $z_{it} = 1$ and c_i the latest period. If δ_{irt} is 0, the allocated quantity of resource r to activity i remains constant from period $t-1$ to t. If its value is 1, the allocated resource quantity may change and, thus, a new block may start. However, whether a block actually starts in the resource profile is determined by the SP.

Constraints (6.2) and (6.3) ensure that activity durations are within the bounds $\underline{d}_i$ and $\overline{d}_i$. Constraint (6.4) ensures that the sum of the lower resource usage bounds $\underline{q}_{ir}$ of active activities does not exceed the resource availability b_r. This prevents cases of resource constraint violations that are already obvious in the MP. In constraint (6.5), z_{it} is set to 0 in periods $\underline{s}_i - 1$ and $\overline{c}_i + 1$, which are required by constraints (6.6) to (6.8). Constraint (6.6) sets $\delta_{irt} = 1$ in the activity start period s_i, while constraint (6.7) sets $\delta_{irt} = 1$ in period $c_i + 1$ after the completion period of the activity. Both constraints together ensure that the first block of activity i starts in s_i while the last block completes in c_i. Constraint (6.8) assures that additional blocks may only start while the activity is active. Constraints (6.9) to (6.15) are taken from model FP-DT1 of Naber and Kolisch (2014b) and are briefly summarized. Constraint (6.9) determines C_{max} as the last period in which any activity is active. The model features precedence relations (6.10) and non-preemption constraints (6.11). Constraint (6.12) enforces the minimum block length l_{ir}. The binary decision variables are defined in (6.14) and (6.15).

The Benders optimality cuts are given in constraint (6.13). For each subproblem $h \in \mathcal{H}$ that has been solved to optimality, one Benders optimality cut is generated. The cut forces η to be greater than or equal to the sum of C_{max} and the objective function value of the dual SP, which is expressed as the summation of the products of the SP right-hand-sides multiplied by the optimal dual values γ^h_{rt}, ζ^h_{irt}, θ^h_{irt}, ι^h_{ikrt}, κ^h_{ir}, μ^h_{irt}, and ν^h_{irt} that are associated to SP constraints (6.17) to (6.23), respectively. Due to the formulation of the SP, Benders feasibility cuts (Benders, 1962) are not required, as explained in the next section.

6.1.2 Subproblem

The SP determines the resource allocation after the values of z_{it} and δ_{irt} have been set in the MP. The SP is modeled as a linear program:

$$\text{Minimize} \quad \pi_o \cdot \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} o_{rt} + \pi_q \cdot \sum_{i \in \mathcal{V}} \sum_{r \in \mathcal{R}_i} \sum_{t \in \mathcal{T}_i} q_{irt} \tag{6.16}$$

subject to

$$\sum_{i \in \mathcal{V}} q_{irt} - o_{rt} \leq b_r \qquad \forall r \in \mathcal{R}, t \in \mathcal{T}_i \tag{6.17}$$

$$q_{irt} \geq \underline{q}_{ir} \cdot z_{it} \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i, t \in \mathcal{T}_i \tag{6.18}$$

$$q_{irt} \leq \overline{q}_{ir} \cdot z_{it} \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i, t \in \mathcal{T}_i \tag{6.19}$$

$$q_{irt} - \alpha_{ir} \cdot q_{ikt} \geq \beta_{ir} \cdot z_{it} \qquad \forall (i,k,r) \in \Omega, t \in \mathcal{T}_i \tag{6.20}$$

$$\sum_{t \in \mathcal{T}_i} q_{irt} \geq w_{ir} \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i \tag{6.21}$$

$$q_{irt} - q_{ir(t-1)} \leq \overline{q}_{ir} \cdot \delta_{irt} \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i, t \in \mathcal{T}_i \cup \{\overline{c}_i + 1\} \tag{6.22}$$

$$q_{ir(t-1)} - q_{irt} \leq \overline{q}_{ir} \cdot \delta_{irt} \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i, t \in \mathcal{T}_i \cup \{\overline{c}_i + 1\} \tag{6.23}$$

$$o_{rt} \geq 0 \qquad \forall r \in \mathcal{R}, t \in \mathcal{T} \tag{6.24}$$

$$q_{irt} \geq 0 \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i, t \in \mathcal{T}_i \tag{6.25}$$

The SP objective (6.16) is to determine a resource profile that minimizes the weighted sum of the allocated resource quantity and outsourcing. By introducing **outsourcing**, represented as the continuous decision variable o_{rt}, the resource constraints are relaxed. In other words, allocating a continuous resource quantity q_{irt} that exceeds the resource availability b_r is allowed. Specifically, o_{rt} defines the additional quantity of resource r allocated in period t beyond the resource availability b_r, similar to the critical resource variables of Deckro and Hebert (1989). The outsourcing ensures that the SP is always feasible and Benders feasibility cuts are not required. As resource profiles without outsourcing are desired, o_{rt} is penalized in the objective function with a sufficiently large cost factor π_o. This penalty ensures that a feasible solution results in a better objective function value in the MP than a resource infeasible solution with outsourcing. The allocated resource quantity q_{irt} is only minimized in order to identify slacks. The **slack** is the quantity of unallocated resources per period. Without minimizing q_{irt}, slacks may not be observed as it is possible to overallocate resources. Since the objective function should effectively only penalize outsourcing but not the allocated resource quantity, a sufficiently small cost factor π_q is required such that the weighted sum of the resource quantities is smaller than one period: $\pi_q \cdot \sum_{i \in \mathcal{V}} \sum_{r \in \mathcal{R}_i} \sum_{t \in \mathcal{T}_i} q_{irt} < 1$. This ensures that an MP solution with a smaller makespan always receives a better objective function value than a solution with a higher makespan, regardless of the quantity of allocated resources.

The resource constraint (6.17) introduces outsourcing beyond the resource availability. The model can easily be generalized to time-variant resource availabilities by considering different values of b_r for each time period. Constraints (6.18) to (6.23) are taken from Naber and Kolisch (2014b) and are briefly summarized. The lower and upper bounds on allocated resource quantity are defined in constraints (6.18) and (6.19), respectively. For each tuple $(i, k, r) \in \Omega$, the allocated quantity q_{irt} of dependent resource r is defined as a linear function of the allocated quantity q_{ikt} of the principal resource k (6.20). The resource requirement is modeled in (6.21). The start of a block is defined through a change up (6.22) or down (6.23) in the allocated resource quantity. In periods with $\delta_{irt} = 1$, a new block can but not necessarily has to start. In periods with $\delta_{irt} = 0$, the current block has to continue. Finally, the domains of the continuous decision variables are defined in (6.24) and (6.25).

6.1.3 Reduced Subproblem

The RSP is a reformulation of the SP that takes advantage of the block structure of resource profiles. The RSP is defined on intervals of periods with a constant quantity of allocated resources instead of single periods. Each interval is solely represented by its start period, whereas its remaining periods with constant resource allocation are not explicitly modeled. Based on the fixed MP variables, two reduced sets of periods are defined:

$$\mathcal{T}_{ir}^{\delta} = \left\{ t \in \mathcal{T} \,\middle|\, z_{it} = 1 \land \delta_{irt} = 1 \right\} \tag{6.26}$$

$$\mathcal{T}_{r}^{\delta} = \left\{ t \in \mathcal{T} \,\middle|\, \sum_{i \in \mathcal{V}} z_{it} \geq 1 \land \sum_{i \in \mathcal{V}} \delta_{irt} \geq 1 \right\} \tag{6.27}$$

Set $\mathcal{T}_{ir}^{\delta}$ from (6.26) contains the **interval start periods of activity** i **and resource** r. These are periods in which activity i is active and may change its allocated quantity of resource r by starting a new block. $\mathcal{T}_{r}^{\delta}$ from (6.27) contains the **interval start periods of resource** r. These are periods in which any active activity may change its allocated quantity of resource r.

An **interval for activity** i **and resource** r is defined as the contiguous periods beginning in an interval start period from $\mathcal{T}_{ir}^{\delta}$ up to (exclusive) either the next interval start period from $\mathcal{T}_{ir}^{\delta}$ or period $c_i + 1$. Let $e_{irt} \in \mathcal{T}_{ir}^{\delta}$ denote the interval start period of the interval for activity i and resource r that contains period t and let g_{irt} indicate the duration of this interval. Similarly, an **interval for resource** r are the contiguous periods beginning in an interval start period from $\mathcal{T}_{r}^{\delta}$ up to (exclusive) either the next interval start period from $\mathcal{T}_{r}^{\delta}$ or period $C_{max} + 1$. The duration of the interval for resource r that contains period t is given by g_{rt}. Since e_{irt}, g_{irt}, and g_{rt} are parameters, the RSP is a linear program:

$$\text{Minimize} \quad \pi_o \cdot \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}_{r}^{\delta}} o_{rt} \cdot g_{rt} + \pi_q \cdot \sum_{i \in \mathcal{V}} \sum_{r \in \mathcal{R}_i} \sum_{t \in \mathcal{T}_{ir}^{\delta}} q_{irt} \cdot g_{irt} \tag{6.28}$$

subject to

$$\sum_{t \in \mathcal{T}_{ir}^{\delta}} q_{irt} \cdot g_{irt} \geq w_{ir} \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i \tag{6.29}$$

$$\sum_{\substack{i \in \mathcal{V} \\ z_{it}=1}} q_{ire_{irt}} - o_{rt} \leq b_r \qquad \forall r \in \mathcal{R}, t \in \mathcal{T}_{r}^{\delta} \tag{6.30}$$

$$q_{irt} \geq \alpha_{ir} \cdot q_{ikt} + \beta_{ir} \qquad \forall (i,k,r) \in \Omega, t \in \mathcal{T}_{ir}^{\delta} \tag{6.31}$$

$$o_{rt} \geq 0 \qquad \forall r \in \mathcal{R}, t \in \mathcal{T}_{r}^{\delta} \tag{6.32}$$

$$\underline{q}_{ir} \leq q_{irt} \leq \overline{q}_{ir} \qquad \forall i \in \mathcal{V}, r \in \mathcal{R}_i, t \in \mathcal{T}_{ir}^{\delta} \tag{6.33}$$

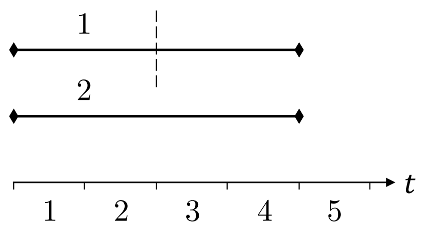

Figure 6.1: Intervals in the RSP

The RSP objective (6.28) is to determine a resource profile that minimizes the weighted sum of the allocated resource quantity and outsourcing. The optimal objective function value is identical to that of (6.16). In order to quantify the outsourcing and the allocated resource quantity of an interval, the continuous variables o_{rt} and q_{irt} are multiplied by the corresponding interval durations.

Constraint (6.29) ensures that the resource requirements are fulfilled. As only the interval start periods from $\mathcal{T}_{ir}^{\delta}$ are considered, the allocated resource quantity is multiplied by the duration g_{irt} of the interval of activity i and resource r. The resource constraints (6.30) are only defined in interval start periods $t \in \mathcal{T}_{r}^{\delta}$ for resource r. However, an activity's resource allocation may not be defined in all of these periods. Therefore, $q_{ire_{irt}}$ is used. It is the resource allocation in the interval start period e_{irt} of the interval of activity i and resource r that contains period t. Similar as in the SP, time-variant resource availabilities may easily be introduced. Constraint (6.31) represents the resource function for each tuple $(i, k, r) \in \Omega$ of activity i's principal resource k and dependent resource r. Constraint (6.32) defines the outsourcing o_{rt} and constraint (6.33) imposes the lower and upper resource usage bounds on variable q_{irt}.

The example in Figure 6.1 illustrates the interval concept for two activities and a single resource. The horizontal lines indicate periods with $z_{it} = 1$. The diamonds at both ends of the horizontal lines and the vertical dashed line for activity 1 illustrate (the beginning of) periods with $\delta_{irt} = 1$. Activity 1 features two intervals for the single resource: the first from period 1 to 2 and the second from 3 to 4. Activity 2 only features a single interval from period 1 to 4. Hence, constraint (6.29) for activity 2 is only defined at the activity's interval start period 1. The resource features an interval from period 1 to 2 and another from 3 to 4. The resource constraints (6.30) are defined at the interval start periods 1 and 3 of the resource. Since activity 2's resource allocation is not defined in period 3, the resource quantity $q_{ire_{irt}}$ for its interval start period $e_{irt} = 1$ is used in constraint (6.30).

The RSP does not explicitly contain the MP variables z_{it} and δ_{irt}. It also does not feature constraints for periods with $z_{it} = 0$. In contrast, the MP is defined for the whole time window $\mathcal{T}_i$. Due to complementary slackness properties, periods with $z_{it} = 0$ may as well feature non-zero dual values, which would then be missing in Benders cuts. Therefore, the RSP is only used to determine resource profiles but not to generate Benders optimality cuts.

6.2 Genetic Algorithms

Since the MP constitutes a scheduling problem with high symmetry, numerous iterations of the classic Benders decomposition method are likely required to solve the FRCPSP. Repeatedly solving the MP to optimality in each iteration becomes time consuming, especially after Benders optimality cuts have been added. Therefore, two metaheuristics are proposed to determine MP solutions: The Benders genetic algorithm (BGA) and the reduced subproblem genetic algorithm (RGA). Both GAs gradually improve a population of encoded MP solutions. Since they, directly or indirectly, derive the fitness of MP solutions from the resulting resource profiles by solving the SP or the RSP, they are categorized as model-based metaheuristics.

As the MP is not repeatedly solved to optimality anymore, the fitness evaluation becomes the most time consuming operation. Although solving a single SP is in the order of milliseconds, the high number of SPs makes the fitness evaluation computationally expensive. The BGA and the RGA take different measures to overcome this. The BGA approximates the fitness from Benders optimality cuts, instead of determining resource profiles for all encoded MP solutions. As a result, fewer SPs have to be solved. According to Poojari and Beasley (2009), a population-based metaheuristic, such as the proposed BGA, is particularly useful in this context, as multiple Benders optimality cuts are generated in a single iteration. The RGA, on the other hand, determines resource profiles for all MP solutions by solving the RSP, which requires less computation time than the SP.

MP solutions may result in infeasible resource profiles with outsourcing. In case such an infeasible solution has a low makespan, it is worthwhile to repair the infeasibility if the makespan is maintained or only marginally increased. Similarly, if an MP solution features slacks, there may be potential to further reduce the makespan. In both cases MP solutions have to be modified. Instead of solely relying on the GA's evolutionary process, two heuristic improvement operators are introduced. They directly use information on the resource profiles to locally improve MP solutions.

The next section explains the BGA's and the RGA's overall design before details on specific components are provided in the following sections.

6.2.1 Overall Design

A high level outline of the BGA and the RGA is provided in Figure 6.2. The steps that differ between the two methods are highlighted in gray.

Both methods encode solutions to the MP by using the representation proposed in Section 6.2.2. The encoded solutions are always feasible with regards to constraints (6.2) to (6.12). Beginning with the creation of an initial population, described in Section 6.2.3, a population of encoded MP solutions is evolved over multiple generations until the given maximum number of solutions F has been generated. In each generation, the population is split into two disjunctive subsets depending on the makespan: the **top solutions** and the **bottom solutions**. The top solutions are the most promising part of the population and are regarded as the source of improvement. MP solutions with a makespan that is at most 10% higher than the best makespan of the initial population are added to this set. Its size is limited to 20% of the population size. By just slightly improving these high quality solutions, a new best solution may be found. Therefore, the improvement operators, described in Section 6.2.4, are applied to locally improve the top solutions. The remaining bottom solutions have worse makespans and are mainly regarded as a source of diversity in the population. The fitness of an MP solution is the sum of its makespan and

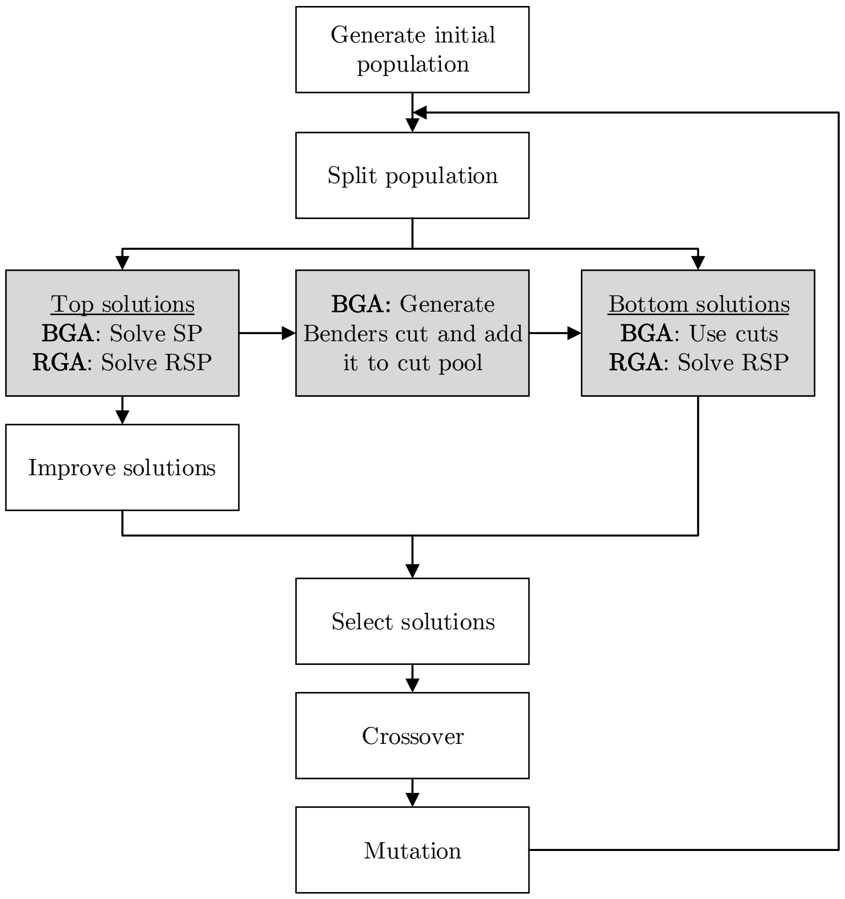

Figure 6.2: Overall design of the BGA and the RGA

the SP or RSP objective function value. Hence, MP solutions that result in infeasible resource profiles with outsourcing are penalized with a worse fitness. The elite solution with the best fitness of the current generation is inserted unmodified into the next generation. Using on the elite solution's makespan, both GAs update the upper bounds $\overline{s}_i$, $\overline{c}_i$, and $\overline{d}_i$, which are therefore tightened with each makespan improvement. The solutions for the next generation are selected depending on their fitness by stochastic universal sampling of Baker (1987), an improved version of roulette wheel selection (Reeves, 2010). The selected solutions are then recombined and modified by the genetic operators from Section 6.2.5. The BGA's and the RGA's different approaches to calculate the fitness are detailed in the next two sections.

6.2.1.1 Benders Genetic Algorithm

The BGA determines the fitness for the top and bottom solutions differently. The **fitness of top solutions** is obtained by solving the SP to optimality with sp^* denoting the optimal value of the SP objective function (6.16):

$$C_{max} + sp^* \tag{6.34}$$

Since solving a high number of SPs would become computationally expensive, the fitness of the large subset of bottom solutions is only approximated. For these solutions, the BGA follows the approach of Sirikum et al. (2007) to use η from the MP objective (6.1) as a lower bound of the fitness (6.34). η is obtained from Benders optimality cuts that are present in the cut pool. A Benders optimality cut is generated and added to the cut pool, each time the SP is solved for a top solution. The cut pool also contains cuts from previous generations. As the BGA only generates valid MP solutions, constraints (6.2) to (6.12), (6.14), and (6.15) can be ignored. In addition, the terms of the Benders optimality cuts (6.13) solely contain parameters. Hence, determining the value of the single remaining variable η in expression (6.35) is considerably faster than solving the SP:

$$\text{(6.1) subject to (6.13)} \tag{6.35}$$

In order to strengthen the fitness, the lower bound of outsourcing $\underline{o}_r$ from Appendix C is also considered. Let η^* denote the optimal value of expression (6.35). The **fitness of the bottom solutions** is:

$$\max\left(\eta^*, C_{max} + \pi_o \sum_{r \in \mathcal{R}} \underline{o}_r\right) \tag{6.36}$$

To limit the SP model size, an upper limit for the makespan applies. It is set at 120% of the best makespan of the initial population. Since solutions that exceed this bound are unlikely to produce high quality offspring, their fitness is penalized with a sufficiently large factor without solving the SP.

6.2.1.2 Reduced Subproblem Genetic Algorithm

The RGA reduces the computation time by using the RSP instead of the SP. It determines resource profiles for all MP solutions by solving the RSP to optimality. Let rsp^* denote the optimal value of the RSP objective function (6.28). The **fitness of top and bottom solutions** is:

$$C_{max} + rsp^* \tag{6.37}$$

The RGA limits the RSP model size by applying the same upper limit for the makespan as in the BGA.

6.2.2 Solution Representation

When an MP solution is decoded by the GAs, its resource profile has not been determined yet. Especially for the BGA, the resource profile may not even be determined at all. Common solution representations, such as the activity list or random keys (Hartmann, 1998), cannot be used in the decomposition because they specifically require knowledge on the resource allocation to generate schedules. For example, if an activity list is employed, values for q_{irt} are required in order to decide whether activities that are not connected by precedence relations are scheduled in parallel or sequentially. However, due to the decomposition, q_{irt} can only be determined after the scheduling decisions have been made. A representation using a vector of explicit start times is not appropriate either, because genetic operators would commonly violate the precedence relations, as criticized by Kolisch and Hartmann (2006).

A new solution representation for the FRCPSP is proposed. An MP solution is represented by a tuple (A, B, D) that consists of list A encoding disjunctive arcs, list B encoding blocks, and list D encoding activity durations. Section 6.2.2.1 first describes how start times and durations are represented by A and D. Then, Section 6.2.2.2 explains the encoding of blocks in B. The solution representation is illustrated in an example in Section 6.2.2.3.

6.2.2.1 Start Periods and Durations

To determine activity start periods, disjunctive arcs as of Shaffer et al. (1965) are used. A **disjunctive arc** $(i \to j) \in \mathcal{E}_{\mathcal{D}}$ is an additional precedence relation between two activities i and j that are not (transitively) connected by regular precedence relations from set $\mathcal{E}_{\mathcal{C}}$, the conjunctive arcs. In contrast to Alvarez-Valdés and Tamarit (1989), Bell and Han (1991), and similar approaches for the RCPSP, disjunctive arcs are not employed to resolve resource constraints by destroying minimal forbidden sets. Instead, the disjunctive arcs only determine activity start periods in the MP. Therefore, the solution representation can be limited to at most one incoming disjunctive arc for each activity j. Multiple incoming disjunctive arcs would be redundant, because the start period is determined by the completion period of the latest predecessor.

Set $\mathcal{E}_d$ of disjunctive arcs is encoded as list A that features at position j activity j's predecessor i from disjunctive arc $(i \to j)$. In case that j has no incoming disjunctive arc, the value at list position j is empty. Furthermore, activity durations are directly represented by list $D = (d_1, \ldots, d_n)$ that has activity i's duration d_i at list position i.

Let P_j denote the set of predecessors of activity j defined by conjunctive and disjunctive arcs. The start and completion periods as well as the corresponding variable z_{it} are decoded by creating an earliest start schedule that specifically respects the disjunctive arcs:

$$s_j = \max(\max_{i \in P_j}(c_i) + 1, \underline{s}_j) \tag{6.38}$$

$$c_i = s_i + d_i - 1. \tag{6.39}$$

$$z_{it} = \begin{cases} 1 & \text{if } s_i \leq t \leq c_i \\ 0 & \text{otherwise} \end{cases} \tag{6.40}$$

The decoding procedure checks whether MP constraint (6.4) is fulfilled in each period, such that the sum of the minimum resource usage bounds of active activities does not exceed the resource availability. If the constraint is violated, two activities i and j that are active in that period are randomly chosen and a disjunctive arc $(i \to j)$ is added. This step is repeated until the constraint is fulfilled.

The addition of disjunctive arcs may cause cycles in the precedence network. To detect cycles, the fast algorithm of Tarjan (1972) is applied on new child solutions after the mutation. To break a cycle, a randomly selected disjunctive arc is simply removed from it.

By inserting disjunctive acs, the start of activities can be delayed after the completion of other activities. Thus, the optimal solution from Figure 4.4b in Section 4.1.2 can be generated. To allow a fully free selection of activity start periods, the representation can easily be extended by adding an offset to the activity start periods. This option is evaluated in the computational study in Section 7.4.5.

6.2.2.2 Blocks

The MP specifies periods with $\delta_{irt} = 1$ in which a new block in the resource profile may start. According to the notation introduced in Section 6.1.3, these periods are interval start periods $t \in \mathcal{T}_{ir}^{\delta}$ for activity i and resource r. These interval start periods are encoded as offsets relative to the activity's start period s_i in list B_{ir}. For example, if an activity starts in period 10 and features an interval start in period 15, the resulting offset in B_{ir} is 5. List B contains B_{ir} for each combination of activity i and principal or independent resource r. Dependent resources are not encoded but the corresponding periods of the principal resource are used. The periods s_i and $c_i + 1$ are not encoded either, as $\delta_{irt} = 1$ holds due to constraints (6.6) and (6.7). δ_{irt} is decoded as follows:

$$\delta_{irt} = \begin{cases} 1 & \text{if } t \in \{s_i, c_i + 1\} \text{ or } t - s_i \in B_{ir} \\ 0 & \text{otherwise} \end{cases} \tag{6.41}$$

If the start and completion periods of an activity change, the activity's blocks may have to be reassigned such that they are synchronized with the blocks of other activities. This ensures that whenever an activity changes its allocated resource quantity, other activities may do so as well. Since a GA's random evolutionary process cannot efficiently update the underlying interval start periods, a heuristic is proposed. Its idea is that a block at one activity is associated with the start, the completion or a block of another activity. The heuristic sets interval start periods at these events while maintaining the minimum block length. B is updated in three steps:

1. **Synchronize blocks to activities:** Whenever an activity i starts in a period t or has completed in the previous period, add a new interval start period t to all other active activities that share resources with activity i. If two newly added interval start periods for an activity j and a resource r have a distance smaller than the minimum block length l_{jr}, remove one interval start period randomly.

2. **Enforce the minimum block length:** For each activity j and resource r, remove existing interval start periods that have a distance smaller than l_{jr} to the interval start periods added in Step 1.

3. **Synchronize blocks to other blocks:** For each interval start period t for resource r that remains after Step 2, add a new interval start period t to all other active activities that also require resource r. A new interval start period for an activity i and a resource r is only added if it has a distance of at least l_{ir} to activity i's existing interval start periods. Again, if two newly added interval start periods have a distance of less than l_{ir} to each other, remove one interval start period randomly.

6.2.2.3 Example

To illustrate the representation, consider the project from Figure 6.3. It contains a single resource with availability of 8 units and 5 non-dummy activities, each with a minimum block length of 2 periods. The resource index is omitted.

Table 6.2 presents an encoded MP solution. List A represents three disjunctive arcs $(2 \rightarrow 3)$, $(1 \rightarrow 4)$, and $(3 \rightarrow 5)$. Figure 6.4 illustrates the resulting MP solution in which the scheduled activities are depicted as horizontal lines. Activities 1 and 2 start immediately in period 1. Due to disjunctive arc $(2 \rightarrow 3)$, activity 3 can only start in period 5 after activity 2 is completed. Similarly, activity 4 starts in period 7 and activity 5 in period 10.

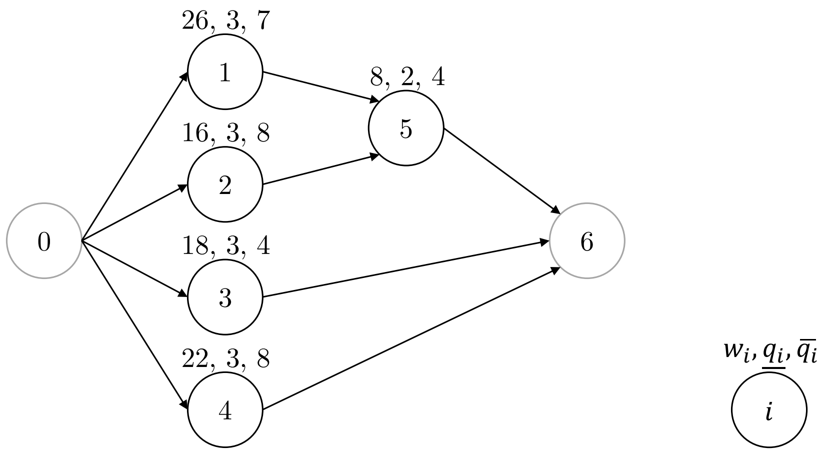

Figure 6.3: Project featuring a single resource with availability of $b = 8$ and a minimum block length of $l_i = 2$ for all activities

Then B is decoded. Activity 2 receives the interval start period 3, which is 2 periods after the start of the activity. Activity 3 features interval start period 8, which is 3 periods after the start of the activity. New interval start periods, illustrated as dashed vertical lines in Figure 6.4, are set according to Steps 1 to 3 from Section 6.2.2.2:

1. As activity 3 starts in period 5, activity 1 receives the interval start period 5 in Step 1. In a similar manner, activity 4 causes the interval start period 7 for activity 3 and activity 5 causes the interval start period 10 for activity 4.

2. In Step 2, activity 3's interval start period 8 is deleted, as it is less than one minimum block length away from the interval start period 7, which has been added in the Step 1.

3. In Step 3, activity 2's interval start period 3 causes activity 1 to obtain the interval start period 3.

By solving the SP or the RSP, the exemplary resource profile illustrated in Figure 6.5 is obtained. In periods 10 and 11, the resource allocation of 9 units exceeds the availability of 8 units, leading to 2 units of outsourcing. Note that the blocks of activities 1 and 2 continue across the interval start period 3.

Table 6.2: Encoded master problem solution for project from Figure 6.3

i	1	2	3	4	5
A			2	1	3
B		2	3		
D	6	4	5	5	2

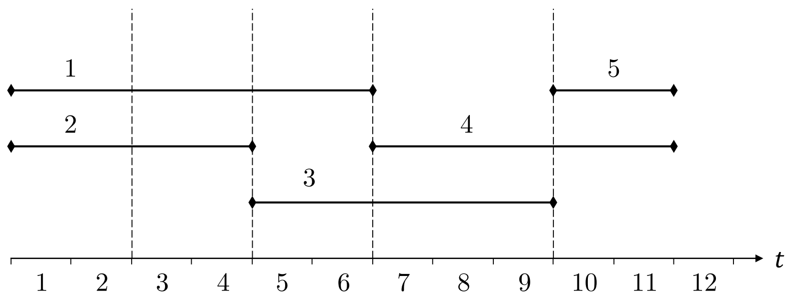

Figure 6.4: Master problem solution for encoding from Table 6.2

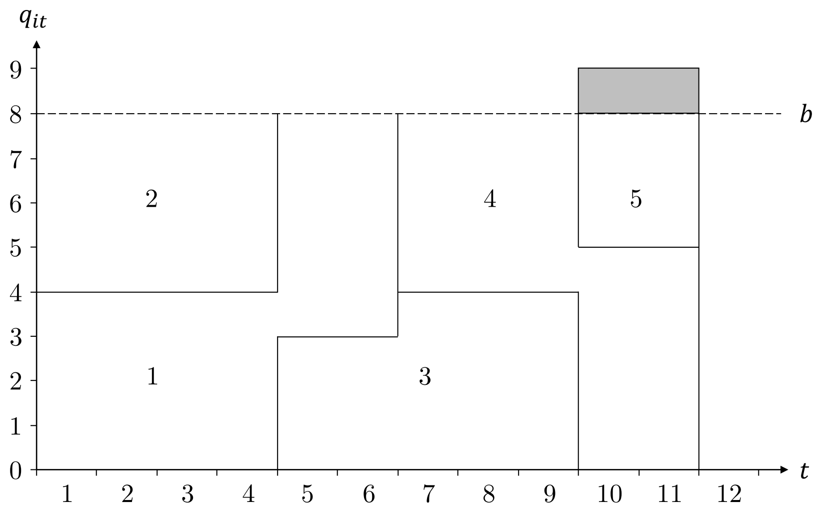

Figure 6.5: Resource profile for the master problem solution from Figure 6.4

6.2.3 Initial Population

As a random initial population would mostly lead to inconsistent MP solutions and resource profiles with outsourcing, both GAs employ the HM's FSGS to create an initial population as described in Section 4.2.2.1.

From the resulting solutions of the FSGS, activity start and completion periods are adopted by setting disjunctive arcs and durations accordingly. Activities may start earlier as determined by the FSGS in order to ensure an earliest start schedule with respect to the disjunctive arcs. For each period in which a new block starts in the FSGS solution, an interval start period is encoded in B. The heuristically determined resource profiles are not used.

6.2.4 Improvement Operators

A valid MP solution can result in an infeasible resource profile with outsourcing. The resource allocation may exceed the availability because too many activities are scheduled in parallel or the activity durations are too short. An MP solution may also result in a resource profile with slacks if activity durations are longer than actually required. To locally improve such solutions, two operators are proposed.

6.2.4.1 Reduction Operator

The purpose of this operator is to improve the makespan by reducing activity durations. It is only applied to top solutions because by reducing their already low makespans, a new best solution is more likely to be obtained. The operator calculates for each activity i with $d_i > \underline{d}_i$ the slack of each required resource r:

$$\sum_{t=s_i}^{c_i} \max(b_r - \sum_{\substack{j \in V \\ z_{jt}=1}} q_{jrt}, 0) \tag{6.42}$$

The operator then checks by how many periods the duration of the activity may be reduced if the slacks are additionally allocated. Of course, a reduction is only possible if all required resources suffice. From the activities that may be reduced by at least one period, the operator randomly selects one activity and reduces its duration by the identified number of periods. To obtain the new resource allocation, the RSP is solved to optimality. The SP is not applied because Benders cuts are not required. If the resulting resource profile features outsourcing, for example due to the minimum block length, the modified solution is discarded and the original is maintained.

6.2.4.2 Repair Operator

The purpose of this operator is to repair infeasible solutions by eliminating outsourcing. As its goal is to obtain feasible solutions with low makespans, it is only applied to the top solutions due to their already low makespans. The operator increases activity durations or reduces the number of parallel activities by inserting disjunctive arcs.

Following the interval definition from Section 6.1.3, the operator identifies groups of contiguous intervals with outsourcing for principal or independent resource r. In each such group, at least one activity that requires r has to be constantly active in all intervals of the group. Let us denote the interval start period of the first interval in the group as t_1 and the last period of the last interval as t_2. Then the operator randomly applies one out of two heuristics.

- **Increase activity duration**: An activity is considered as a candidate if it is active in all intervals of the group and if increasing its duration, and thus reducing the allocated resource quantity per period, may eliminate the outsourcing. Two necessary conditions have to be fulfilled for each resource r:

$$\sum_{t=t_1}^{t_2} (q_{irt} - \underline{q}_{ir}) \geq \sum_{t=t_1}^{t_2} o_{rt} \tag{6.43}$$

$$d_i + \left\lceil \sum_{t=t_1}^{t_2} \frac{o_{rt}}{\overline{q}_{ir}} \right\rceil \leq \overline{d}_i \tag{6.44}$$

  Condition (6.43) ensures that a reduction of the currently allocated resource quantity q_{irt} can compensate the outsourcing. Condition (6.44) ensures that the minimum activity duration that is required to compensate the outsourcing does not exceed the upper bound of activity duration.

From the identified candidates, the operator randomly selects one activity and sets its duration to a random value between (6.45) and (6.46). The first number (6.45) is the minimum required duration to compensate the outsourcing. It assumes that a resource quantity equal to the upper resource usage bound can be allocated. The second number (6.46) is the maximum required duration. It assumes that only a quantity equal to the minimum resource usage bound can be allocated:

$$d_i + max_{r \in R_i}\Big(\Big\lceil \sum_{t=t_1}^{t_2} \frac{o_{rt}}{\overline{q}_{ir}} \Big\rceil\Big) \tag{6.45}$$

$$min\Big(\overline{d}_i, d_i + max_{r \in R_i}\Big(\Big\lceil \sum_{t=t_1}^{t_2} \frac{o_{rt}}{\underline{q}_{ir}} \Big\rceil\Big)\Big) \tag{6.46}$$

- **Insert disjunctive arc**: From the set of activities that are active in all intervals of the group, the operator randomly selects one activity j. It inserts a disjunctive arc $(i \rightarrow j)$ from another activity i that is active in the last interval of the group and completes in a period greater than or equal to t_2. As a result, the start of activity j is delayed after period t_2.

In both cases, the RSP is solved to obtain a new resource profile. If the resulting solution still features outsourcing, the procedure is repeated until the given maximum number of iterations is reached.

6.2.4.3 Example

Assume that the repair operator processes the solution from Figure 6.5. The operator identifies only one interval with outsourcing from periods $t_1 = 10$ to $t_2 = 11$. Two operations are possible to repair the resource infeasibility:

- **Increase activity duration**: Activities 4 and 5 are active in periods 10 and 11 and fulfill the required conditions (6.43) and (6.44). The operator randomly selects activity 5. As terms (6.45) and (6.46) both equal to 3 periods, activity 5's duration is increased to 3 periods.

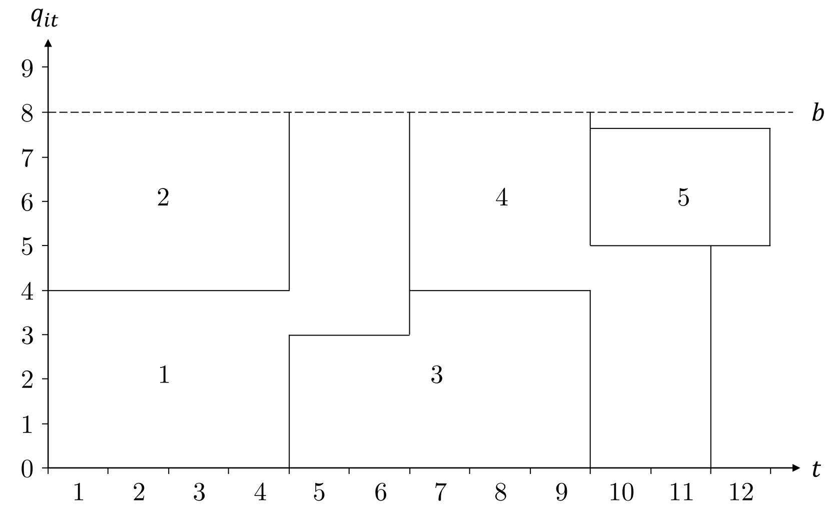

Figure 6.6: Resource profile after applying the repair operator on the solution from Figure 6.5

- **Insert disjunctive arc**: Either disjunctive arc $(4 \rightarrow 5)$ or $(5 \rightarrow 4)$ may be inserted, since both activities 4 and 5 are active during the whole interval and complete in period $t_2 = 11$.

Figure 6.6 illustrates the resulting resource profile for the case that the duration of activity 5 is increased from 2 to 3 periods.

6.2.5 Genetic Operators

The crossover operator creates two child solutions from two parent solutions in two steps. First, the activities of each parent solution are sorted according to their start periods in ascending order. Activities with similar or equal start periods are close to each other in the resulting activity sequence. Second, the partially-matched two-point crossover of Hartmann (1998) is applied on the sorted activity sequences. The encoding (A, B, D) is modified according to the crossover of the activities in the sequence. The benefits of this two-step approach are increased locality and the recombination of contiguous partial schedules.

The probabilities of the mutation operators are provided in the implementation details in Section 7.3. Disjunctive arcs are mutated by iterating through list A and modifying each entry j with a certain probability. A new random disjunctive arc $(i \to j)$ is inserted by randomly selecting an activity i from the set of all activities that do not have direct or transitive precedence relations from set $\mathcal{E}_{\mathcal{C}}$ with j. Additionally, no incoming disjunctive arc may be set with equal probability.

Each activity duration $d_i \in D$ is modified with a certain probability by either increasing or decreasing it by 1 period with equal probability. d_i is kept within the bounds $\underline{d}_i$ and $\overline{d}_i$.

For each B_{ir} in B, one random interval start period is added with a certain probability. A distance of at least l_{ir} to the start period and of at least $l_{ir} - 1$ to the completion period of the activity is maintained. If there is already an interval start period in the designated period, it is removed. Other interval start periods with a distance of less than l_{ir} to the newly inserted one are also removed.

Chapter 7

Computational Study

This chapter presents the computational study. The study design is outlined in Section 7.1, the test data is described in Section 7.2, and implementation details are provided in Section 7.3. Finally, the results are analyzed and interpreted in Section 7.4.

7.1 Study Design

The computational study evaluates the performance of six methods:

- **HM**: The hybrid metaheuristic from Chapter 4.
- **BGA**: The Benders genetic algorithm from Chapter 6.
- **RGA**: The reduced subproblem genetic algorithm from Chapter 6.
- **SGA**: The self-adaptive genetic algorithm from Chapter 5.
- **PRS**: As a benchmark, a parallel random sampling heuristic is employed. It resembles the method of Kolisch et al. (2003). The PRS constructs activity lists by random sampling and generates solutions with the SGA's standard parallel SGS.
- **SRS**: The serial random sampling heuristic is the same as the PRS but uses the SGA's standard serial SGS.

The methods are compared based on a maximum number of generated solutions per problem instance. This widely accepted methodology has previously been applied in studies on the RCPSP (Hartmann and Kolisch, 2000; Kolisch and Hartmann, 2006) and on the MRCPSP (van Peteghem and Vanhoucke, 2014). According to Kolisch and Hartmann (2006), this termination criterion has the advantages that it is platform independent and allows direct comparison with future studies. Limits of 1,000, 5,000, 15,000 and 25,000 solutions are used. If an SGS is applied, each started schedule generation process is counted once. This also applies to the HM's VNS. For the BGA and the RGA, each generated MP solution is counted regardless of whether the SP or the RSP are solved or not. Solutions resulting from the improvement operators are counted as well. If a method determines a makespan equal to T_{min}, the instance is solved to optimality and the method terminates immediately.

As a reference to compare the other methods, the results of MIP model FP-DT3 from Naber and Kolisch (2014b) are used. The authors considered the best solution obtained after a time limit of 2 hours of single-threaded CPU time per problem instance.

Before solving a problem instance, the preprocessing of Naber and Kolisch (2014b) is applied. It removes redundant precedence relations, tightens resource usage bounds, revises resource requirements, calculates the earliest start and completion periods, and derives the lower bound of the makespan T_{min} from Chapter 2. Latest start and completion periods and an upper bound of the makespan are not calculated because the metaheuristics obtain these values.

7.2 Test Data

The computational study is conducted on 2,909 problem instances from test sets A and B of Fündeling and Trautmann (2010). Details on the generation of the instances are provided in Fündeling (2006).

Test set A contains instances with up to 4 resources, derived by Fündeling (2006) from the RCPSP instance sets J30, J60 and J90 of the PSPLIB (Kolisch and Sprecher, 1997). The study only includes instances of test set A with at most 55 activities because for larger instances no MIP results are available.

The remaining 509 instances of test set A are denoted as instance set $A_{\leq 55}$. The subscript indicates the number of activities.

Test set B consists of instance sets B_{10}, B_{20}, B_{40}, B_{100}, and B_{200} with 10, 20, 40, 100, and 200 activities, respectively, and up to 4 resources. Fündeling and Trautmann (2010) generated 480 problem instances in each set by using a factorial design of the problem parameters order strength, resource factor, and resource strength. Demeulemeester and Herroelen (2002) define the order strength (OS) as the number of existing precedence relations divided by the maximum possible number of precedence relations. The higher the OS is, the more precedence relations are in the project network and the more sequential is the resulting schedule. OS values of 0.25, 0.5, and 0.75 were used. The resource factor (RF) indicates the number of required resources per activity (Kolisch and Sprecher, 1997). Its values were set to 0.25, 0.5, 0.75, and 1. For example, a value of 0.5 indicates that each activity requires 2 out of the 4 resources. The resource strength (RS) measures the scarcity or constrainedness of resources by comparing the resource requirements to the resource availability (Kolisch and Sprecher, 1997). RS values of 0, 0.25, 0.5, and 0.75 were used. A lower value indicates a higher scarcity. For RS = 0, there is, according to Fündeling and Trautmann (2010), for each resource at least one activity that may exclusively occupy the resource due to the upper bound of resource usage $\overline{q}_{ir}$. For RS = 1, the resource availability does not constrain the scheduling. The minimum block length was randomly assigned to values between 2 and 4 periods.

All activities of a problem instance feature the same minimum block length and require the same principal resource k. To ensure that a feasible solution exists, the resource function coefficients are set to $\alpha_{ir} = (\overline{q}_{ir} - \underline{q}_{ir})/(\overline{q}_{ik} - \underline{q}_{ik})$ and $\beta_{ir} = \underline{q}_{ir} - \underline{q}_{ik}\alpha_{ir}$ for each dependent resource r, as in the study of Naber and Kolisch (2014b).

7.3 Implementation

The study is conducted on a desktop PC with a 3.3GHz Intel Core i3-2120 CPU and 16GB RAM running the 64-bit edition of Windows 8. Elapsed time for a single-threaded execution of each method is considered. All methods are implemented in Java 8. The parameters of the metaheuristics have been determined in a preliminary study. No parameter tuning software or automated algorithm configurators have been used.

To adapt the HM's GA to different problem sizes, its population is defined by the function $\min(10 \cdot n, 400)$ of the number of activities n. The mutation rate for the activity list λ is 5%, as suggested by Hartmann (1998). The same mutation rate is also applied for the resource allocation limit list ρ. For the start delay list σ, a low mutation rate 0.5% is used. As the GA and the VNS are sequentially executed, their interplay is facilitated by the number of solutions assigned as termination criterion to each method. The VNS solution limit is set to $F_{VNS} = \lfloor F \cdot \min(n/200, 0.25) \rfloor$ solutions with F denoting the overall solution limit. The GA solution limit of $F_{GA} = F - F_{VNS}$ is rounded down to the closest multiple of the population size in order to ensure that the GA always generates full generations. The remaining solutions are added to F_{VNS}. The following limits apply for the VNS: 1,000 non-improving solutions per incumbent solution, 200 generated solutions per neighborhood, a maximum neighborhood of 5, and 5 longest paths in the activity selection.

The SGA's population size is set to $\min(5 \cdot n, 200)$. In line with Hartmann (2002), mutation rates for the activity list and for the SGS flag of 5% are used.

The BGA and the RGA have a population size of 500. Good results have been obtained by using a 3% probability for the mutation of the disjunctive arcs in A and 5% for the mutation of the durations in D and blocks in B. Moreover, the repair operator is limited to at most 5 iterations per solution and the reduction operator to a single iteration. The size of the cut pool is limited to 1,000. The limit is maintained by removing cuts on a first-in/first-out basis. Solutions that exceed the BGA's and RGA's upper limit for the makespan are penalized to $(C_{max})^2$ without solving the SP or RSP.

The SP and the RSP are solved by IBM ILOG CPLEX 12.6.2 integrated through the Concert Technology Java API. The SP is implemented as a single LP in which only the constraint right-hand-sides are subsequently modified each time it is solved for a new MP solution. For the RSP, a new LP is built for each evaluated MP solution. Since LP columns and rows change significantly depending on the intervals defined by z_{it} and δ_{irt}, completely rebuilding the model in one operation is mostly faster than performing multiple small modifications. Both models are solved single-threaded with activated presolve. The penalty factor for outsourcing is $\pi_o = 10$. The factor for resource allocation is $\pi_p = 10^{-4}$ for instances with fewer than 100 activities. For larger instance, it is set to $\pi_p = 10^{-5}$ because the weighted sum of allocated resource quantities would otherwise exceed 1 (see Section 6.1.2).

Naber and Kolisch (2014b) solved their MIP model FP-DT3 with CLPEX on a faster computer with a 3.4GHz Intel Core i7-3770 CPU and 16GB RAM.

7.4 Results

Whenever stating statistical significance, the $\alpha = 0.05$ significance level applies. The statistical significance of the difference in the results is tested by a Kruskal-Wallis one-way analysis of variance. For pairwise comparisons, Mann-Whitney U tests with Bonferroni correction are used as post-hoc analysis. Non-parametric statistical tests are applied instead of their parametric counterparts because normal distribution and homogeneity of variances are not always given.

The results are presented throughout the next sections. In Section 7.4.1, the methods' solution quality is analyzed on all instances. In Section 7.4.2, the analysis is focused on a subsample of hard problem instances in order to better differentiate the methods. Sections 7.4.4 and 7.4.5 provide an assessment of the metaheuristics' components. The required computation times are compared in Section 7.4.6. Finally, the models of the model-based metaheuristics are evaluated in Section 7.4.7.

7.4.1 Solution Quality: All Instances

A summary of the best results obtained is presented in Figure 7.1. The figure visualizes the average gap to the lower bound of the makespan T_{min} for the highest solution limit of 25,000 generated solutions per problem instance of sets B_{10} to B_{200}. The results of the BGA and the RGA as well as of the PRS and the SRS are shown as combined averages.

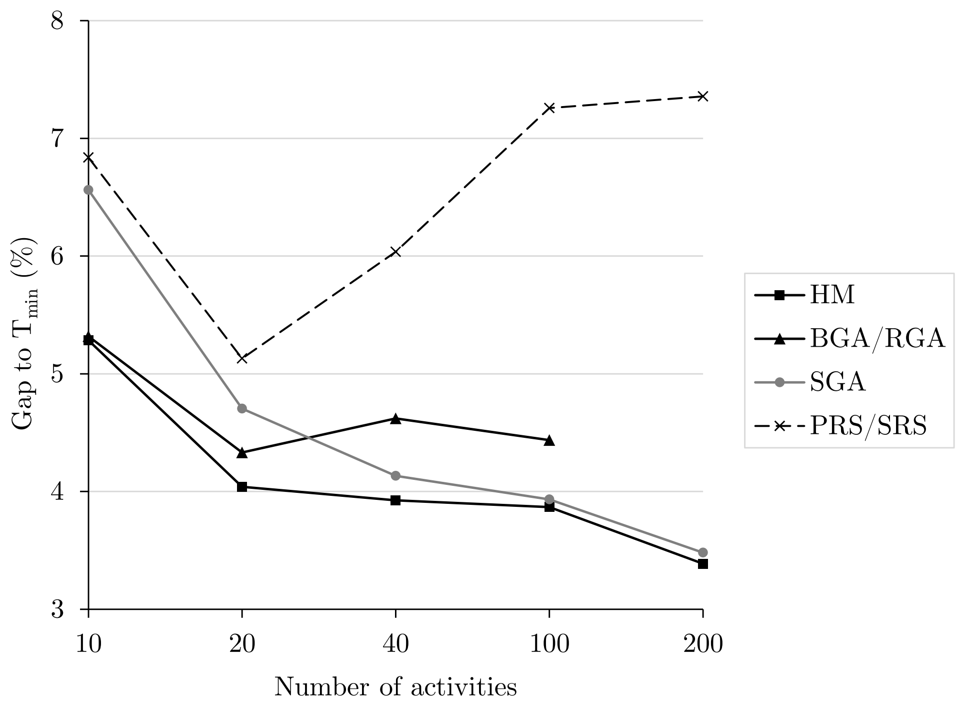

Figure 7.1: Best results obtained

More details on the gap to T_{min} are provided in Table 7.1. In addition, Table 7.2 lists the average deviation from the best MIP solution. Optimal MIP solutions are available for all instances of set B_{10}, whereas for sets B_{20}, B_{40}, and $A_{\leq 55}$ not all MIP solutions are optimal. No MIP results are available for instance sets B_{100} and B_{200}. An optimal solution may also feature a gap to T_{min}. Both tables show results for each instance set with limits of 1,000 to 25,000 generated solutions as well as the average result across all limits in bold. The last bold row of each table provides the overall results across all instance sets and solution limits. The difference in the results is statistically significant in all bold table rows. The model-based metaheuristics are only compared up to 100 activities because the BGA's computation time becomes restrictively high.

Let us start with the HM. It yields better overall results than all other methods with statistical significance. The HM achieves the largest advantages over the other SGS-based methods on the small instance sets. On set $A_{\leq 55}$, the HM's average gap to T_{min} is more than 2 percent points lower than the SGA's gap. For set B_{10}, the values reported in Table 7.2 constitute the optimality gaps,

Table 7.1: Average gap to T_{min} in percent

Set / Sol.	HM	BGA	RGA	SGA	PRS	SRS
$\mathbf{A_{\leq 55}}$	**5.60**	**6.04**	**5.91**	**7.82**	**8.41**	**9.83**
1,000	6.27	6.25	6.22	8.30	9.09	10.68
5,000	5.58	6.07	6.01	7.83	8.45	9.89
15,000	5.30	5.96	5.75	7.63	8.10	9.47
25,000	5.25	5.88	5.67	7.53	7.98	9.28
$\mathbf{B_{10}}$	**5.40**	**5.44**	**5.46**	**6.57**	**6.81**	**6.88**
1,000	5.62	5.73	5.72	6.61	6.83	6.91
5,000	5.38	5.40	5.43	6.56	6.82	6.87
15,000	5.31	5.32	5.35	6.56	6.81	6.87
25,000	5.28	5.30	5.34	6.56	6.81	6.87
$\mathbf{B_{20}}$	**4.24**	**4.49**	**4.48**	**4.77**	**5.34**	**5.40**
1,000	4.60	4.71	4.72	4.90	5.73	5.84
5,000	4.23	4.52	4.51	4.78	5.37	5.39
15,000	4.10	4.39	4.37	4.71	5.16	5.23
25,000	4.04	4.35	4.31	4.70	5.10	5.16
$\mathbf{B_{40}}$	**4.08**	**4.71**	**4.67**	**4.29**	**6.21**	**6.74**
1,000	4.35	4.79	4.78	4.55	6.84	7.44
5,000	4.09	4.72	4.70	4.31	6.26	6.85
15,000	3.96	4.68	4.62	4.18	5.94	6.41
25,000	3.93	4.66	4.58	4.13	5.81	6.26
$\mathbf{B_{100}}$	**3.94**	**4.45**	**4.45**	**4.05**	**7.15**	**8.09**
1,000	4.07	4.47	4.47	4.21	7.68	8.68
5,000	3.94	4.45	4.45	4.08	7.21	8.14
15,000	3.89	4.45	4.44	3.97	6.92	7.84
25,000	3.87	4.44	4.43	3.93	6.80	7.71
$\mathbf{B_{200}}$	**3.41**			**3.55**	**7.12**	**8.20**
1,000	3.46			3.65	7.53	8.67
5,000	3.40			3.57	7.16	8.27
15,000	3.39			3.51	6.95	7.98
25,000	3.39			3.48	6.83	7.88
Overall	**4.46**	**5.04**	**5.00**	**5.20**	**6.86**	**7.55**

Table 7.2: Average deviation from the best MIP solution in percent (negative values are better)

Set / Sol.	HM	BGA	RGA	SGA	PRS	SRS
$\mathbf{A_{\leq 55}}$	**0.03**	**0.41**	**0.30**	**2.12**	**2.64**	**3.89**
1,000	0.64	0.61	0.57	2.52	3.26	4.66
5,000	0.01	0.44	0.38	2.13	2.67	3.95
15,000	-0.24	0.34	0.15	1.95	2.37	3.57
25,000	-0.27	0.26	0.09	1.88	2.26	3.40
$\mathbf{B_{10}}$	**0.24**	**0.27**	**0.29**	**1.31**	**1.55**	**1.59**
1,000	0.44	0.55	0.54	1.34	1.56	1.62
5,000	0.22	0.24	0.27	1.30	1.55	1.58
15,000	0.15	0.16	0.19	1.30	1.55	1.58
25,000	0.14	0.15	0.18	1.30	1.55	1.58
$\mathbf{B_{20}}$	**0.28**	**0.50**	**0.49**	**0.75**	**1.28**	**1.31**
1,000	0.60	0.71	0.71	0.86	1.63	1.70
5,000	0.27	0.53	0.52	0.75	1.31	1.30
15,000	0.15	0.41	0.39	0.70	1.12	1.15
25,000	0.10	0.37	0.34	0.69	1.07	1.09
$\mathbf{B_{40}}$	**-1.88**	**-1.34**	**-1.38**	**-1.73**	**-0.05**	**0.32**
1,000	-1.66	-1.27	-1.28	-1.53	0.49	0.93
5,000	-1.88	-1.33	-1.36	-1.72	-0.01	0.42
15,000	-1.98	-1.38	-1.43	-1.82	-0.29	0.04
25,000	-2.01	-1.39	-1.47	-1.85	-0.40	-0.09
Overall	**-0.33**	**-0.03**	**-0.07**	**0.63**	**1.37**	**1.81**

since all MIPs were solved to optimality. On this instance set, the HM achieves for a limit of 25,000 generated solutions an optimality gap of 0.14%, which is more than 9 times lower than the SGA's gap. This relates to optimal solutions for 95% of all instances. The HM further improves the makespan on set B_{10} when increasing the solution limit from 15,000 to 25,000 solutions, whereas all other SGS-based methods are stagnant or likely trapped to suboptimality. With growing instance size, the difference to the SGA becomes smaller, though. For the medium instance set B_{40}, the HM finds many new best-known solutions that have not been determined by the MIP. For the large sets B_{100} and B_{200}, the average gaps to T_{min} of the HM and the SGA decrease, indicating good search performance, whereas they increase for the random sampling heuristics.

The model-based metaheuristics BGA and RGA produce similar results with statistically insignificant differences on all instance sets. Hence, the BGA's approach of approximating the fitness results in a competitive solution quality, while effectively reducing the number of SPs to solve. On instance set B_{10}, the results are nearly identical to the results of the HM. Both model-based methods also further improve the makespan from 15,000 to 25,000 generated solutions. In contrast to the SGS-based methods, the BGA and the RGA may also generate infeasible solutions with resource profiles that feature outsourcing. While the results are similar on small instances, the HM obtains significantly better results on the larger instance sets B_{40} and B_{100}. On these larger instances, the infeasible resource profiles seem to hinder the search progress.

The HM, the BGA, and the RGA expand the search space when compared to the SGA that only employs standard SGSs. This expansion leads to statistical significant improvements on the small instance set B_{10}, where the SGA behaves similarly to the random sampling heuristics. Overall, all four metaheuristics provide significantly better results than the heuristics, especially for large problem instances.

The PRS and the SRS perform best on instance sets B_{20} and B_{40}. For smaller problem instances, the applied standard SGSs are the limiting factor to obtain better solutions. If a resulting makespan in instance set B_{10} is just 1 period above the optimum, the relative gap to T_{min} is much bigger than in instance sets B_{20} and B_{40}. For the largest instances in sets B_{100} and B_{200}, the mainly random search leads to worse results due to the sheer number of possible activity list permutations and the fixed limit on the number of generated solutions.

Regarding the overall results from Table 7.1, the methods are ranked from best to worst as follows. The relation $<$ indicates that the first method has a lower gap to T_{min} than the second method:

$$\text{HM} < \text{RGA} < \text{BGA} < \text{SGA} < \text{PRS} < \text{SRS} \tag{7.1}$$

A slightly different ranking results if statistical significance is considered. Let relation $\prec$ indicate that the first method's gap to T_{min} is with statistical significance lower than the second's one. If the difference is statistically insignificant, the relation $=$ applies. The resulting ranking from best to worst is:

$$\text{HM} \prec \text{RGA} = \text{BGA} = \text{SGA} \prec \text{PRS} = \text{SRS} \tag{7.2}$$

7.4.2 Solution Quality: Hard Instances

The instance sets contains numerous easy problem instances for which both the standard serial and the standard parallel SGS obtain a makespan equal to T_{min} after a single run with the LPF priority rule. The instance sets B_{10} to B_{200} contain 36–42% easy instances, whereas the PSPLIB instances in set $A_{\leq 55}$ only contain 5%. The HM and the SGA solve all easy instances to optimality. Similarly, the BGA and the RGA find optimal solutions for all except two instances. Since the performance of the metaheuristics can hardly be differentiated on the easy instances, the instances are excluded from the analysis presented in this section. The remaining 1,950 problem instances are considered as hard problem instances. They are distributed across set $\text{A}'_{\leq 55}$ with 485 problem instances, as well as sets B'_{10} with 287, B'_{20} with 279, B'_{40} with 309, B'_{100} with 292, and B'_{200} with 298 problem instances.

Table 7.3 provides the average gap to T_{min} for the hard problem instances. Its structure is identical to the tables in the previous section. Now the difference in the results is statistically significant in all table rows. The most remarkable change in the results are the larger gaps to T_{min} for all methods. As this is also observed for the optimal MIP solutions of set B'_{10}, the hard instances seem to simply feature a larger gap between the optimal makespan and T_{min}. The widened spread between the metaheuristics and heuristics is clearly due to the search performance. Especially on the largest instances, the HM obtains on average up to 7 percent points lower gaps to T_{min}.

Excluding the easy instances does not change the relative performance of the methods. Hence, rankings (7.1) and (7.2) remain the same for the hard instances.

Table 7.3: Hard problem instances: Average gap to T_{min} in percent

Set / Sol.	HM	BGA	RGA	SGA	PRS	SRS
$\mathbf{A'_{\leq 55}}$	**5.88**	**6.34**	**6.21**	**8.21**	**8.82**	**10.32**
1,000	6.58	6.56	6.52	8.71	9.54	11.21
5,000	5.86	6.37	6.31	8.22	8.86	10.38
15,000	5.56	6.26	6.04	8.00	8.50	9.94
25,000	5.51	6.17	5.96	7.91	8.38	9.74
$\mathbf{B'_{10}}$	**9.03**	**9.09**	**9.13**	**10.99**	**11.40**	**11.50**
1,000	9.39	9.57	9.55	11.05	11.42	11.56
5,000	9.00	9.03	9.08	10.97	11.40	11.48
15,000	8.87	8.90	8.95	10.97	11.39	11.48
25,000	8.84	8.86	8.93	10.97	11.39	11.48
$\mathbf{B'_{20}}$	**7.30**	**7.72**	**7.70**	**8.21**	**9.18**	**9.30**
1,000	7.91	8.09	8.09	8.43	9.85	10.04
5,000	7.28	7.77	7.76	8.22	9.24	9.27
15,000	7.05	7.55	7.52	8.10	8.87	9.00
25,000	6.95	7.48	7.42	8.09	8.77	8.88
$\mathbf{B'_{40}}$	**6.34**	**7.32**	**7.26**	**6.67**	**9.65**	**10.47**
1,000	6.75	7.43	7.42	7.07	10.61	11.54
5,000	6.35	7.34	7.30	6.69	9.73	10.64
15,000	6.16	7.27	7.18	6.49	9.22	9.95
25,000	6.10	7.24	7.12	6.42	9.03	9.73
$\mathbf{B'_{100}}$	**6.47**	**7.32**	**7.30**	**6.65**	**11.74**	**13.30**
1,000	6.68	7.34	7.34	6.92	12.60	14.25
5,000	6.47	7.31	7.30	6.70	11.83	13.38
15,000	6.39	7.31	7.29	6.53	11.37	12.89
25,000	6.36	7.30	7.28	6.47	11.18	12.67
$\mathbf{B'_{200}}$	**5.49**			**5.72**	**11.44**	**13.20**
1,000	5.57			5.88	12.10	13.94
5,000	5.48			5.74	11.51	13.30
15,000	5.47			5.66	11.18	12.85
25,000	5.46			5.61	10.99	12.69
Overall	**6.65**	**7.41**	**7.36**	**7.76**	**10.22**	**11.26**

7.4.3 Instance Parameters

Let us now asses the influence of the instance parameters on the solution quality. Figures 7.2 to 7.5 visualize the gap to T_{min} for different values of order strength (OS), resource factor (RF), resource strength (RS), and minimum block length. The averages across all solution limits for instance sets B_{10} to B_{200} are shown.

All methods are relatively robust against changes in the OS and produce similar results for different network structures. For example, the results of the HM only change within a range of 0.5 percent points. In line with the results reported by Fündeling (2006), instances with fewer precedence relations and more resulting scheduling possibilities are not significantly harder to solve. In the classic RCPSP, the resource constraints often prevent activities that are not connected by precedence relations to be scheduled in parallel. In the FRCPSP, given that the lower bounds of resource usage are sufficiently low in relation to the resource availability, such activities can actually be scheduled in parallel, likely resulting in a high resource utilization.

The parameters RF and RS have a much higher impact on the results. Problem instances in which activities require a high number of resources or resources are scarce, are either more difficult to solve or the difference between the optimal makespan and T_{min} is higher. Especially if resources are relatively scarce (RS $= 0$), the gap to the lower bound noticeably increases. In this setting, the metaheuristics produce significantly better results than the heuristics with up to 9 percent points lower gaps to T_{min}. If resources are nearly abundant (RS $= 0.75$), all methods solve the majority of instances to optimality. A lower minimum block length just leads to slightly better results. For example, reducing the minimum block length from 4 to 2 periods lets the HM's gap to T_{min} decrease by just 0.6 percent points. Similar as with the OS, the choice of the resource usage bounds may mitigate the effects of the minimum block length.

In summary, the methods show similar behavior for changing instance parameters. Clear differences are only visible for the case of RS $= 0$, in which the metaheuristics perform better.

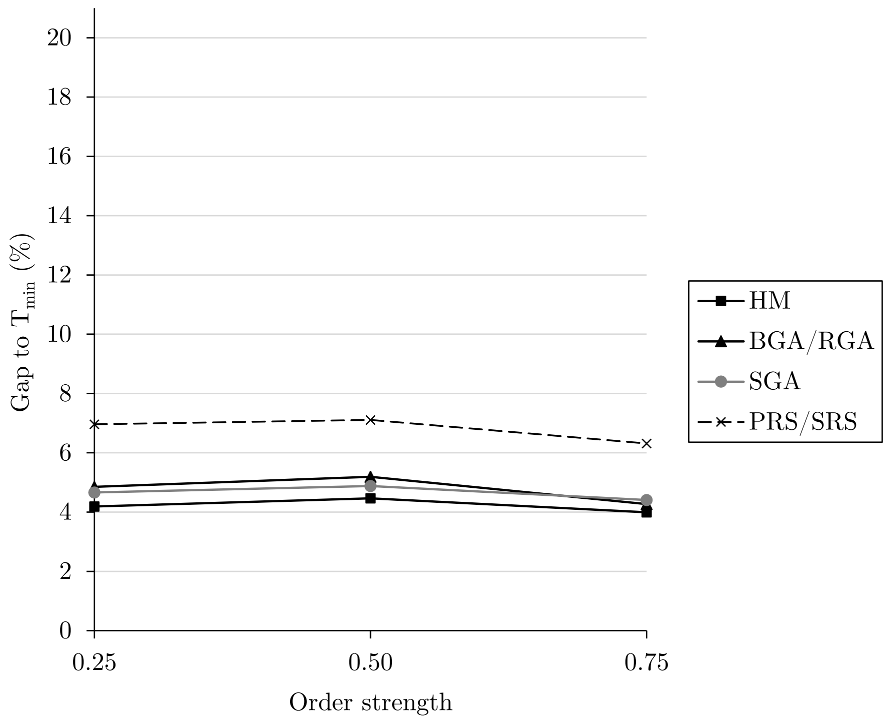

Figure 7.2: Influence of the order strength

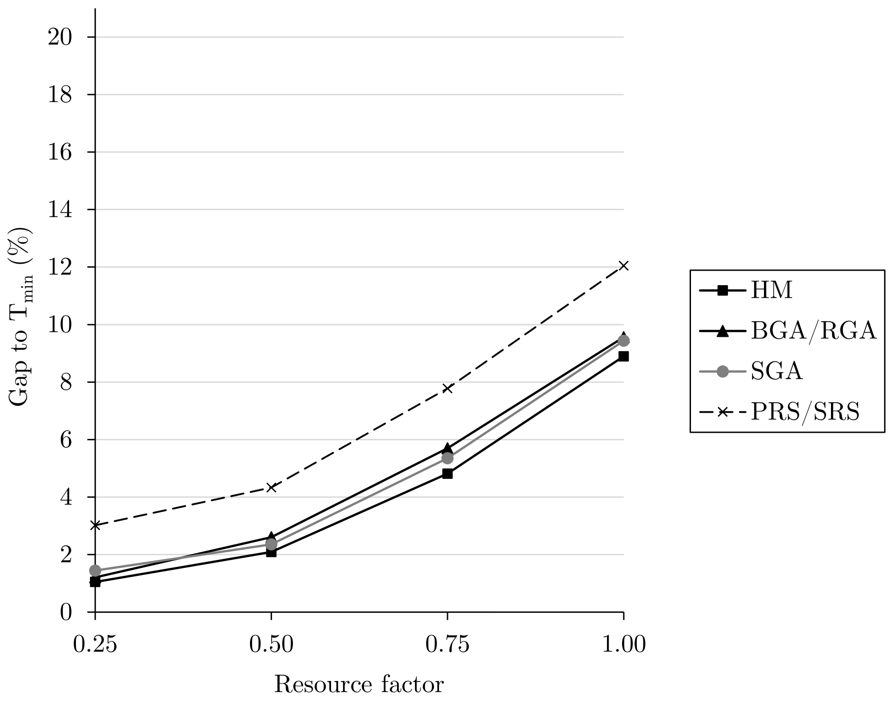

Figure 7.3: Influence of the resource factor

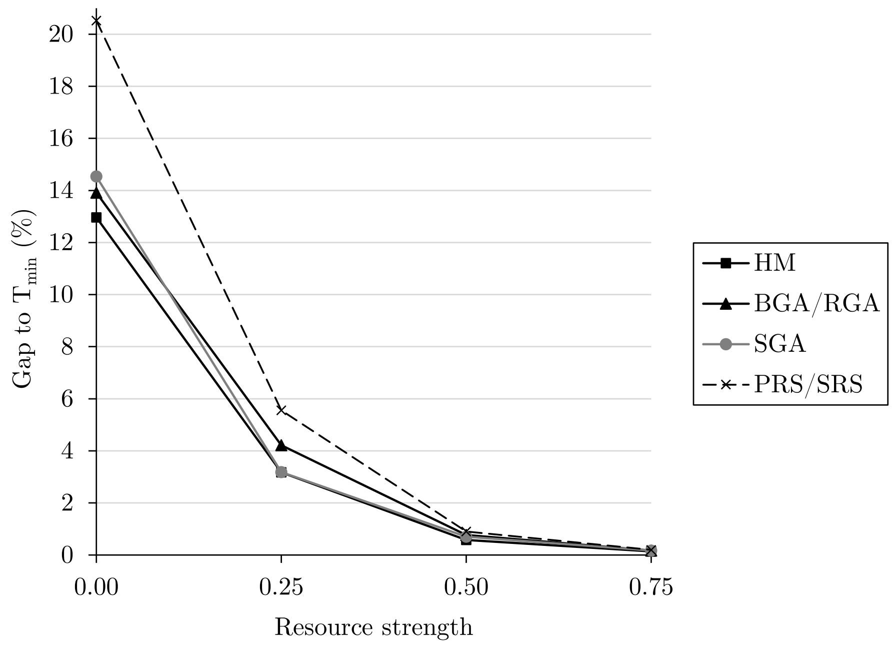

Figure 7.4: Influence of the resource strength

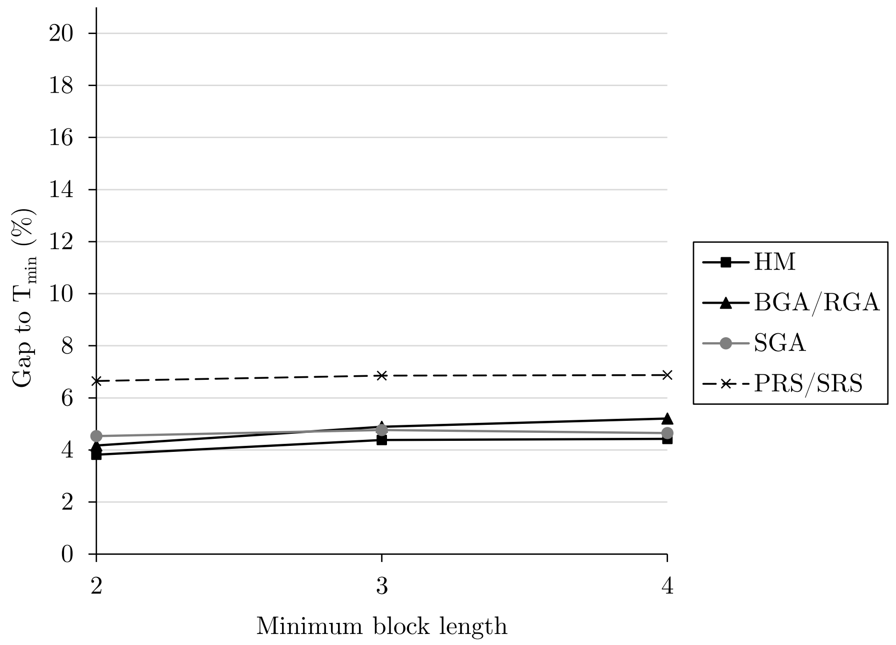

Figure 7.5: Influence of the minimum block length

Table 7.4: HM components: Average gap to T_{min} in percent

Set	HM	GA-FSGS	PR-FSGS	GA-SGS
$A_{\leq 55}$	5.60	5.69	6.92	7.88
B_{10}	5.40	5.41	5.60	6.85
B_{20}	4.24	4.28	5.02	5.09
B_{40}	4.08	4.17	5.84	4.54
B_{100}	3.94	4.09	5.23	4.16
B_{200}	3.41	3.58	4.24	3.69
Overall	**4.46**	**4.55**	**5.49**	**5.39**

7.4.4 Components of the Hybrid Metaheuristic

To assess the impact of the HM components on solution quality, four combinations are compared:

- **HM**: The complete hybrid metaheuristic.
- **GA-FSGS**: The FSGS is embedded into the GA without the VNS. This combination is used to evaluate the impact of the VNS.
- **PR-FSGS**: The FSGS is embedded into a multi-start priority rule heuristic. The purpose of this combination is to assess the GA's impact. λ is determined by the priority rules LPF, MTS, and MWR as well as random selection, similar as in Fündeling and Trautmann (2010). ρ and σ are determined randomly within their bounds.
- **GA-SGS**: The GA operates only on λ and is combined with the SGA's standard parallel SGS. This combination is used to evaluate the FSGS.

Table 7.4 lists the gap to T_{min} averaged across all solution limits. The difference in the results is statistically significant in all table rows. Considering the overall results, the GA-FSGS generates better results than the PR-FSGS and the GA-SGS with statistical significance. Indeed, non-greedy resource allocation and delayed scheduling result in significantly better solutions. Also the GA leads to a significantly higher solution quality. Similar results with larger differences between the methods but the same statistical significances

Table 7.5: VNS: Improved instances (Inst. %) and makespan reduction per improved instance in periods (ΔC_{max}) and in percent ($\Delta C_{max}\%$)

Set	Inst. %	ΔC_{max}	$\Delta C_{max}\%$
$A_{\leq 55}$	5.59	1.06	2.27
B_{10}	1.56	1.06	2.44
B_{20}	7.57	1.23	1.39
B_{40}	20.63	1.34	0.85
B_{100}	54.24	2.12	0.57
B_{200}	61.84	3.84	0.55
Overall	**22.91**	**2.58**	**1.34**

are also obtained on the subset of hard problem instances. The improvements of the complete HM compared to the GA-FSGS are statistically insignificant when considering all instances. However, this analysis also includes instances on which the VNS is not even executed. Since the VNS operates within the HM after completion of the GA, the VNS is not started in case the GA has already solved an instance to optimality.

The potential of the VNS becomes visible when investigating only the instances on which the VNS is actually executed. The question is: Given the same solution limit, is it better to apply the VNS or to only use the GA? Table 7.5 lists the percentage of instances for which the HM including VNS yields a lower makespan than the GA-FSGS (Inst. %). The larger the problem size, the more instances are improved by the VNS, peaking at 62% on instance set B_{200}. Note that the VNS solution limit F_{VNS} is also highest on this instance set. On average, the VNS further improves 23% of the instances. For the sake of completeness, this number corresponds to 12% of all instances, however, then also including instances on which the VNS is not executed. Table 7.5 also provides the absolute makespan reduction per improved instance in periods (ΔC_{max}) and in percent of the GA-FSGS's makespan ($\Delta C_{max}\%$). Although the absolute makespan reduction in periods increases with the problem size, the relative makespan reduction declines because larger problem instances also feature higher makespans.

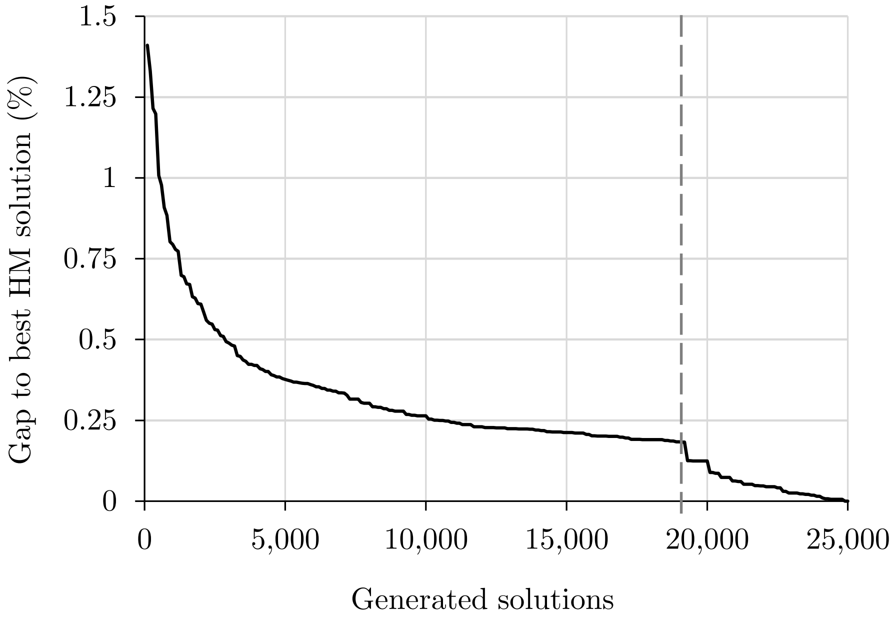

Figure 7.6: Convergence of the HM

Next, the HM solution representation is analyzed. The HM features on average in 59% of its best solutions a resource allocation limit $\rho_i > 0$ and in 25% a start delay $\sigma_i > 0$ for at least one activity. Consider again that the resource allocation is not necessarily reduced for the whole duration of an activity and that the start delay is relative to the period in which the FSGS would otherwise schedule the activity. Nevertheless, the results indicate that the HM actually uses the FSGS features in the search process.

Finally, the convergence behavior of the HM is illustrated in Figure 7.6. The x-axis is the number of generated solutions and the y-axis is the average gap to the best solutions found. The figure visualizes the average for a limit of 25,000 generated solutions on instance sets B_{10} to B_{200}. Since the VNS's start is determined by its variable solution limit F_{VNS}, the vertical dashed line indicates the earliest start at 18,000 generated solutions. Left of the line, only the GA operates and shows a typical convergence behavior. Right of the line, the VNS noticeably improves the results but its improvement rate declines over time, similar as in the GA. The preliminary study indicated that earlier starts of the VNS do not further improve the shown results.

Table 7.6: Improvement operators: Average gap to T_{min} in percent

Set	Operators	No operators
$A_{\leq 55}$	5.86	6.12
B_{10}	5.33	5.48
B_{20}	4.38	4.58
B_{40}	4.65	4.77
B_{100}	4.44	4.46
Overall	**4.94**	**5.10**

Table 7.7: Improvement operators: Improved instances (Inst. %) and makespan reduction per improved instance in periods (ΔC_{max}) and in percent (ΔC_{max}%)

Set	Inst. %	ΔC_{max}	ΔC_{max}%
$A_{\leq 55}$	12.80	1.07	3.26
B_{10}	10.40	1.15	2.63
B_{20}	27.65	1.26	1.53
B_{40}	21.91	1.53	1.02
B_{100}	10.04	1.23	0.40
Overall	**16.19**	**1.25**	**1.77**

7.4.5 Components of the Model-based Metaheuristics

To evaluate the impact of the improvement operators, the results of the BGA and the RGA with improvement operators are compared to the results obtained without the operators. Table 7.6 provides the average gap to T_{min} for a limit of 15,000 generated solutions per problem instance. As the differences between the BGA and the RGA are marginal, their combined average is shown. The improvement operators are able to generate better solutions but the differences are statistically insignificant.

The percentage of improved problem instances (Inst. %) is listed in Table 7.7. Again, only instances are considered on which the improvement operators are actually executed. Instances are excluded for which a makespan equal to T_{min} is determined within the initial population before the improvement operators are used. The operators have potential, as they achieve to improve on average

16% of the considered instances. This relates to 9% of all instances, then also including the ones on which the operators are not executed. However, the operators' performance declines after a peak on set B_{20}. It seems that the simple heuristic approaches come to their limits on large problem instances.

Next, the split of the population between the top solutions and bottom solutions is investigated. On average, across all instance sets and solution limits, 18% of the solutions in the population belong to the top solutions and the remaining 82% to the bottom solutions. Close to the top solutions' size limit of 20%, the population features a sufficiently large number of solutions that have a low makespan. Again, the results for the BGA and the RGA are nearly identical with differences of ± 1 percent point. 19% of all solutions exceed the BGA's and the RGA's upper limit for the makespan.

Lastly, let us consider an extension of the solution representation. As noted in Section 6.2.2.1, free activity start periods can be introduced by adding a start offset to the representation. To evaluate the resulting impact without interference of other factors, only the RGA is compared to variant RGA-SO that uses a start offset list. Similar to the HM's list σ, each activity's list entry delays the activity start by the specified number of periods. Two settings are tested for the HM's mutation operator for σ from Section 4.2.2.2: a mutation rate of 0.5% and one of 5%. Now, a different upper bound is applied. Since disjunctive arcs can already delay activities, the additional offsets are limited to the maximum of the minimum block lengths per problem instance. With this setup, the RGA-SO is capable of generating the optimal solution to the example depicted in Baumann et al. (2015, p. 536), which the original RGA cannot obtain. Nevertheless, the computational results of both methods are virtually identical with statistically insignificant differences. For the 0.5% mutation rate, the RGA and the RGA-SO differ in the gap to T_{min} by at most only ± 0.03 percent points for each instance set. The overall result is with 0.01 percent points minimally better for the RGA. Using the 5% mutation rate, the RGA-SO performs worse. Its overall result is 0.07 percent points worse than the RGA's. Thus, we can conclude that the delayed scheduling resulting from the disjunctive arcs is sufficient and no additional start offset is required.

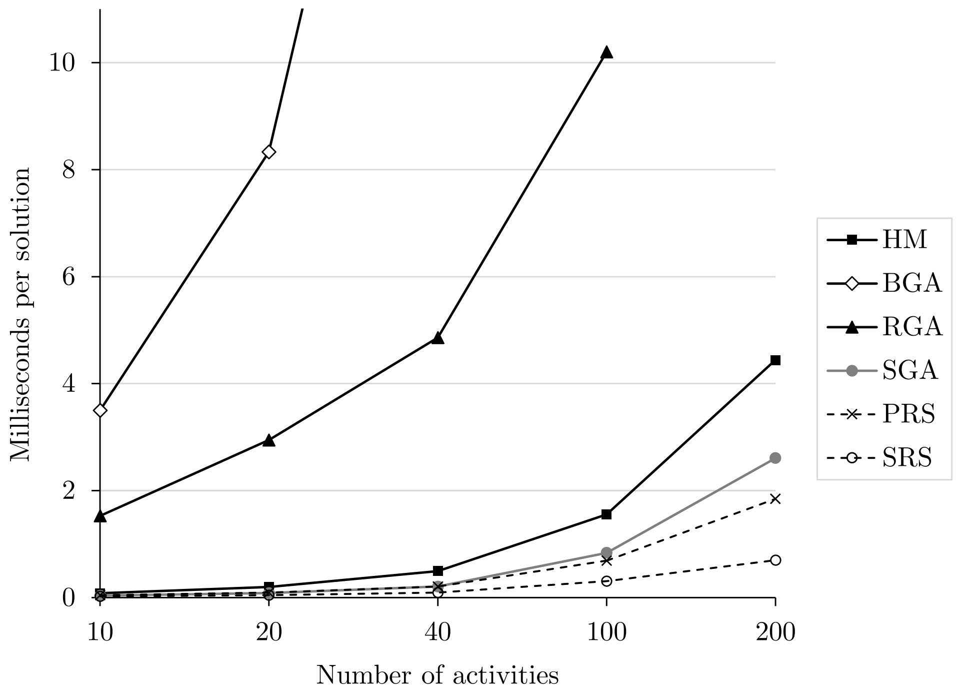

Figure 7.7: Average time to generate one solution in ms.

7.4.6 Computation Time

Besides solution quality, also the computation time is analyzed. Figure 7.7 provides an overview of the average time required to generate one solution per problem instance. It visualizes the average elapsed time per solution for a limit of at most 25,000 solutions per problem instance of sets B_{10} to B_{200}. The values on the x-axis are not equidistant. The data is listed in Table 7.8. The difference in the results is statistically significant in all table rows.

Computation time is highly implementation specific but the results give an indication on the scalability of the methods. Although not perfectly linear, the SGS-based methods scale well. For example, by doubling the number of activities, the HM's computation time grows by a constant factor of 2.5, when considering the range from 10 to 100 activities. From 100 to 200 activities, the factor increases to 2.86. A profiler analysis of the Java implementation revealed that this slight increase is mainly due to storage and handling of q_{irt} in the VNS and not due to the general algorithmic approach.

Table 7.8: Average time to generate one solution in ms.

Set	HM	BGA	RGA	SGA	PRS	SRS
$A_{\leq 55}$	0.14	5.55	1.89	0.05	0.06	0.02
B_{10}	0.08	3.50	1.53	0.04	0.04	0.02
B_{20}	0.20	8.33	2.94	0.09	0.09	0.05
B_{40}	0.50	21.80	4.86	0.21	0.21	0.10
B_{100}	1.55	76.91	10.20	0.84	0.69	0.31
B_{200}	4.44			2.61	1.85	0.70
Overall	**1.15**	**23.22**	**4.28**	**0.64**	**0.49**	**0.20**

Table 7.9: HM components: Average time to generate one solution in ms.

Set	GA	VNS
$A_{\leq 55}$	0.11	0.24
B_{10}	0.08	0.17
B_{20}	0.18	0.28
B_{40}	0.38	0.65
B_{100}	1.19	2.72
B_{200}	3.58	9.08
Overall	**0.92**	**2.19**

The computation times of the HM's GA and VNS are listed in Table 7.9. The GA's time develops roughly similar to the SGA's time from Table 7.8. The VNS requires more time due to the repeated solution analysis in the activity selection. As the VNS is not executed on all solutions, its average computation time may exceed that of the HM.

The key driver for computation time of the SGS-based methods is the applied SGS. Since the HM's FSGS performs more complex operations, its required time is higher. All methods that employ the parallel scheme, including the FSGS, require more time than the SRS that relies on the serial SGS. A likely reason is the resource allocation. In each decision period, the parallel SGS first continues active activities before it allocates additional resource quantities to active and eligible activities. In the serial SGS, an activity may as well be scheduled after the latest period of the current partial schedule, where no other activity is active. In this case, the resource allocation for the whole duration of the activity can be done in a single operation. Furthermore, dependent resources may cause the quantities of allocated principal or other dependent resources to be revised. In the parallel SGS, this revision may take place in each decision period and cause increased computation time when compared to the serial SGS.

Unsurprisingly, the model-based metaheuristics require more time than the SGS-based methods. Due to the BGA's high computation time, instance set B_{200} is not compared for the model-based methods. The BGA's computation time is not competitive with the other methods, as already the RGA is between 2 to 7 times faster. This difference is mainly caused by the underlying models, as discussed in the next section.

7.4.7 Models

The SP and the RSP mainly determine the model-based metaheuristics' computation time. Table 7.10 and Table 7.11 provide the average number of LP columns and rows before presolve, the average time to build or modify one LP prior to solving as well as the actual solve time including presolve. The averages across all instance sets and a limit of 25,000 generated solutions per instance are shown.

The RSP dominates the SP in all categories and is clearly the better model formulation for determining resource profiles. As the CPLEX presolve eliminates more than 90% of the SP's LP columns and rows, the SP model size after presolve is in the same order of magnitude as the size of the RSP. Still, the RSP is smaller and faster to solve.

The RSP's solve time grows moderately with the model size but its build time increases noticeably. As the RSP's constraints and variables may completely change depending on the intervals from the MP solutions, the model is rebuilt each time. On the one hand, the low solve time demonstrates the efficiency of the model, but on the other hand the high build time also indicates limitations of the implementation. The build time mainly determines the RGA's overall computation time. The BGA's ratio of build time to solve time is smaller, since only the constraint right-hand-side coefficients have to be modified in

Table 7.10: Statistics for the subproblem

Set	Columns	Rows	Build (ms)	Solve (ms)
$A_{\leq 55}$	1,904	7,540	14.99	7.17
B_{10}	1,215	4,343	6.64	3.30
B_{20}	3,655	14,087	19.05	11.10
B_{40}	11,858	47,971	73.05	35.47
B_{100}	65,004	272,669	385.86	235.42
Overall	**15,662**	**64,877**	**93.85**	**54.76**

Table 7.11: Statistics for the reduced subproblem

Set	Columns	Rows	Build (ms)	Solve (ms)
$A_{\leq 55}$	89	85	1.31	0.41
B_{10}	73	73	1.13	0.42
B_{20}	146	145	2.09	0.34
B_{40}	287	282	4.33	0.90
B_{100}	705	694	9.98	1.55
Overall	**258**	**254**	**2.00**	**0.72**

subsequent iterations. When comparing the SP's combined build and solve time to the BGA's average time per solution from Table 7.8, the time savings from the reduced number of solved SPs become obvious.

Chapter 8

Conclusions

This chapter summarizes the main research findings, provides concluding remarks, and outlines directions for further research.

8.1 Summary and Concluding Remarks

This work considered the resource-constrained project scheduling problem with flexible resource profiles (FRCPSP) for continuous resource quantities and discrete time periods. Chapter 1 gave an introduction to the topic and defined the research scope. The FRCPSP was described in Chapter 2. The problem was distinguished from related project scheduling problems in Chapter 3, where also existing solution approaches and relevant literature were reviewed. In Chapters 4 to 6, four metaheuristics were proposed. The methods were evaluated and compared to benchmark methods in the computational study presented in Chapter 7.

The hybrid metaheuristic (HM) from Chapter 4 embeds the flexible resource profile parallel schedule generation scheme (FSGS) into a metaheuristic framework that combines a genetic algorithm (GA) with a variable neighborhood search (VNS). The computational results demonstrated that the HM achieves significantly better results than all other compared methods. On small problem instances with 10 activities, for which the optimal solutions are known, the HM's results are near-optimal with an optimality gap of just 0.14%. The FSGS and the GA have significant positive impact on the overall results. Especially the applied concepts of non-greedy resource allocation and delayed scheduling allow the FSGS to generate significantly better solutions than a parallel standard schedule generation scheme (SGS). Although the VNS's impact on the overall results is smaller, it further improves on average every fifth problem instance that has already been processed by the GA before. Thus, the hybridization yields benefits. In comparison to existing metaheuristics, the VNS introduces new concepts to the FRCPSP by using resource transfers and a critical path analysis of resource flows.

The self-adaptive genetic algorithm (SGA) from Chapter 5 directly adapts the method of Hartmann (2002) to the FRCPSP. However, by employing standard SGSs, the SGA is not able to reach the HM's average solution quality. Specifically on small problem instances, the standard SGSs are the limiting factor and the HM's problem-adapted FSGS achieves significant advances.

Two model-based metaheuristics, the Benders genetic algorithm (BGA) and the reduced subproblem genetic algorithm (RGA), were proposed in Chapter 6. Whereas the HM and the SGA simultaneously schedule activities and determine resource profiles, the model-based methods rely on a problem decomposition into a master problem and a subproblem. In the master problem, activities are scheduled by determining their start periods and durations and additionally the minimum block length is enforced. In the subproblem, resource profiles are created by allocating resources. Both GAs use a novel solution representation based on disjunctive arcs to encode master problem solutions. The computational results indicated that the BGA's approach to derive the fitness from Benders optimality cuts and a lower bound of the resource constraint violations results in nearly the identical solution quality as the RGA's approach to always calculate resource profiles. However, by using the reduced subproblem, a compact reformulation of the subproblem, the RGA achieves significantly lower computation times. As valid master problem solutions may result in infeasible resource profiles, improvement operators were proposed. While they are able to slightly improve solutions, they do not have a statistically significant impact on the results.

All proposed metaheuristics yield significantly better solutions than two benchmark random sampling heuristics. Especially for large problem instances or instances with scarce resources, the metaheuristics achieve substantial advantages. The metaheuristics also find new best-known solutions that have not been determined by a mixed integer program on a commercial solver. Except for the BGA, the required computation time of all proposed metaheuristics can be regarded as low.

A common denominator in the findings from this work is that in highly constrained problems, such as the FRCPSP, the coverage of the search space requires careful consideration when designing metaheuristics. At the one end of the spectrum of considered approaches are metaheuristics that use standard SGSs. These methods are fast and always generate feasible solutions. However, their search space excludes certain optimal solutions that require non-greedy resource allocation and delayed scheduling. To overcome this limitation, the FSGS was introduced and embedded into a metaheuristic framework. The FSGS expands the search space of a standard parallel SGS while still only generating feasible solutions. On the other end of the spectrum are the model-based metaheuristics. Their benefit of always determining optimal resource profiles comes at the cost of higher computation time and solutions that violate the resource constraints.

8.2 Further Research

After a critical review of the proposed methods, further research may be directed at specific improvements: First, an optimized procedure for revising resource quantities of dependent resources in the FSGS from Chapter 4 and the standard SGSs from Chapter 5 may reduce the required computation time. Second, combining components of the HM and the model-based metaheuristics may have potential. The HM's VNS from Chapter 4 can also be applied on the resource profiles obtained by an LP in order to improve master problem solutions. Similarly, the heuristic improvement operators from Chapter 6 may benefit from the critical-path analysis of resource flows and the more sophisticated search framework of the VNS. Third, a new interval-based subproblem formulation that is directly compatible with the BGA's fitness evaluation using Benders optimality cuts may reduce the computation time of the model-based metaheuristics.

Further research beyond the scope of this dissertation may consider approaches for the FRCPSP with both continuous and discrete resources. Besides unifying two separate strands of literature, the practical applicability of the methods could be further increased, as all types of real-world resources are appropriately represented. From a theoretical perspective, a generalization to a continuous time axis may lead to new insights and further reduce the makespan. Moreover, it would be interesting to compare the proposed metaheuristics on larger problem instances of a more comprehensive FRCPSP instance library.

All in all, the metaheuristics provided and the concepts depicted in this work shall encourage researchers as well as practitioners to apply the versatile FRCPSP to further problem settings from business practice.

Appendices

Appendix A

Abbreviations

BGA	Benders genetic algorithm
DTRTP	Discrete time-resource tradeoff problem
FRCPSP	Resource-constrained project scheduling problem with flexible resource profiles
FSGS	Flexible resource profile parallel schedule generation scheme
GA	Genetic algorithm
GA-FSGS	The FSGS embedded into the HM's GA
GA-SGS	The standard parallel SGS embedded into the HM's GA
HM	Hybrid metaheuristic
LP	Linear program
LPF	Longest path following
MIP	Mixed integer program
MP	Master problem
MRCPSP	Multi-mode resource-constrained project scheduling problem
MTS	Most total successors
MWR	Most work remaining
OS	Order strength
PRS	Parallel random sampling (heuristic)
PR-FSGS	The FSGS embedded into a multi-start priority rule heuristic
RCPSP	Resource-constrained project scheduling problem
RF	Resource factor

RGA	Reduced subproblem genetic algorithm
RGA-SO	The RGA using an additional start offset
RS	Resource strength
RSP	Reduced subproblem
SGA	Self-adapting genetic algorithm
SGS	Schedule generation scheme
SP	Subproblem
SRS	Serial random sampling (heuristic)
VNS	Variable neighborhood search

Appendix B

Notation

Indices

i, j	Activity
r, k	Resource, specifically the principal resource k
t	Time period

Sets

$\mathcal{E}_{\mathcal{C}}$	Precedence relations: Conjunctive arcs
$\mathcal{E}_{\mathcal{D}}$	Precedence relations: Disjunctive arcs
$\mathcal{H}$	Solved subproblems
$\mathcal{P}_i$	Predecessors of activity i
$\mathcal{R}$	Resources
$\mathcal{R}_i$	Required resources of activity i
$\mathcal{R}_i^{pi}$	Required principal and independent resources of activity i
$\mathcal{T}$	Discrete time horizon
$\mathcal{T}_i$	Time window for activity i
$\mathcal{T}_r^{\delta}$	Interval start periods for resource r
$\mathcal{T}_{ir}^{\delta}$	Interval start periods for activity i and resource r
$\mathcal{T}_{ij}^{pair}$	Time periods for resource transfer of activity pair (i, j)
$\mathcal{V}$	Activities (excluding dummy start and sink)
$\mathcal{V}_{\mathcal{A}}$	Active activities
$\mathcal{V}_{\mathcal{C}}$	Critical activities

$\mathcal{V}_{\mathcal{E}}$	Eligible activities
$\mathcal{V}^{pair}$	Selected activity pairs (i, j)
$\mathcal{V}^{rec}$	Activities with rectangular resource profiles
Ω	Tuples of activity i, principal k, and dependent resource r

Constants and parameters

b_r	Availability of resource r in each period
$\underline{c}_i, \overline{c}_i$	Earliest and latest completion period of activity i
$\underline{d}_i, \overline{d}_i$	Lower and upper bounds of duration of activity i
e_{irt}	Interval start period of the interval of activity i and resource r that contains period t
g_{rt}	Duration of the interval of resource r that contains period t
g_{irt}	Duration of the interval of activity i and resource r that contains period t
l_{ir}	Minimum block length of activity i and resource r
n	Number of non-dummy activities
$\underline{o}_r$	Lower bound of outsourcing for resource r
$\underline{q}_{ir}, \overline{q}_{ir}$	Lower and upper usage bounds of activity i for resource r
$\underline{s}_i, \overline{s}_i$	Earliest and latest start period of activity i
T_{min}	Lower bound of the makespan
w_{ir}	Requirement of activity i for resource r
α_{ir}, β_{ir}	Coefficient α_{ir} and constant β_{ir} of linear resource function for dependent resource r of activity i
π_o	Penalty for outsourcing
π_q	Penalty for allocated resource quantity
$\gamma_{rt}^h, \zeta_{irt}^h, \theta_{irt}^h$ $\iota_{ikrt}^h, \kappa_{ir}^h, \mu_{irt}^h$ ν_{irt}^h	Dual values associated to SP constraints (6.17) to (6.23)

Binary decision variables

z_{it}	1 if activity i is active in period t, 0 otherwise

δ_{irt}	1 if activity i is allowed to change its allocated quantity of resource r from period $t-1$ to t, 0 otherwise

Integer decision variables

c_i	Completion period of activity i
C_{max}	Project makespan
d_i	Duration of activity i
s_i	Start period of activity i

Continuous decision variables

o_{rt}	Outsourcing of resource r in period t
q_{irt}	Quantity of resource r allocated to activity i in period t
η	MP objective function value

General variables

f	FRCPSP solution
F	Maximum number of generated solutions as termination criterion

Variables of the hybrid metaheuristic

$delay_i$	Number of periods that activity i has been delayed for
f^{rec}	FRCPSP solution featuring only rectangular resource profiles
$\vec{f}_{gen}$	List containing the unique best encoded solution for each GA generation as well as the corresponding decoded solution
F_{GA}	Maximum number of generated solutions for the HM's GA
F_{VNS}	Maximum number of generated solutions for the HM's VNS
l_{irt}	Number of periods in the block of resource r for activity i up to period t
v	Neighborhood number and number of resource transfers
λ	Activity list
ρ	Resource limit list
ρ_i	Resource limit of activity i

$\overline{\rho}_i$	Upper bound of resource limit of activity i
σ	Start delay list
σ_i	Start delay of activity i
$\overline{\sigma}_i$	Upper bound of start delay of activity i
χ_{ijrt}	Transfer quantity of resource r from activity i to activity j in period t
φ_r	Current leftover quantity of resource r
ξ_{ir}	Current remaining requirement of activity i for resource r

Variables of the model-based metaheuristics

A	Encoded disjunctive arcs
B_{ir}	Encoded interval start periods for activity i and resource r
B	All encoded interval start periods: B_{ir} for each activity i and required principal or independent resource r
D	Encoded durations
rsp^*	Optimal value of the RSP objective function (6.28)
sp^*	Optimal value of the SP objective function (6.16)
η^*	Optimal value of expression (6.35)

Appendix C

Lower Bound of Outsourcing

The total sum of outsourcing for resource r is defined as:

$$\sum_{t \in \mathcal{T}} o_{rt} \tag{C.1}$$

The **lower bound of outsourcing** $\underline{o}_r$ of (C.1) can be calculated for a valid MP solution without solving the SP/RSP or knowing the quantity of allocated resources:

- Based on the MP solution, the periods up to C_{max} are separated into disjunct timeframes without "overlapping" activities. That means, the first period t of such a timeframe has the property that all activities that are active in this period also start in this period. Similarly, the last period t' of the timeframe has the property that all activities that are active in this period also complete in this period.
- Outsourcing occurs if the compound resource requirement of the activities that are scheduled in the timeframe from t to t' exceeds the compound resource availability. Term (C.2) is the minimum resulting outsourcing in this timeframe. It contains the lower resource usage bound $\underline{q}_{ir}$ for activities that have not been completed up to period t' because the actual

quantities of allocated resources are unknown since a resource profile has not been determined.

$$\max(\sum_{\substack{i\in\mathcal{V}\\ s_i\geq t\\ c_i\leq t'}} w_{ir} + \sum_{\substack{i\in\mathcal{V}\\ s_i\geq t\\ c_i>t'}} \sum_{\tau=s_i}^{t'} \underline{q}_{ir} - \sum_{\tau=t}^{t'} b_r, 0) \tag{C.2}$$

- The lower bound of outsourcing $\underline{o}_r$ is obtained by summing up term (C.2) for all identified timeframes of the MP solution.

Bibliography

Alvarez-Valdés, R. O. and Tamarit, J. (1989). Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis. In Słowiński, R. and Węglarz, J., editors, *Advances in project scheduling*, volume 9 of *Studies in production and engineering economics*, pages 113–134. Elsevier, Amsterdam and New York.

Artigues, C., Demassey, S., and Neron, E. (2008). *Resource-constrained project scheduling: Models, algorithms, extensions and applications.* Control systems, robotics and manufacturing series. ISTE and Wiley, London and Hoboken.

Artigues, C., Michelon, P., and Reusser, S. (2003). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2):249–267.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms and Their Application*, pages 14–21, Hillsdale. L. Erlbaum Associates Inc.

Baumann, P., Fündeling, C.-U., and Trautmann, N. (2015). The resource-constrained project scheduling problem with work-content constraints. In Schwindt, C. and Zimmermann, J., editors, *Handbook on Project Management and Scheduling Vol.1*, International Handbooks on Information Systems, pages 533–544. Springer International Publishing, Cham.

Baydoun, G., Haït, A., and Pellerin, R. (2014). A rough-cut capacity planning model with overlapping. In Fliedner, T., Kolisch, R., and Naber, A., editors, *Proceedings of the 14th International Conference on Project Management and Scheduling*, pages 28–31, Munich. TUM School of Management.

Bedworth, D. D. and Bailey, J. E. (1982). *Integrated production control systems: Management, analysis, design.* Wiley, New York.

Bell, C. E. and Han, J. (1991). A new heuristic solution method in resource-constrained project scheduling. *Naval Research Logistics*, 38(3):315–331.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(3):238–252.

Bertrand, J. W. M. and Fransoo, J. C. (2002). Operations management research methodologies using quantitative modeling. *International Journal of Operations & Production Management*, 22(2):241–264.

Bianco, L. and Caramia, M. (2013). A new formulation for the project scheduling problem under limited resources. *Flexible Services and Manufacturing Journal*, 25(1-2):6–24.

Blazewicz, J., Lenstra, J. K., and Kan, A. R. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24.

Boschetti, M. and Maniezzo, V. (2009). Benders decomposition, lagrangean relaxation and metaheuristic design. *Journal of Heuristics*, 15(3):283–312.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., and Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41.

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.

Deckro, R. F. and Hebert, J. E. (1989). Resource constrained project crashing. *Omega*, 17(1):69–79.

Demeulemeester, E., De Reyck, B., and Herroelen, W. (2000). The discrete time/resource trade-off problem in project networks: a branch-and-bound approach. *IIE Transactions*, 32(11):1059–1069.

Demeulemeester, E. and Herroelen, W. (2002). *Project scheduling: A research handbook*, volume 49 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, New York.

Elmaghraby, S. E. (1977). *Activity networks: project planning and control by network models*. John Wiley & Sons, New York.

Frey, M., Kiermeyer, F., and Kolisch, R. (2014). Baggage flows at airports: A survey and generic model. Technical report, TUM School of Management, Technische Universität München, Munich.

Fündeling, C.-U. (2006). *Ressourcenbeschränkte Projektplanung bei vorgegebenen Arbeitsvolumina (Resource-constrained project scheduling with given work contents)*. 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.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. Series of books in the mathematical sciences. W.H. Freeman, San Francisco.

Gendreau, M. and Potvin, J.-Y., editors (2010). *Handbook of metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer, New York, 2nd edition.

Haït, A. and Baydoun, G. (2012). A new event-based MILP model for the resource-constrained project scheduling problem with variable intensity activities. In *IEEE International Conference on Industrial Engineering and Engineering Management*, Hong Kong. Institute of Electrical and Electronics Engineers.

Hans, E. W. (2001). *Resource loading by branch-and-price techniques.* Twente University Press, Enschede.

Hansen, P. and Mladenović, N. (2005). Variable neighborhood search. In Burke, E. and Kendall, G., editors, *Search Methodologies*, pages 211–238. Springer US, Boston.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, 45(7):733–750.

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics*, 49(5):433–448.

Hartmann, S. and 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.

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*, 127(2):394–407.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor.

Józefowska, J., Mika, M., Różycki, R., Waligóra, G., and Węglarz, J. (2000). Solving the discrete-continuous project scheduling problem via its discretization. *Mathematical Methods of Operations Research*, 52(3):489–499.

Kelley, J. E. (1963). The critical-path method: Resources planning and scheduling. In Muth, J. F. and Thompson, G. L., editors, *Industrial Scheduling*, pages 347–365. Prentice-Hall, New Jersey.

Kis, T. (2006). RCPS with variable intensity activities and feeding precedence constraints. In Józefowska, J. and Węglarz, J., editors, *Perspectives in Modern Project Scheduling*, volume 92 of *International Series in Operations Research & Management Science*, pages 105–129. Springer US, New York.

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2):320–333.

Kolisch, R. and Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37.

Kolisch, R., Meyer, K., Mohr, R., Schwindt, C., and Urmann, M. (2003). Ablaufplanung für die Leitstrukturoptimierung in der Pharmaforschung (Scheduling of lead structure optimization in pharmaceutical reserach). *Zeitschrift für Betriebswirtschaft*, 73(8):825–848.

Kolisch, R. and Sprecher, A. (1997). PSPLIB - a project scheduling problem library. *European Journal of Operational Research*, 96(1):205–216.

Kuhlmann, A. (2003). *Entwicklung eines praxisnahen Project-Scheduling-Ansatzes auf der Basis von genetischen Algorithmen (Development of a practical project scheduling approach based on genetic algorithms).* Logos-Verlag, Berlin.

Leachman, R. C., Dtncerler, A., and Kim, S. (1990). Resource-constrained scheduling of projects with variable-intensity activities. *IIE Transactions*, 22(1):31–40.

Lewis, J. P. (2006). *Fundamentals of project management.* American Management Association, New York, 3rd edition.

Li, H. (2015). Benders decomposition approach for project scheduling with multi-purpose resources. In Schwindt, C. and Zimmermann, J., editors, *Handbook on Project Management and Scheduling Vol.1*, International Handbooks on Information Systems, pages 587–601. Springer International Publishing, Cham.

Maniezzo, V., Stützle, T., and Voß, S. (2009). *Matheuristics: Hybridizing metaheuristics and mathematical programming*, volume 10 of *Annals of Information Systems.* Springer, New York.

Meyer, K. (2003). *Wertorientiertes Projektmanagement in der Pharmaforschung*. Berichte aus der Betriebswirtschaft. Shaker, Aachen.

Mitroff, I. I., Betz, F., Pondy, L. R., and Sagasti, F. (1974). On managing science in the systems age: Two schemas for the study of science as a whole systems phenomenon. *Interfaces*, 4(3):46–58.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Naber, A. and Kolisch, R. (2014a). A continuous-time model for the resource-constrained project scheduling with flexible profiles. In Fliedner, T., Kolisch, R., and Naber, A., editors, *Proceedings of the 14th International Conference on Project Management and Scheduling*, pages 166–168, Munich. TUM School of Management.

Naber, A. and Kolisch, R. (2014b). MIP models for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, 239(2):335–348.

Naber, A. and Kolisch, R. (2015). MIP models for resource-constrained project scheduling with flexible resource profiles: Comparisons between Baumann and Trautmann (2013) and Naber and Kolisch (2014). Technical report, TUM School of Management, Technische Universität München, Munich.

Poojari, C. A. and Beasley, J. E. (2009). Improving Benders decomposition using a genetic algorithm. *European Journal of Operational Research*, 199(1):89–97.

Project Management Institute (2013). *A guide to the Project Management Body of Knowledge (PMBOK Guide)*. Project Management Institute, Newtown Square, 5th edition.

Raidl, G. R. (2015). Decomposition based hybrid metaheuristics. *European Journal of Operational Research*, 244(1):66–76.

Raidl, G. R., Puchinger, J., and Blum, C. (2010). Metaheuristic hybrids. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of metaheuristics*, volume

146 of *International Series in Operations Research & Management Science*, pages 469–496. Springer, New York.

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. (2007). Solving the discrete time/resource trade-off problem in project scheduling with genetic algorithms. *Applied Mathematics and Computation*, 191(2):451–456.

Ranjbar, M. and Kianfar, F. (2010). Resource-constrained project scheduling problem with flexible work profiles: A genetic algorithm approach. *Transaction E: Industrial Engineering*, 17(1):25–35.

Reeves, C. (2010). Genetic algorithms. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 109–139. Springer, New York.

Schrage, L. (1970). Solving resource-constrained network problems by implicit enumeration—nonpreemptive case. *Operations Research*, 18(2):263–278.

Schramme, T. (2014). *Modelle und Methoden zur Lösung des ressourcenbeschränkten Projektablaufplanungsproblems unter Berücksichtigung praxisrelevanter Aspekte (Models and methods to solve the resource-constrained project scheduling problem under consideration of practical aspects).* Paderborn University, Paderborn.

Shaffer, L. R., Ritter, J., and Meyer, W. (1965). *The critical-path method.* Mcgraw-Hill, New York.

Sirikum, J., Techanitisawad, A., and Kachitvichyanukul, V. (2007). A new efficient GA-Benders' decomposition method: For power generation expansion planning with emission controls. *IEEE Transactions on Power Systems*, 22(3):1092–1100.

Talbot, B. F. (1982). Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science*, 28(10):1197–1210.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160.

Tritschler, M., Naber, A., and Kolisch, R. (2014a). A genetic algorithm for the resource-constrained project scheduling problem with flexible resource profiles. In Fliedner, T., Kolisch, R., and Naber, A., editors, *Proceedings of the 14th International Conference on Project Management and Scheduling*, pages 230–233, Munich. TUM School of Management.

Tritschler, M., Naber, A., and Kolisch, R. (2014b). A hybrid metaheuristic for the resource-constrained project scheduling problem with flexible resource profiles. In *International Conference on Operations Research 2014*, Aachen. German Operations Research Society.

Tritschler, M., Naber, A., and Kolisch, R. (2015a). A hybrid metaheuristic for resource-constrained project scheduling with flexible resource profiles. Working paper, TUM School of Management, Technische Universität München, Munich.

Tritschler, M., Naber, A., and Kolisch, R. (2015b). Model-based metaheuristics for resource-constrained project scheduling with flexible resource profiles. Working paper, TUM School of Management, Technische Universität München, Munich.

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(1):62–72.

Węglarz, J. (1981). Project scheduling with continuously-divisible, doubly constrained resources. *Management Science*, 27(9):1040–1053.

Węglarz, J., Józefowska, J., Mika, M., and Waligóra, G. (2011). Project scheduling with finite or infinite number of activity processing modes – a survey. *European Journal of Operational Research*, 208(3):177–205.

Zhang, L. and Sun, R. (2011). An improvement of resource-constrained multi-project scheduling model based on priority-rule based heuristics. In *8th International Conference on Service Systems and Service Management*, pages 1–5, Tianjin. Institute of Electrical and Electronics Engineers.