Ingenieurfakultät Bau Geo Umwelt

Lehrstuhl für Computergestützte Modellierung und Simulation

Simulation-based methods for float time determination and schedule optimization for construction projects

Gergő Dori

Vollständiger Abdruck der von der Ingenieurfakultät Bau Geo Umwelt
der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.rer.nat. Ernst Rank

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. André Borrmann
2. Univ.-Prof. Dr.-Ing. Markus König
   Ruhr-Universität Bochum

Die Dissertation wurde am 18.06.2015 bei der Technischen Universität München
eingereicht und durch die Ingenieurfakultät Bau Geo Umwelt
am 22.12.2015 angenommen.

Abstract

Every construction site is unique. Even if identical buildings or bridges are built, the boundary conditions vary from site to site. Therefore, every single construction site needs an individual schedule designed only for that specific project based on its individual conditions. Nowadays in the construction industry schedules are created manually in a time consuming and laborious way. To speed up this demanding process computer-aided approaches such as simulation-based techniques have been adapted from the manufacturing industry. However, these techniques are hardly known in the construction industry. Therefore, the aim of the author was to enhance, accelerate and, where possible, to automate this computer-based scheduling process so gaining more acceptance for it in the construction industry. The scheduling problem in the construction industry can be accurately described by the resource-constrained project scheduling problem (RCPSP) that is an optimization problem to find the one schedule with the shortest makespan under consideration of precedence and resource constraints. This is considered as an NP-hard problem and therefore to find the optimal solution for it is not possible in polynomial computational time.

To simulate construction sites and to generate schedules for the RCPSP the constraint-based discrete event simulation was used. This simulation is capable to determine single solutions for the RCPSP that are however not necessarily optimal, but feasible considering precedence and resource constraints. Though, to prepare the necessary input data for the simulation such as tasks, their durations, precedence and resource constraints is a time consuming process.

To enhance this preparation process a software-tool called Preparator was introduced by the author that applies the levels-of-detail approach, process patterns and activity packages to organize the large number of project tasks and their necessary attributes. With the Preparator the user can connect predefined process patterns and activity packages with 3D objects of the building to provide a clear overview of the already assigned construction tasks. The use of a 3D model also facilitates the creation of precedence constraints between construction tasks that belong to different building components.

Beside the enhancement of the input data preparation of process simulation, a newly developed simulation-based technique that is able to determine float time in one determination step for every individual task in the project also taking precedence and resource constraints into account was introduced. This was achieved by a methodology that is similar to the forward and backward pass of the conventionally used network scheduling techniques. The concept of the newly developed backward simulation is basically the same as that of the backward pass – the simulation actually runs forward in time but with reversed execution conditions. In order to achieve schedule compatibility and so identical schedule sequence in the backward simulation and the forward simulation, every task has to start at the same time or later in time than in the forward simulation. To this end, the task selection algorithm was set to a priority-based approach to determine the next executable task. The priorities of the tasks are assigned according to the completion date of the task in the forward simulation. By the introduction of the sequence enforcement constraints further developments were made to keep an identical schedule sequence for the backward simulation and the forward simulation.

By comparing the results of the backward simulation with the schedule of the forward simulation, the time difference between the earliest and latest start time of a task represents its total float time without exceeding the resource limits at any time during the project. The tasks without float time comprise the critical chain of tasks for the respective configuration of resources. A comprehensive case study was introduced to illustrate the application of this new approach.

In order to determine not only one feasible but also near optimal solutions for the RCPSP three new heuristic optimization strategies were introduced. The idea behind these approaches is to use an algorithm that swaps the position of certain task pairs within the schedule and so it is capable of traversing through the search space of the RCPSP. Furthermore, it was important that applying a swap of tasks within the program results in a new schedule. The first phase of the research was to identify the task pairs that when swapped result in a change in the schedule. These are called reasonable swaps. The next phase of the research was to define an algorithm that is capable of using these reasonable swaps and steering the simulation to generate schedules that work toward the optimum. An enumeration tree was used to represent the search space and the results of the optimization. A neighbor solution is generated by applying one reasonable swap to the schedule. To traverse effectively in this enumeration tree three heuristic approaches were introduced. The first steering technique is a Greedy-like algorithm that uses a constant, relatively low tolerance factor that neglects solutions that are beyond the "tolerated" worse results. The second steering technique is a simulated annealing-based technique which starts with a higher tolerance factor then the greedy-like algorithm. The tolerance factor is then reduced at each new level of the enumeration tree (new neighbor solutions).

Both strategies are enhanced with a tabu search algorithm that prohibits the "swap-back" of the previously applied last reasonable swap thereby lowering the chance of identifying the same schedule multiple times. To reduce the effect of the potential of identifying the same schedule multiple times within the enumeration tree, a third approach has been developed. This approach is called the depth oriented heuristic search algorithm. This approach uses the simulated annealing approach to determine results only few levels deep within the enumeration tree and then restarts the optimization with the current best solution as root node. Thus after the predefined number of levels the duplicated schedules will be ignored and the optimization continues with only the current best solution. Two comprehensive case studies have been investigated to test and compare the applicability of these heuristic optimization strategies.

A detailed outlook and suggestions for further research is conducted at the end of the thesis.

Acknowledgements

First and foremost I would like to express my sincere gratitude to my first advisor Prof. Dr.-Ing. André Borrmann for his continuous support of my PhD study and related research during the past six years. Your guidance and encouragement helped me in all the time of research and writing of this thesis even after I have left the university two years ago. You have always provided insightful discussions about research and supported me to visit various conferences around the world, thus creating unforgettable memories for me.

My sincere thanks also goes to Prof. Dr.rer.nat. Ernst Rank for employing me first as a PhD student at the Chair for Computational Engineering. Thank you also for your advises at the beginning of my research and for your support during my complete PhD studies.

I am also deeply grateful to Prof. Dr.-Ing. Markus König, my second supervisor, for his guidance and hard questions which incented me to widen my research from various perspectives. Thank you for the many opportunities to visit you and your chair in Bochum. My gratitude is also extended to your former team members especially to Matthias Hamm, Kamil Szczesny and Arnim Marx, who have spent long nights without sleeping while leading discussions about different research topics with me.

I would like to thank to the TUM Graduate School for their funding and support. Without them I could not have been visited the University of Alberta, spend six beautiful weeks in Canada and reach great scientific progress.

A special thanks goes to Prof. Simaan M. AbouRizk for arranging and facilitating my research visit at his chair and giving me insight of their main research topics. I would also like to thank to all the team member of his chair for all their support and guidance in Canada. A special thanks goes to my dear friend Ronald Ekyalimpa. Thank you for your encouraging words, for introducing me your beautiful family and for making my visit in Canada unforgettable.

I will be forever thankful to my former colleague, Hagen Wille, who introduced me to my former first advisor, Prof. Rank, such making me possible to start my PhD in Munich.

My gratitude goes also to Dr.rer.nat. Angelika Kneidl, my former office neighbor, for guiding me academically and emotionally through the rough road to complete this thesis.

I would like to thank to all my former colleagues, but especially my dear friends, Fabian Ritter, Alexander Braun, Simon Daum and Maximilian Bügler for their help whenever I approached them and their support in completing my PhD. You have always been there for me and I hope you will also get your PhD soon.

A special thanks goes to the companies Max Bögl and Fahrner for providing me the schedules and further information about the two real bridge construction sites that has been introduced in this thesis.

I would also like to thank for the support to my current colleagues from the "Ingenieurbüro Grassl", especially to the directors, Dr.-Ing. Hans Grassl und Markus Karpa for granting me the necessary time and giving me guidance on how to complete my PhD.

Many thanks to my friends at the "Gelb-Schwarz Casino München" dancing studio, especially to Melanie Ruf and Bastian Kunst, for their support in good and bad times.

I would also like to thank to my old friends, Gergely Szénássy and Tamás Hasenauer, for being always there for me and I am sure that we will spend much beautiful time together in the future.

My great thanks goes also to Aleksandra Rawa, who not only encouraged me for the last six year, but has also been one of my best friends.

Another person I would like to express my gratitude is Felícia András. Although we live from each other far-far away you are still my best friend and I am grateful that we can always count on each other.

I must acknowledge with tremendous and deep thanks my dancing partner and friend, Julia Hofmann. Through your support and unwavering belief in me, I have been able to complete this long dissertation journey. You have motivated me in so many ways and there are no words that can express my gratitude and appreciation for all you have done and been for me.

Lastly, I would like to thank my family for all their love and encouragement. For my parents who raised me with a love of science and have sacrificed everything for my brother and myself and always provided unconditional love and care. Although the physical distance between us is large now, you must know that I love you with all my heart and I would not have made it this far without you. Thank you!

January 2016, Munich
Gergő Dori

Contents

1 Introduction

1.1 Executive summary

Every construction site is unique. Even if identical buildings are built, the surrounding conditions (e.g. the ground material, traffic, already existing buildings, site layout and weather) or the construction conditions (e.g. available time, available resources or budget limits) vary from site to site. Therefore, every single construction site needs an individual schedule designed only for that building based on its individual conditions. Currently this is a time-consuming and labor-intensive iterative process that is performed manually in the construction industry. To speed up this demanding process computer-aided approaches such as simulation-based techniques have been adapted from the manufacturing industry and further development was made by researchers to meet the needs of the construction industry. These techniques are capable of evaluating and generating feasible construction schedules in short computational time. In this thesis, new, sophisticated extensions will be introduced for the simulation-based scheduling technique which enable the generation of near optimal and flexible schedules.

1.2 The construction project

A construction project is always made up of a set of *construction tasks* that symbolize atomic processes on a construction site. These construction tasks are necessary to erect the designed building. Every task requires a different kind of labor force, along with the machines and materials for its execution. When one of these resources is not available, the execution of the task cannot be started

and the construction might have to be delayed. Furthermore, the construction tasks have different kinds of interrelationships or dependencies upon each other that define the exact execution order of the affected tasks. The End-Start relationship is the most commonly one in use. It is also called *precedence relationship*, which defines that a task cannot be started until all of its predecessors have been completed. The construction tasks with their interrelationships can be visualized by a precedence graph that provides a clear representation of the overall project (Figure 1-1).

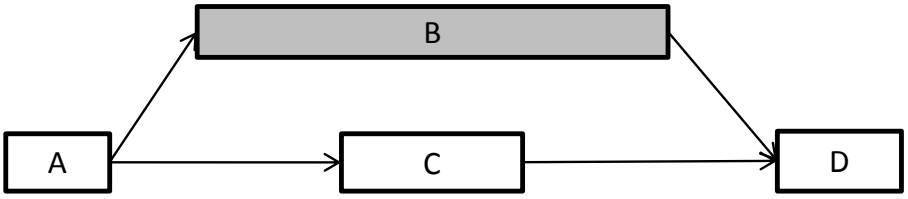


Figure 1-1: Precedence graph of four construction tasks: A, B, C and D. The different colors correlate to different resource needs and the arrows represent precedence relationships between the tasks. The length of the task boxes is proportional with the execution duration of the task.

Each of the construction tasks have estimated durations and might also have measures such as cost or quality. In order to know when a task should be executed and when the necessary resources should be available on the site, a construction schedule is necessary. Construction schedules are determined by the Project Manager and are used to control and monitor the progress of the construction. Scheduling is a complex and challenging task that can have different objectives. These objectives are discussed in the next section.

1.3 Objectives of scheduling

Not only do construction conditions vary from site to site, but also the contractor company may have different scheduling expectations that must be taken into account. These expectations can be formulated as the objectives of the scheduling.

1.3.1 Time-cost trade-off

The most common objective of scheduling is to find a schedule that is completed in a given time and has a low budget. These two criteria have an influence on each other and so it is complex to satisfy both. For example, when a task on a construction site must be executed faster than it has been planned, e.g. because of some delays of other tasks, additional resources need to be applied to accelerate the completion of the task (also called crashing the task), but this will also raise the costs. In other words, there is a trade-off between the makespan and the costs of a project (Hendrickson and Au 1989, Berthaut et al. 2011).

Therefore the aim of scheduling is to find a schedule for which this so called *time-cost trade-off* is acceptable. Figure 1-2 represents a diagram showing the relation between the total cost and the makespan of a construction project. It is assumed, that the relation-line represents the best possible solution (the one with least total costs) for the corresponding schedule makespan including an optimal execution order of the tasks and resource utilization. Any change within the schedule (changing

execution order of tasks, adding or subtracting resources, etc.) will introduce extra costs when wanting to keep the same makespan.

As demonstrated in Figure 1-2, there is a schedule with the makespan t_m where the total costs (c_m) of the project are minimal. From this point, the makespan of a project can be shortened by adding extra resources, using a second shift and/or overtime and using machines with higher performance factors[1]. However, due to the time-cost trade-off this crashing process will induce additional total costs for the project. It may be that the profit, induced by the shorter makespan, becomes less than the additional costs of the acceleration. Alternatively, the minimal cost point should not be considered as an option for an ideal schedule, since both the makespan and the costs of the project increase [2]. The favored time-cost trade-off values are always located on the left hand side from the t_m makespan value.

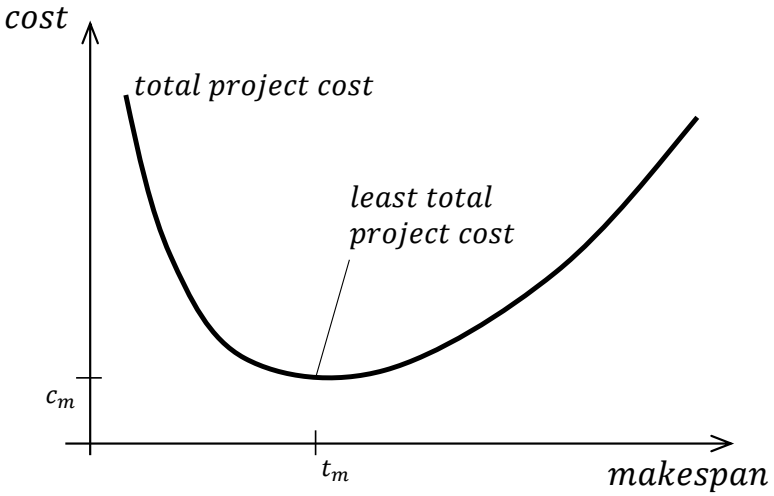


Figure 1-2: Relation between the schedule makespan and the total cost of a project (based on Berthaut et al. 2011 and Baker 1991)

1.3.2 Time and cost limits of the project

A tighter and also often necessarily used variation of the above introduced objective is not only to search for a good time-cost trade-off value, but also to ensure that you do not exceed a predefined makespan and/or cost limit (Gordon et al. 2002, Brucker 2007). In this case the schedule must be finished before a predefined deadline (due date – t_2 und corresponding c_2) and/or must not exceed a predefined budget (Figure 1-3 – c_1 and corresponding t_1). These constraints limit the feasible region of the possible solutions for the makespan between $t_1 \leftrightarrow t_2$ and for the costs between $c_1 \leftrightarrow c_2$. Therefore, when the duration of the project is more important than the costs, the solution with the lower makespan and higher costs $(t_1; c_1)$ shall be chosen. Alternatively when the costs play a more important role, the one with the lower cost but longer makespan $(t_2; c_2)$ will be selected. In the case

[1] This technique is called crashing

[2] Right from the minimal cost point often the daily costs of the workforce and machines are, due to lower performance factors and/or lower amounts, lower than the daily costs of the one with minimal total costs, but due to the longer working time of these work forces, their total costs become higher than the referenced one.

where both components are equally important, a solution somewhere in between these boundaries might be selected.

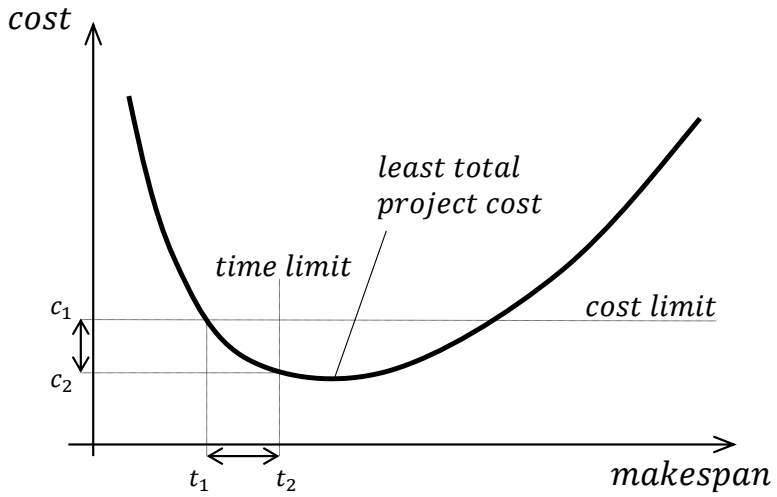
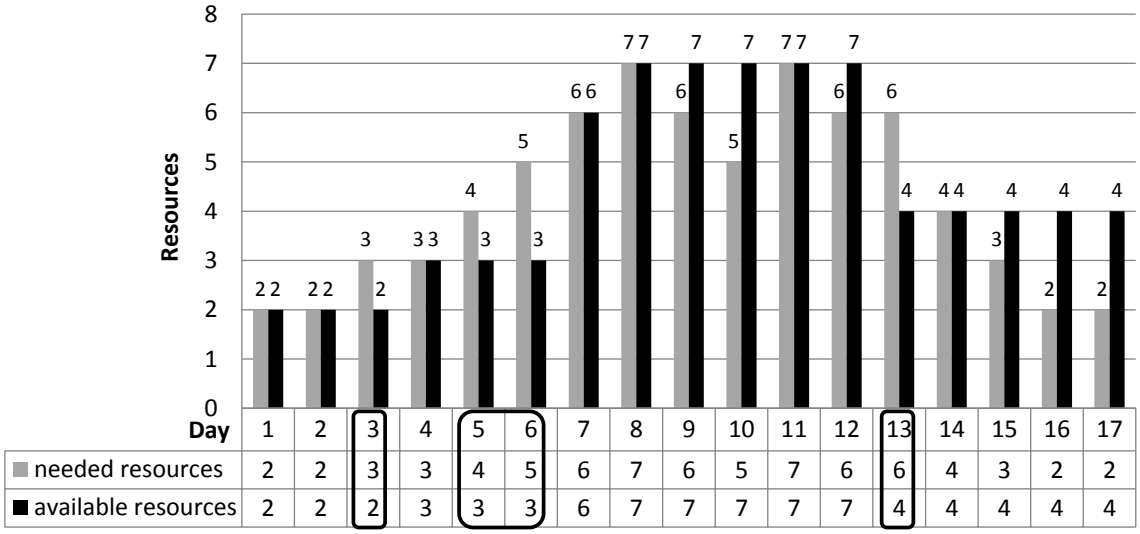


Figure 1-3: Restricting the area of feasible schedules by applying cost and time limits for the completion of the project

1.3.3 Scheduling under limited resource availability

Another important objective of scheduling is to not exceed predefined resource limits on the construction site where the availability of the resources is limited, or possibly even unavailable, for a certain time frame (Brucker 2007, Klein 2000 and Blazewicz et al. 1983). Violation of these limits might lead to a delay in the project and therefore an increase in the total costs. The example on Figure 1-4 represents the resource diagram of a small project with a makespan of 17 workdays. On days 3, 5, 6 and 13 of this project the number of needed resources exceeds the limits of the available resources. In this scenario is likely that the project cannot be finished in time and rescheduling will be necessary.



Day	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
needed resources	2	2	3	3	4	5	6	7	6	5	7	6	6	4	3	2	2
available resources	2	2	2	3	3	3	6	7	7	7	7	7	4	4	4	4	4

Figure 1-4: Diagram of the available and necessary resources of a short project (necessary resources: gray, available resources: black, resource limits exceeded: black box)

Related to the aforementioned objectives, an optimization problem can be formulated. The optimization problem is a search for the optimal execution sequence of construction tasks with a minimal makespan for the project, while not exceeding any resource limits. Based on the makespan of this optimal schedule, it can be determined if a predefined time limit for the project is manageable, or if further resources must be applied to meet this limit.

The search for the most balanced resource utilization for a schedule with a predefined makespan is defined as another related optimization problem. This is called the resource-leveling problem. Here, the tasks are sorted and swapped in such a manner that the utilization level of the diverging resources for the whole makespan becomes as close to constant as possible.

1.3.4 Quality aspects

One further criterion for scheduling is to create a high quality product. The quality of a product depends primarily on the qualifications of the labor working on the product and the time being invested to create the product (Kang and Myint 1999, Tareghian and Taheri 2006). When a high quality product is desired the time and cost factors are not as important as the quality itself. To reach this goal highly qualified labor should be used with an extended time frame to complete the required work. Hence the higher the required quality of the product, the higher its costs will be.

1.3.5 Flexibility of a schedule – the float time

Another important aspect of scheduling is to create a flexible schedule. The key measure of flexibility within a schedule is the float time (Raz and Marshall 1996). Float time describes the time frame within which the execution of a task can be moved or how much the duration leeway of a task can change, without impacting the makespan of the project or the execution of subsequent tasks. The tasks without float time constitute the *critical path* of the project (Figure 1-5). The critical path can also be defined as the longest path or sequence of tasks throughout the project. The length of the path

represents the shortest duration required to complete the project. A delay in any of these critical tasks will result in an increase of the project's overall makespan and therefore in a delay of the project itself.

Figure 1-5 presents an example schedule for the four tasks (A, B, C, and D) that have already been introduced in Figure 1-1. The precedence relationships of the tasks allow the simultaneous execution of tasks B and C, since B has a longer duration than C. Because C has a time frame from the end date of task A until the commencing date of task D, it can be moved freely without impacting the project's makespan. This time frame represents the total float time of C. The tasks A, B and D are part of the critical path because they have no float time and moving or extending them will change the overall completion time of the project. To identify these critical paths within a project is very important since any change to these tasks will directly influence the makespan and also the costs of the project. Therefore, when two schedules exist for the same project containing the same makespan and similar costs, the preferred one is the schedule in which the tasks have more float time.

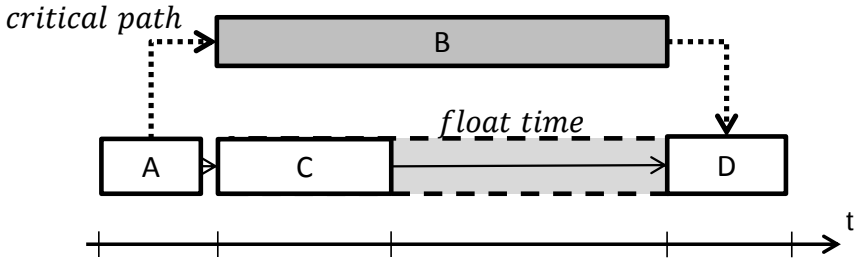


Figure 1-5: Representation of float time and critical path within a schedule. Tasks: boxes; task interrelationships: arrows; critical path: dotted arrows; float time: grey, dashed box

1.4 Scheduling and optimization

After setting up the objectives and restrictions of the construction project, a scheduling process that meets the objectives and satisfies the restrictions is carried out in order to establish a reasonable sequence of project tasks. The most commonly used scheduling techniques in the construction industry are the network scheduling techniques. Network scheduling techniques include the Program Evaluation and Review Technique (PERT), the Critical Path Method (CPM) and the Precedence Diagram Method (PDM) (Kerzner 2003). Further existing methods are the Gantt chart and linear scheduling methods. All of these techniques will be introduced and discussed in detail in Chapter 2.

The use of one of these techniques always results in one single schedule that barely meets the aforementioned objectives or restrictions. The schedule determined by network scheduling techniques for example has an optimal makespan, but it is not capable of addressing resource restrictions on the construction site. Since in reality the amount of available resources is limited, delays will most likely occur resulting in an increase in both makespan and the costs of the project compared to the calculated values.

Therefore, to determine a schedule that satisfies all the aforementioned requirements, further enhancements to the schedule are necessary. A scheduling process that satisfies all the objectives and restrictions can also be described as a multi-objective optimization, with the objectives of finding the schedule with the shortest makespan, lowest costs, highest quality, etc. An example of this problem is the so called *Resource-Constrained Project Scheduling Problem* (RCPSP – see Section 2.2.2)

(Pritsker et al. 1969, Blazewicz et al. 1983, Hartmann and Briskorn 2010, Beißert 2012). The RCPSP consists of finding the optimal schedule with the shortest makespan with respect to the precedence constraints between the tasks and the resource availabilities. This problem accurately describes the scheduling problem in the construction industry and therefore its solution will be one of the main topics of this thesis.

The RCPSP is a *combinatorial optimization problem* (Blazewicz et al. 1983) that, due to the limited amount of available resources and the aim to find the optimal solution for complex cases, is considered an NP-hard problem and therefore it is infeasible to be solved within polynomial computational time (Blazewicz et al. 1983) (see Section 6.4).

In Section 6.5.1 solution finding procedures (such as Branch-and-Bound, and Integer Programming) will be introduced that are capable of determining the exact optimal solution of this optimization problem. However, due to the complexity, the duration of the computation cannot be predicted.

Therefore, further approaches have been developed that are capable of delivering not the optimal solution, but also good, near-optimal solutions for the problem in a shorter computational time. These solution approaches are called the heuristic methods and include simulated annealing, genetic algorithms, etc. (introduced in Section 6.5.2). The solution mechanism of the heuristics is based on intuitions and previous experiences. One goal of the author was to develop a method of solving the RCPSP in a short computational time, the trade-off being good solutions rather than optimal solutions.

One such possible solution is the iterative enhancement of a schedule. This iterative scheduling process consists of four phases (Tulke 2010, Beißert 2012):

> *Phase one*: the fragmentation of the project into individual construction tasks and the definition of their interrelationships according to a desired construction scenario. The result of this step can be visualized amongst others by a precedence graph (Figure 1-1).
>
> *Phase two*: the determination of the duration for every task is based on the amount and performance of the available resources and the size of the task.
>
> *Phase three*: the connection and ordering of the individual construction tasks into a schedule according to their duration and their dependencies on other tasks.
>
> *Phase four*: phase two and phase three are repeated until the schedule meets all desired objectives of the scheduling. For that, in phase two, the amount and/or the performance of the available resources should be varied. If the objectives cannot be met this way, at phase one a new construction scenario should be defined and the complete process should be restarted.

These four phases consist of the solution of an *ordering, an assignment and an optimization problem* (Beißert 2012). The ordering problem is the investigation of a multitude of possible schedules that can vary in the sequence order of the construction tasks or differ in their construction scenario. The assignment problem describes the decision made when the available resources are limited and a choice needs to be made as to which of several competing tasks should get the available resources. The optimization consists of the search for the schedule that best fits the desired optimization criteria.

In the construction industry today these phases are carried out manually using conventional scheduling techniques (see Section 2.2) for phase three. The improvement and quality of the schedule is mainly dependent on the knowledge and previous experience of the designer. This is a time-consuming process. To enhance and accelerate this process, computer-aided simulation techniques

from different industries have been investigated that are capable of efficiently solving scheduling problems with resource constraints.

1.4.1 Simulation-based scheduling in the manufacturing industry

Similar to the construction industry, the manufacturing industry also has scheduling problems. The shop problem, where an optimal sequence of tasks and machines is searched for, is a well-known scheduling problem. To find feasible solutions for such a problem computer-aided simulation methods have been applied in addition to the manual techniques. Simulation is defined by Shannon (1975) as "the process of describing a real system and using this model for experimentation, with the goal of understanding the system´s behavior or to explore alternative strategies for its operation." In this definition, the exploration of alternative strategies for a shop problem can also be formulated as searching for feasible and productive schedules. Process simulation methods not only take resource limits into account, but also handle the other restrictions of the problem, e.g. interrelationships between the tasks and resource constraints that define which resources are necessary in order to execute the proper task.

In contrast to the analytical solutions (such as Branch-and-Bound and Integer Programming – see Section 6.5), simulations can handle more complex interactions and can therefore result in a more realistic schedule. However, they are only capable of generating single stand-alone schedules each of which is one feasible solution to the problem but not the optimal one. It then becomes the job of the scheduler to optimize the process by changing the parameters of the model. Therefore, simulations are used to generate schedules for different scenarios. The results of these different scenarios are compared to each other so as to explore a wide range of possible schedules. Simulations also provide the possibility of testing every small detail of the schedule before starting the actual production. These tests can lower the risk of collisions and lower the high idle times for machines by identifying bottlenecks within the schedule. Since simulations are mathematically simpler then the analytical methods, they are easier to understand and use (Page and Kreutzer 2005). Therefore, these methods are not only used for research purposes, but also in the industry. However, such a simulation tool is quite pricey. The company needs both the software to work with, but also a skilled employee and a significant amount of accurate data about the project to be simulated. Fortunately all the costs and efforts associated with simulations will pay off during the production phase because of the advantages and capabilities of the simulation methods mentioned above (Page and Kreutzer 2005).

1.4.2 The adaption of process simulation methods into the scheduling of construction projects

Due to the success of the process simulation technique in the manufacturing industry, researchers adapted it for the construction industry (Halpin 1977, Martinez and Ioannou 1994, AbouRizk and Hajjar 1998a, Günthner and Borrmann 2011). The discrete event simulation (DES) turned out to be the most suitable technique for construction simulation, as tasks are modelled as a pair of events (start and finish) and the simulation time of the model jumps forward between these events while the model´s state stays constant between these discrete time steps (Banks and Carson 2009). The concept of the DES will be introduced in Section 3.3 in detail. Further suitable approaches to simulate construction processes are the Petri-Nets (Petri 1962, Petri 1966, Sawhney 1997) and the agent-based simulations (Knotts et al. 2000, Horenburg et al. 2012; Sawhney et al. 2003, Horenburg et al. 2012,

Kooragamage et al. 2013) which will be introduced in Section 3.5.9 and 3.5.10, respectively. To apply these simulation techniques to the simulation of construction projects, some changes and extensions of the simulation concept were necessary.

Manufacturing processes are executed repetitively in a place bounded environment where machines are static and products travel on conveyor belts between them, whereas in the construction industry the resources (machines, labour) are dynamic and the product (the building) is static. In addition, operations in the manufacturing industry can only be executed by one specific machine, while in the construction industry one task can be executed with diverging resources without impacting the duration of the task. These behaviors lead to a more dynamic and varying site layout for the construction industry compared to the one of the manufacturing industry. Since the scheduling methods for the manufacturing industry are designed for a static factory layout with repetitive processes, the scheduling methods for a construction industry must be customized to a model that describes the dynamically changing layout.

In the 1960s it was realized that although construction projects are unique, they contain repetitive processes and show similarities with the behavior of manufacturing industry processes, e.g. such as earth transport, tunneling or road construction that also can be modelled with a static layout (AbouRizk et al. 1992). These processes were the focus of the most important discrete event simulation-based construction simulation frameworks such as CYCLONE (Halpin 1977), STROBOSCOPE (Martinez and Ioannou 1994) and Symphony (Hajjar and AbouRizk 1999). The detailed introduction of these frameworks and their further development is introduced in Section 3.4.

In order to apply also the layout of the construction site into the simulation Chahrour and Franz (2002) investigated the applicability of the building block-based simulation concept in construction project planning. This is widely used concept in the manufacturing industry. Chahrour (2006) introduced a prototypical simulation framework that uses product modeling concepts as data structure and is capable of integrating simulation concepts with CAD representations of the construction site.

To face the problem with the dynamically changing site layout and to involve also non-repetitive processes into the simulation based on the research of Chahrour and Beißert et al. (2007b) a new simulation concept has been developed. The idea behind this new concept was to generate diverging schedules by following the behavior of real construction projects, such as obtaining diverging execution sequences of the tasks based on dynamic and spontaneous decisions (Beißert et al. 2007a, Wu et al. 2010a). To implement this spontaneous and dynamic behavior into the simulation concept, the DES was combined with the *constraint satisfaction paradigm*.

The constraint satisfaction concept is a strong paradigm to model complex combinatorial problems (Blazewicz et al. 2007, König et al. 2009a), such as scheduling problems like the RCPSP. A constraint satisfaction problem is described by variables, domains and constraints, where the goal is a feasible solution that satisfies every constraint. For a construction scheduling problem, the tasks and resources (e.g. machine, employee and material) are defined as variables, and the interrelationships between the tasks, their resource needs and the availability of the resources are the constraints.

By combining the constraint satisfaction paradigm with the simulation concept, a new and powerful technique was introduced, that is able to generate feasible schedules for construction projects taking also resource limits into account. This technique is called the constraint-based discrete event simulation (CBDES) (Beißert et al. 2007b, Beißert 2012). The CBDES always results in one feasible schedule and in case of a repeated simulation (Monte Carlo simulation), due to the random executable task selection, it is also able to generate diverging schedules. Hence, this technique also can be used for optimization purposes (Beißert 2012, Hamm and König 2010, Szczesny et al. 2012) and to generate feasible solutions for the resource-constrained project scheduling problem. For

optimization purposes, however, an extension is necessary that steers the simulation by defining diverse input data to generate better solutions that get even closer to the desired optimal solution. Near optimal solutions for the resource-constrained project scheduling problem can be generated in an efficient way using the constraint-based simulation approach extended with a steering algorithm (Chapter 7). This is the basis on which this thesis is grounded.

One further drawback of the adapted simulation-based scheduling approaches is the high amount of necessary data to start the simulation. The preparation of all this data is a time consuming and laborious work. Possible solutions to accelerate this process will be introduced in Chapter 4.

Another important characteristic of a task in the construction industry that is described differently in the manufacturing industry (idle time of machines) is its flexibility within the schedule, which can be described by its float time. A new approach will be introduced in Chapter 5 how to determine float time for individual construction tasks within a schedule under consideration of precedence and also resource constraints.

1.5 Main contributions of this thesis

This thesis focuses on construction process scheduling using the constraint-based discrete event simulation. Although the introduced methods have only been tested on bridge construction projects, they are applicable to any kind of construction project. The application of the developed methods has been applied to bridge construction projects, since the diversity of the applicable construction methods is restricted and the project scheme is clearly arranged compared to those of high rise building and further constructions.

Although the introduced constraint-based simulation approach is a powerful tool in generating feasible schedules for construction projects, it requires a large amount of input data, including the list of schedulable tasks, the interrelationships between tasks, the required resources and the identification of the resources actually available on the jobsite. All of this data is required to start the simulation, but obtaining this data is a very time consuming process. Therefore, one aim of this thesis is to develop methods that accelerate the process of organizing all this data in a faster and more efficient manner. The developed and applied techniques are introduced in Section 4.3. These methods are advancements of the research work of Wu et al. (2010a). The levels-of-detail approach has been adapted and further process patterns have been developed. The activity packages have been extended with further attributes such as priority and further requirements relating to the necessary resources. As a result of this research a software program called “Preparator” has been developed that is capable of defining all necessary data for the simulation. The most important task of the Preparator is to connect the objects of a 3D construction model with process patterns, diverging necessary attributes for its construction and ordering the processes into a precedence graph.

A significant advantage of the conventional network-based scheduling techniques over the simulation-based method is the capability of determining float time for every single task. A new method will be described to overcome this drawback of the simulation approach and to increase its competitiveness in comparison to the conventional network-based scheduling techniques. This new method extends the simulation approach with the capability of determining float time for every individual task in one iteration step (Chapter 5). To calculate float times, similar to the backward-pass analysis used in CPM, a newly developed backward simulation method along with a coupling process extending the common forward discrete event simulation is introduced. To achieve schedule compatibility, every task has to start at the same time or later in time than it does in the forward

simulation. To this end, the task selection algorithm of the simulation has been modified so that the backward simulation uses a priority-based approach to determine the next executable task and new constraints are defined based on the results of the forward simulation. By comparing the results with the schedule of the forward simulation, the time difference between the earliest and latest start time of a task represents its total float time.

Since the simulation-based scheduling technique generates feasible schedules in a fast and efficient way, it can be beneficially used for schedule optimization purposes. For this, an external algorithm is necessary that controls the input data of the simulation to generate diverse schedules.

At this point it is important to stress that the number of possible feasible schedules for a large and complex construction project can be significant. Even using a small example where there are 10 tasks that each require the same resource with there being only one resource available, the amount of possible feasible schedules is the permutation of the tasks without repetition is: $n! = 10! = 3628800$. In reality, a small construction project is made up of several hundreds of tasks. A large construction project can have even more than ten thousands tasks and due to the factorial relationship in this particular case between the amount of tasks and the possible feasible schedules the number of possible feasible solutions is tremendous in these cases.

Due to this very large solution space, the algorithm must be able to cover all of these feasible solutions and should be able to traverse between them in an efficient way. Therefore, a new algorithm has been developed based on the principle of swapping position of certain tasks within the schedule by swapping their priorities (see Section 7.3). The advantage of this algorithm compared to other steering algorithms applied to the CBDES is that swapping the priorities of these certain tasks will definitely lead to a new schedule. This cannot be assured with the other existing methods (see Section 6.5.2.2 and Section 7.2).

Although the developed algorithm is capable of covering every possible feasible solution to the problem, the determination of all of these solutions would not be feasible in the polynomial computational time. Therefore, in order to shorten the computational time of the optimization, the developed algorithm has been connected to heuristic algorithms (enumeration tree, greedy-like heuristic, simulated annealing and tabu search) that can result in near optimal solutions.

Since optimization problems are very complex and difficult to solve, a three-staged automation system has been applied that enables the application of the aforementioned methods not only for research purposes, but also for planning purposes in the construction industry. The automation process is presented in Figure 1-6. Step one of the three-staged automation process consists of the preparation of the input data, followed by the simulation which generates a schedule based on the input data. As an iterative process the optimization algorithm controls the simulation and generates diverging schedules to find the schedule with the shortest makespan. The necessary steps to create this automated construction schedule optimization will be introduced in the corresponding sections of the thesis.

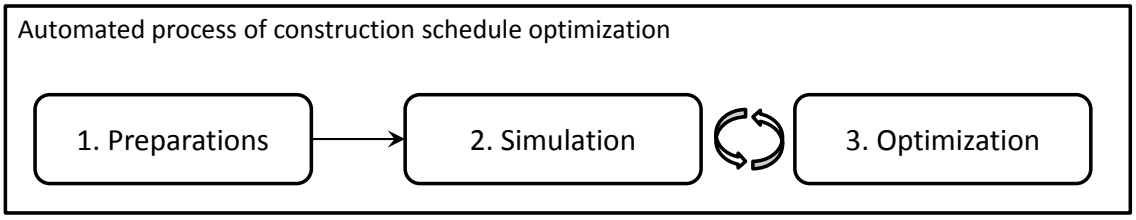


Figure 1-6: The automated process of automation of construction schedule optimization: 1. Preparation of the input data, 2. Simulation, 3. Optimization.

1.6 Structure

This thesis is organized into nine Chapters. The first two Chapters introduce the state of the art in process scheduling. The third Chapter addresses the introduction of simulations and simulation-based scheduling techniques. Furthermore, it contains a detailed literature review in simulation-based construction project scheduling. Chapters 4, 5, 6 and 7 represent the core of the thesis. Here, new methods will be introduced for the simulation-based scheduling technique and sophisticated heuristic optimization techniques to solve the resource-constrained project scheduling problem will be illustrated. These Chapters are closed by a case study, which describes the applicability of the new scheduling and optimization methods to the construction industry. The thesis is concluded by the summary of the results and an outlook. The Appendix contains simulation results for the validation of the new methods, code segments and simulation input data for the introduced case studies.

- Chapter 2: Scheduling problems and conventional scheduling techniques

The second Chapter of the thesis focuses on the general introduction of diverging scheduling problems such as the construction project scheduling problem, the shop problems and the resource-constrained project-scheduling problem. The conventional scheduling techniques and solution methods for the introduced scheduling problems in both the construction and manufacturing industries will be reviewed in detail. Among others, the Gantt chart, network-based scheduling techniques and particularly the advantages of a simulation-based scheduling technique will be analyzed. After the comparison of the main characteristics of the construction and manufacturing industries, the expectations for an adaption of the simulation-based scheduling technique to the construction industry will be introduced.

- Chapter 3: Simulation-based scheduling

The application of the simulation-based scheduling technique forms the basis of this thesis. In addition to providing the definition and the basic terms of the discrete event simulation, an explicit historical review of applied simulation approaches from the emergence of simulation-based construction project scheduling approaches up to the most recently distributed simulation methods is presented. Further, the best-suited simulation approach for scheduling purposes in the construction industry will be discussed in depth.

- Chapter 4: Discrete event simulation for generating construction schedules

Here, the principles and the application of the constraint-based discrete event simulation that is capable to generate diverging schedules for project scheduling problems such as the RCPSP will be introduced. After introducing the state of the art of the constraint-based discrete event simulation and the data preparation a new approach will be introduced that accelerates the preparation process for the simulation-based scheduling caused by the large amount of input data that has to be prepared before the start of the simulation. When the input data for the simulation is complete, the simulation can be started. The exact working mechanism of the simulation will be explained on the using an example.

- Chapter 5: Determination of float time with constraint-based discrete event simulation

To enable the simulation to determine float time for every single construction task also taking resource constraints into account, the newly developed backward simulation approach will be

introduced and discussed in detail. Necessary restrictions and limitations of the method will also be addressed.

- Chapter 6: Optimization of construction schedules

Optimization of schedules in both the construction and manufacturing industries is a great challenge. In the manufacturing industry, the shop problems hinder attempts to find the schedule with the shortest makespan. In the construction industry, the resource-constrained project scheduling problem hinders determining the schedule with the shortest makespan. After the introduction of the basic terms of optimization, state-of-art deterministic and heuristic solution approaches for the RCPSP will be introduced.

- Chapter 7: Simulation-based optimization of construction schedules

Since complex and large optimization problems have a huge search space of possible solutions, a new optimization strategy will be introduced that uses the introduced constraint-based discrete event simulation technique and that is able to traverse through the search space of the optimization problem in an efficient way. The application of this strategy will be presented on the basis of three different heuristic optimization approaches. To present the application of the introduced methods, two comprehensive case studies will be presented.

- Chapter 8: Summary and outlook

The last chapter summarizes the results of the thesis and gives an outlook for further possible research topics related to the introduced research work.

2 Scheduling problems and conventional scheduling techniques

2.1 Executive Summary

Due to the uniqueness of each construction project, every single project requires detailed planning. "Planning, in general, can best be described as the function of selecting the enterprise objectives and establishing the policies, procedures, and programs necessary for achieving these objectives. It determines what needs to be done, by whom and by when, in order to fulfill one's assigned responsibility" (Kerzner 2003).

The planning process begins by setting up the global objectives and restrictions of the project. This is followed by a scheduling process to establish a reasonable sequence of project tasks that meet the objectives and satisfy the restrictions of the project. As it was introduced in Section 1.4, scheduling is an iterative process that is primarily carried out manually. It begins with the fragmentation of the project into individual tasks. This can be achieved by creating a specific work breakdown structure (WBS) for the construction tasks, which is a tree subdivision of all the tasks that are necessary to complete the project (Kerzner 2003).

After the interrelationships between the project's tasks are defined, the approximate duration of every single task is determined. This is followed by the determination of a feasible schedule based on the duration of each task and the restrictions of the project. This schedule will be evaluated according to the objectives of the project, such as duration and costs. When the schedule does not meet these global objectives, the schedule must be revised and reconstructed. This can be achieved by modifying either the sequence or the duration of the tasks, usually achieved by varying the amount of available resources like adding extra resources to the project. This however influences the time-cost trade-off of the project. Applying further resources will increase the costs but also will accelerate the project completion time in the most cases and vice versa (see Section 1.3.1). Diverging feasible schedules can be generated by changing the sequence of tasks or modifying the amount of available resources.

Such modifications should be carried out until a schedule is found that meets all the desired objectives of the project.

Methods of determining feasible schedules that satisfy all these requirements will be the main topic of this chapter. First, the specific scheduling problems in the construction industry will be introduced including the resource constrained project-scheduling problem. This is followed by the description of the conventional scheduling techniques used in the construction industry, such as the Gantt or bar chart, linear scheduling- and network scheduling techniques (Kerzner 2003). Since the planning of construction schedules is mostly undertaken manually by a planner, the quality of the schedule depends on the planners' experience. This work is based on the thesis that an improvement of the planning process can be achieved by applying computer-aided methods, such as simulations, which have already been successfully used in the manufacturing industry. Therefore, the second part of this chapter will give an overview of the scheduling problems and the various solutions used in the manufacturing industry, like the disjunctive graph model. The chapter is closed by a comparison of the main characteristics of the two industries with the introduction of possible methods to apply simulation-based scheduling methods to the construction industry.

2.2 Scheduling problems and conventional solution techniques in the construction industry

Due to high construction costs and short due dates, detailed scheduling of construction projects is more important today than ever before. The goal of scheduling is to efficiently arrange the execution order of a multitude of construction tasks with regard to different objectives as introduced in Section 1.3 to save on both time and costs. There are various types of constraints that must be considered in the planning process of construction schedules. The most important ones are the resource restrictions, material availabilities and the technological dependencies, also called precedence relationships, among the construction tasks. To understand the complexity of the problem that needs to be solved, it will be introduced in detail in the following sections.

2.2.1 Specific scheduling problems in the construction industry

As introduced in Section 1.2, a construction project is always made up of t construction tasks ($T = (1 \dots t)$), with t symbolizing atomic construction tasks on the site. Those are connected with each other through interrelationships (precedence constraints) and require a predefined amount of specific resources (e.g. labour and machines). A task can be executed by any resource available on the construction site that are capable of completing the specific task. Each resource can often be used to execute several tasks. For example a laborer can assemble the formwork of an element. On another day, that same laborer can pour the formwork with concrete. The different *skills* that a resource must possess in order to execute one specific task can be described by k ($R_k, (k = 1 \dots l)$. For example, a laborer must have the necessary skill to "armour" a structure in order to complete the task of creating a formwork for a structural component. A specific resource might have more skills that are needed to complete a specific task and a task might need different kinds of skills to be completed. Furthermore, every task has a duration d_t.

A *general scheduling problem* in the construction industry is defined as the search for a sequence of the construction tasks that satisfies all the precedence relationships between the tasks without

taking resources into account. Such a problem can be solved by the *Gantt-chart* (Section 2.2.3), *line-of-balance* (Section 2.2.4) and *network scheduling techniques* (Section 2.2.5). When the resource needs of the tasks and resource limits on a construction site are also taken into account, a *scheduling problem under resource constraints* is defined (see Figure 2-1). This problem is mathematically more complex than the first one and cannot be solved correctly by the aforementioned scheduling techniques (Herroelen et al. 1997, Brucker et al. 1999). To determine a feasible schedule for this problem that satisfies every precedence and resource constraint, further methods must be taken into consideration. These methods will be introduced in Chapter 3. Providing an objective for this scheduling problem, such as finding the shortest makespan for the project, defines a combinatorial optimization problem called *resource-constrained project scheduling problem* (Artigues et al. 2010). This specific problem will be the topic of the next Section.

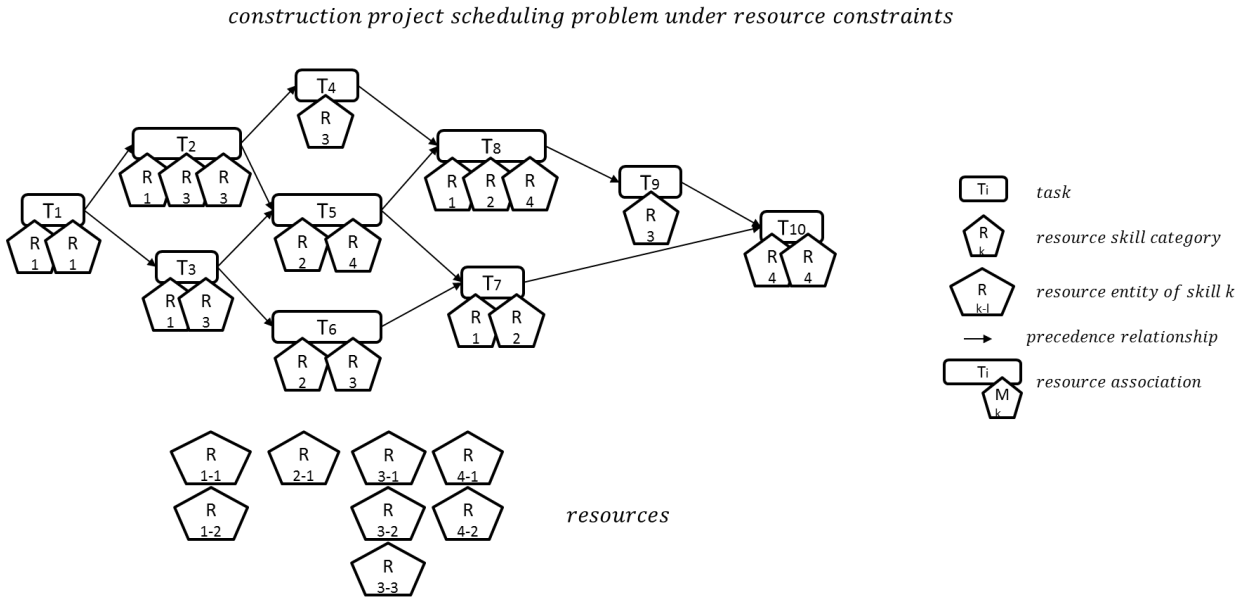


Figure 2-1: Schematic representation of a project scheduling problem under resource constraints. The tasks (T_i) are represented in a precedence graph with their necessary resources (R_k). The longer the box, the longer the duration of the task.

2.2.2 Resource-Constrained Project Scheduling Problem

In order to determine a realistic schedule, not only the precedence relationships between tasks, but also resource constraints need to be considered. Such a problem is described by the resource-constrained project scheduling problem (RCPSP) which has become the standard problem for project scheduling (Hartmann and Briskorn 2010). The RCPSP is a combinatorial optimization problem, which consists of finding the optimal schedule with the shortest makespan with respect to the precedence constraints between the tasks and the resource availabilities without preemption[3]. A formal definition of the problem is presented in Section 2.2.2.2. Further mathematical formulations will be introduced in Section 6.5.1.

[3] When the execution of a task has been started it must be executed until its completion without any interruption.

Blazewicz et al. (1983) have shown that the RCPSP belongs to the class of NP-hard optimization problems. This means that in a complex case it is not possible in polynomial time to find an optimal solution for the optimization problem (see Section 6.4). The RCPSP accurately describes the scheduling problem in the construction industry (Section 2.2.1). The main topic of this thesis is to find solutions for this problem. However, due to the NP-hardness of the problem, the author is not aiming to develop a deterministic method to find the actual optimal solution, but rather to generate feasible schedules (Figure 2-2) that satisfy all constraints and limitations of the RCPSP and improve upon them in an efficient way while still focusing on the goal of finding near optimal solutions.

2.2.2.1 Categorization of schedules for the RCPSP

To understand how a schedule can be described, the categorization of schedules will be introduced in this section. Sprecher et al. (1995) described four categories of schedules for the RCPSP. *Feasible schedules* belong to the first category where a result satisfies every precedence and resource constraint and never exceeds the resource limits. A *semi-active schedule* is a feasible schedule in which the tasks are pushed back in time while keeping the same schedule and task sequence. An *active schedule* is a feasible schedule where none of the tasks can be executed earlier without delaying another task (Sprecher et al. 1995, Hartmann and Kolisch 2000). The last category of schedules are the *non-delay schedules*. Baker (1974) defines a non-delay schedule as a feasible schedule in which "no machine is kept idle at a time when it could begin processing some operation". Under a non-delay schedule every task is executed as early as its precedence and resource constraints allow.

Non-delay schedules are a subset of active schedules, which are the subset of semi-active schedules and the feasible schedules as depicted in Figure 2-2 (Sprecher et al. 1995, Hartmann and Kolisch 2000). While the complete set of active schedules always contains the optimal schedule for the considered RCPSP, the complete set of non-delay schedules may not.

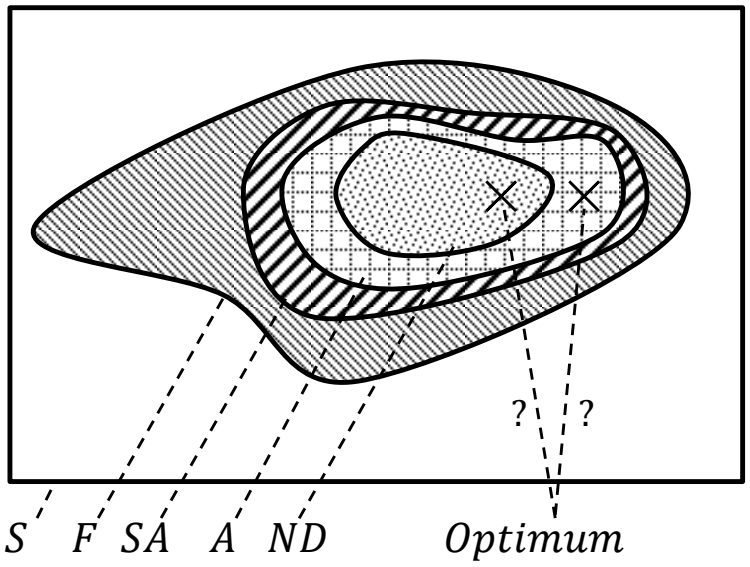


Figure 2-2: The categorization of the solutions of the RCPSP: S represents every possible solutions, F the feasible schedules, SA the semi-active schedules, A the active schedules and ND the non-delay schedules. Outside of F within S are the infeasible solutions that violate at least one constraint or limitation rule. The optimal solution (marked with X) is always in the set of the active schedules, however it might be outside of the non-delay schedules.

This categorization will have an important role later by evaluating the solutions of the developed simulation technique (Section 7.1.1).

Even though the RCPSP describes the project scheduling problem in a general way, it still cannot cover all of the situations that can occur in actual practice. Therefore many researchers have been working on new, more general, project scheduling problems, often based on the basic formulation of the RCPSP (Hartmann and Briskorn 2010). To order and compare these project scheduling problems, Brucker et al. (1999) and Hartmann and Briskorn (2010) integrated them into a survey. The major classes of the RCPSP are the single-mode case, the time-cost trade-off problems and the multi-mode case (Brucker et al. 1999). These cases will be introduced in the next sections. Further cases, such as the time lag problems, scheduling with preemption, scheduling multiple projects, cases with non-regular objective functions (resource leveling problem and net present value problem) and problems with stochastic task durations, are described in Brucker et al. (1999) and Hartmann and Briskorn (2010) in detail.

2.2.2.2 Single mode case of the RCPSP

The single mode case of the RCPSP is the basic formulation of the optimization problem. There are $T = \{0,1 \dots t, t+1\}$ tasks that should be scheduled, where task 0 and t+1 are fictitious tasks for the beginning and the termination of the project. Furthermore there are k = 1,…, K resources with the amount of R_k units available. Every task has a fixed duration d_t. The tasks might be connected to other tasks (predecessors of task t are stored in P_t) with precedence relationships and may need a specific kind and/or amount of resources (r_{tk} units of resource k) for their execution. The single mode RCPSP only uses renewable resources, meaning that the resource is available from the start until the termination of the project. When starting a task, the required resources will be reserved and cannot be picked up by any other tasks. When a task is completed, all of its resources will be set free and they can then be picked up by other tasks. Preemption (interruption) of tasks is not allowed. Once a task has been started, it will be completed within the predefined duration. The objective is to find a makespan-minimal schedule that satisfies all the precedence and resource constraints and resource limitations (Brucker et al. 1999). A mathematical model for the problem has been developed by Pritsker et al. (1969).

2.2.2.3 RCPSP as time-cost trade-off problem

In the single mode RCPSP, the duration of the tasks is predefined. However, the duration of a single task might be changed depending on "how much the planner is willing to pay for it" (Brucker et al. 1999). Therefore, in this generalized form of the problem, the single mode RCPSP is extended with a nonrenewable resource[4]: the budget of the project. The more nonrenewable resource that is allocated to a task, the faster its processing time becomes. The objective of the optimization is either to find the schedule with minimal makespan subject to a fixed upper boundary of the budget (the budget problem), or to find the schedule with minimal budget subject to a highest boundary of makespan (deadline problem). The nonrenewable resource is often measured in money, so these problems are also referred to as time-cost trade-off problems (Brucker et al. 1999).

[4] Nonrenewable resource: A fixed amount of resource is available for the entire project makespan. When a nonrenewable resource is used, the amount of available resources will decrease through the frequency of usage.

This problem has been discussed in detail by Kelley and Walker (1961), Demeulemeester et al. (1996), Ranjbar et al. (2008), Hartmann and Briskorn (2010) and Zhou et al. (2013) amongst others.

2.2.2.4 Multi-mode case of the RCPSP

As introduced in the last sections, in the single mode RCPSP the duration of a task and the necessary resources to reach this duration are fixed and therefore only represent one single method for executing a task. In a multi-mode RCPSP the tasks have different alternatives for their execution. An alternative execution mode is represented by a combination of resource configuration (renewable and nonrenewable) and the corresponding task duration. The task must be performed in only one of its modes. Once a mode has been selected and started no changes or preemptions are allowed. The objective is to find the schedule with the minimal makespan. When only one execution mode is available for the tasks and there are no nonrenewable resources included in the project, the problem is the single mode case of the RCPSP (Hartmann and Briskorn 2010).

The solution for the multi-mode RCPSP without nonrenewable resources has been discussed by several researchers including Hartmann (2001), Bouleimen and Lecocq (2003), Jarboui et al. (2008) and others. Further formulations and literature about the multi-mode RCPSP can be found in Hartmann and Briskorn (2010).

Since the single mode RCPSP is already an NP-hard problem, and the time-cost trade-off and multi-mode problems are even more complex, the author will only focus on generating feasible solutions for the single mode problem. Hence in the introduced examples only renewable resources will be considered and no preemption of tasks will be allowed.

In the next Sections the existing conventional scheduling methods that are capable of determining feasible schedules for the general scheduling problem will be introduced. The next Chapter will address the issue of why simulation-based scheduling can be advantageous compared to the conventionally used methods, and how discrete event simulation can be used to generate feasible solutions for the RCPSP. After introducing the existing methods for determining the optimal or near optimal solutions for the RCPSP in Chapter 6, a new technique will be introduced that uses an algorithm to steer the discrete event simulation and generate diverging feasible schedules in order to find near optimal solutions.

2.2.3 Gantt chart

The Gantt chart, also called the bar chart, was one of the first scheduling methods applied in project management and is still the most frequently used method in the construction industry. The first ideas (Gantt 1903) of the method were developed by Gantt, an American mechanical engineer and management consultant, contemporaneously with the method of Taylor (Taylor 1903) and dated from 1890. The two methods should be considered jointly as an integrated production planning and control system (Wilson 2003). The initial version of Gantt´s planning method was a tabular approach. The current chart form was introduced later (Gantt 1919). The visualization of the project schedules through a clearly arranged diagram, in which the tasks themselves are represented as bars over the temporal axes “provides a quick and easily understood means for describing project activities” (Wilson 2003).

The position of the bar within the chart is described by the execution dates of the task: the beginning of the bar symbolizes the start date and the end of the bar symbolizes the termination date

of the task. Extending the representation of the bars with colors, significant additional information such as criticality, in progress, completed, necessary resources, etc. can be assigned to the tasks. With secondary bars, float time can also be represented. Putting further information on the temporal axis such as capacity, resource needs and costs extends the chart with a further dimension (Burghardt 2007). Placing every task of the project into a Gantt chart provides information about which tasks should be executed on a specific day or within a desired time interval (Figure 2-3).

Due to the clear representation of the project, this method is still widely used for project management purposes and therefore also to plan construction schedules. The most commonly used software programs are MS Project and Primavera. However, the major disadvantage of this method is its inability to clearly show the interrelationships between the tasks leading to difficulties in both rescheduling within the project and in controlling the costs of the project (Kerzner 2003).

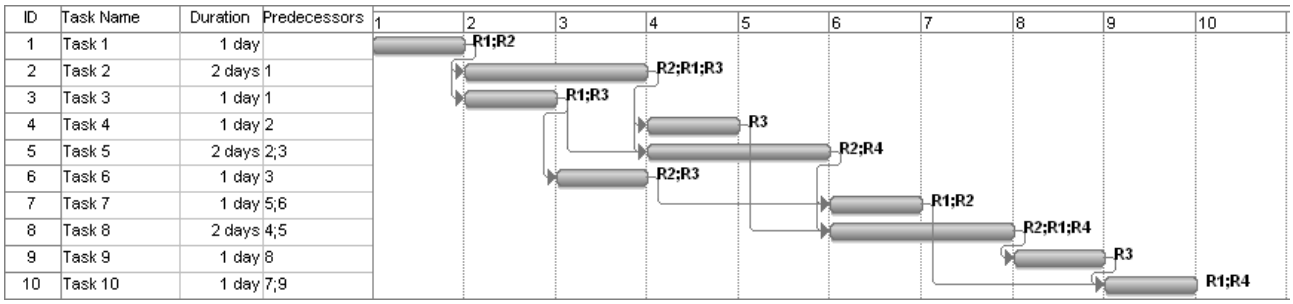


Figure 2-3: Gantt chart representation of the introduced construction project in Figure 2-1. The bars represent the planned execution time interval of a task. Behind the bars the for the tasks execution necessary resources are illustrated. The interrelationships between the tasks are hardly traceable. (MS Project 2007)

2.2.4 Linear scheduling or line-of-balance method

The linear scheduling method, also called line-of-balance method, is a two dimensional graphical representation of the tasks' progress over time. The horizontal axis represents the change of time and the vertical axis represents the completeness of the task. A task is represented by a line. The start and end point represent the start and termination date of the task. Intermediate points on the line not only characterize the relative completion of the task, but also its location on the construction site. Hence the line-of-balance representation of a project illustrates both the temporal and the spatial distribution of the tasks and their interrelationships. This is useful to detect temporal and spatial bottlenecks within the schedule. This method is suitable for scheduling operations of a repetitive nature (like processes in the manufacturing industry, housing projects, high-rise buildings or large bridges), and for linear construction projects like pipelines, highways and tunnels. In the latter case, the linear partition of the construction site will be represented on the secondary axis (Figure 2-4). The first application of the line-of-balance technique was in the manufacturing industry to achieve and evaluate production flow rates for production lines (Al Sarraj 1990). The method was introduced by the Goodyear Company in the 1940s and was developed further by the US Navy in the 1950s (NAVMAT 1962). Lumsden (1968) and Khisty (1970) were the first to apply this method of scheduling to the construction industry. Since construction projects such as high rise buildings and bridges contain many non-repetitive processes including excavations, foundations and superstructures (O´Brian 1975), nowadays this technique is principally used only for the scheduling linear construction projects (Long and Ohsato 2009).

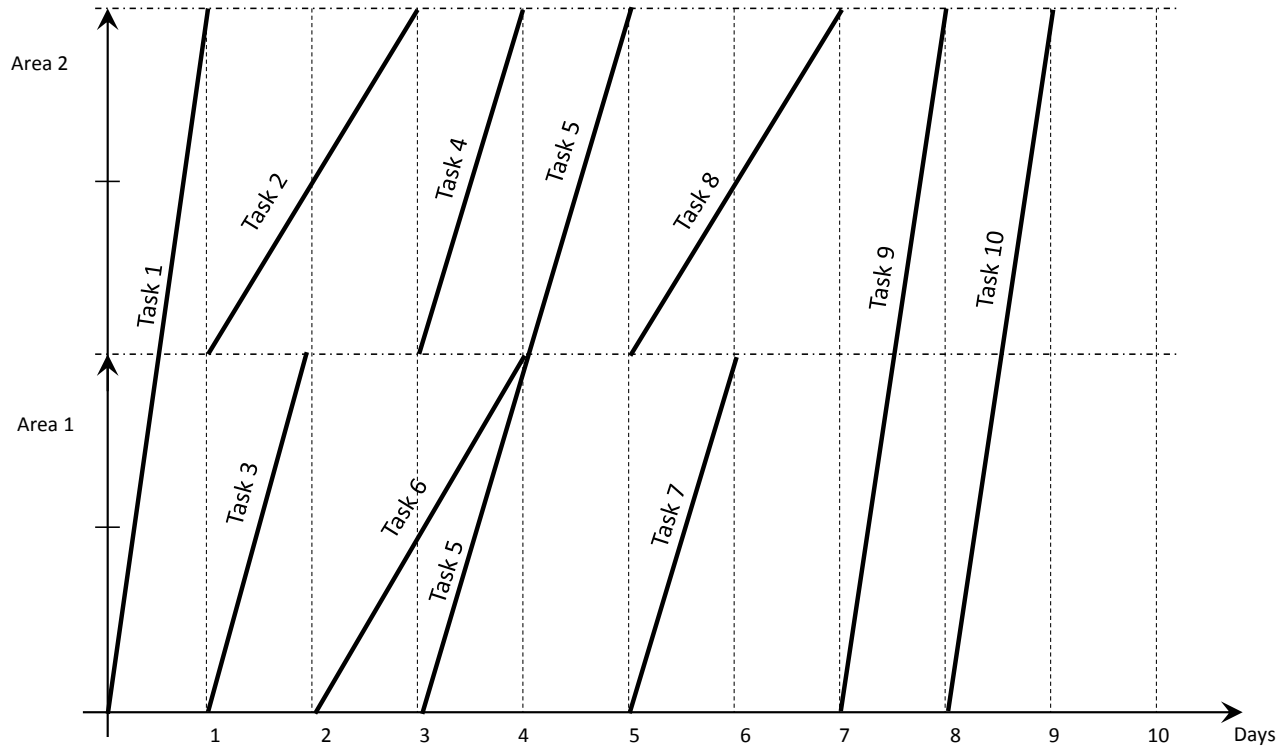


Figure 2-4: Linear schedule representation of the introduced construction project from Figure 2-1. The construction site is divided into two regions: area 1 and area 2. The execution direction of every task is forward from the start point to the end of area two. There are tasks, such as task1, task 5, task 9 and task 10, which are executed continuously in both areas and the other tasks that are executed only in one area (e.g. task 2 in area 2 and task 3 in area 1).

The advantage of this technique, besides the spatial and temporal representation of the project, is the illustration of the production rate for the individual tasks which can be determined as the slope of the production line. This is important because the duration of the project can be shortened by setting the production rate of simultaneously executed tasks to a similar rate by changing their resource needs (Figure 2-5). Using this technique, it is simple to adjust production rates for individual tasks and therefore to create a smooth and efficient flow of resources for the project (Arditi et al. 2002) and decrease the idle time for the machines and labor (Long and Ohsato 2009).

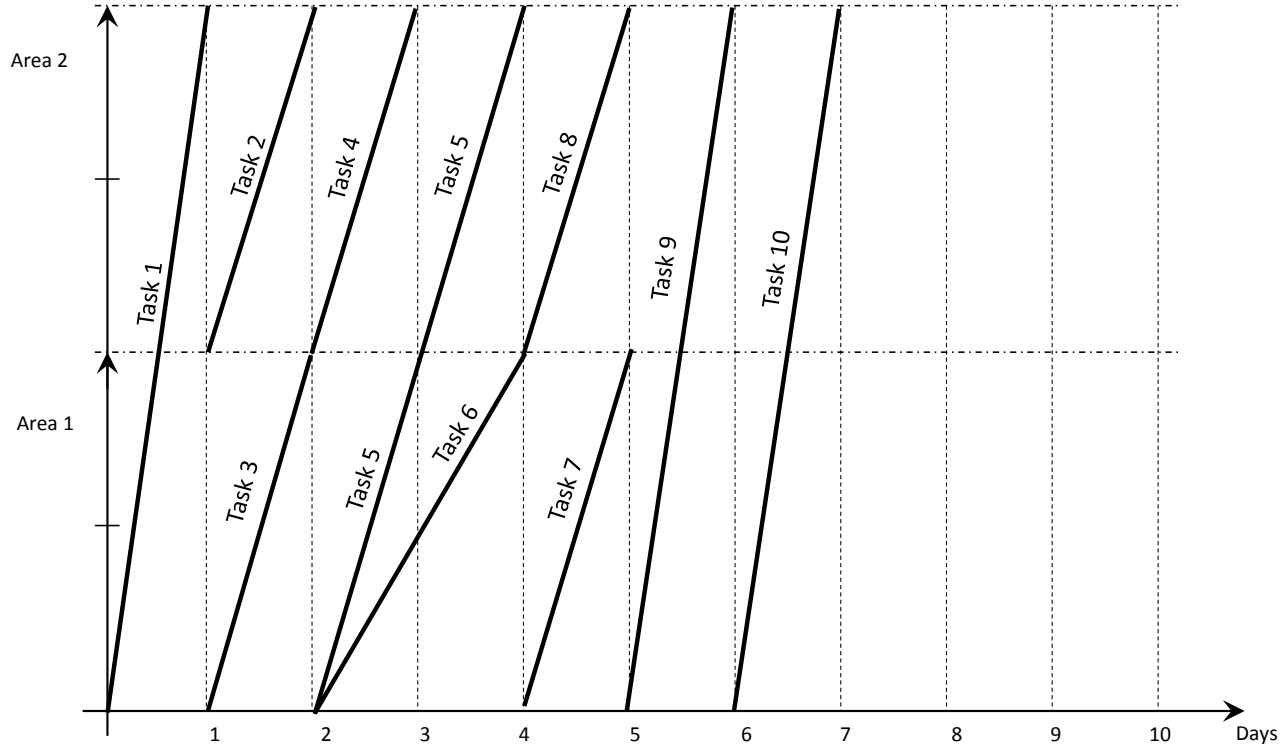


Figure 2-5: Linear schedule representation of the introduced construction project in Figure 2-1. Task 2 and task 8 have been accelerated (duration 1 day) in order to get more parallel execution lines and a more compact schedule.

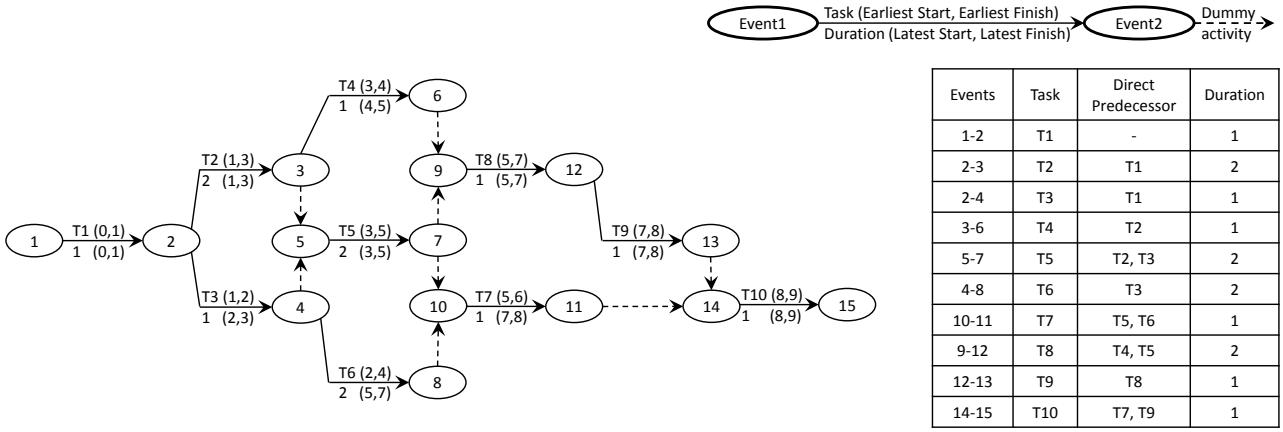
2.2.5 Network scheduling techniques

The best suited technique to visualize interrelationships between tasks is a *network* (Kerzner 2003). Since the emergence of the network scheduling techniques in the 1950´s, they have become the most popular methods in modern project management. This is due to their theoretical accuracy and adaptability in a wide range of fields such as construction engineering (Yang and Wang 2010).

2.2.5.1 Network representation of projects

Networks are graphs that provide a mathematical representation of the project. By visualizing the network, its structure can be extended by further information such as the start and termination date of a task. A network consists of *events* and *activities*. Events characterize the start or end points of tasks and activities specify the work that has to be done between these two events (Kerzner 2003). Networks can be represented in two different kind of forms. The first form is the *activity-on-arrow diagram* (AOA). In this diagram, the nodes symbolize the events and the connecting arrows between them symbolize the activities (Figure 2-6). The second representation form is the *activity-on-node diagram* (AON), where, corresponding to the name, the nodes symbolize the activities and the arrows symbolize the interrelationships between the activities (Figure 2-7). The former AOA diagram representation was widely used by project managers in the construction industry until the emergence of the latter AON diagram in the 1960´s. However, the AOA diagram is still in use although some

deficiencies of the method have been identified, like its inability to illustrate connected tasks and its ability to only represent finish-to-start interrelationships between the tasks. In order to show more complex relationships (e.g. start-to-start, or more predecessors for one activity), ”dummy" activities must be introduced that have zero duration and unfortunately make the network look complex and confusing (see Figure 2-6). In contrast, the latter AON method is more flexible in this sense and can represent every major type of interrelationships (start-to-start, start-to-finish, finish-to-start and finish-to-finish). Since the activities are nodes of the network, the main information and data about the activity (e.g. start and finish date, duration, etc. – see Figure 2-7) can be placed inside the nodes as text providing more definite representation of the project than the AOA diagram where the text is placed on the arrows (Figure 2-6). Furthermore, the AON diagram can easily be converted into a Gantt chart representation. This process is not as simple for an AOA network due to the dummy activities. A detailed comparison of these two network diagrams is provided by O’Brian and Plotnick (2009) and Yang and Wang (2010).



Events	Task	Direct Predecessor	Duration
1-2	T1	-	1
2-3	T2	T1	2
2-4	T3	T1	1
3-6	T4	T2	1
5-7	T5	T2, T3	2
4-8	T6	T3	2
10-11	T7	T5, T6	1
9-12	T8	T4, T5	2
12-13	T9	T8	1
14-15	T10	T7, T9	1

Figure 2-6: Activity-on-Arrow representation of the introduced construction project in Figure 2-1. Ellipses: Events, Arrows: tasks, dashed arrows: dummy activities.

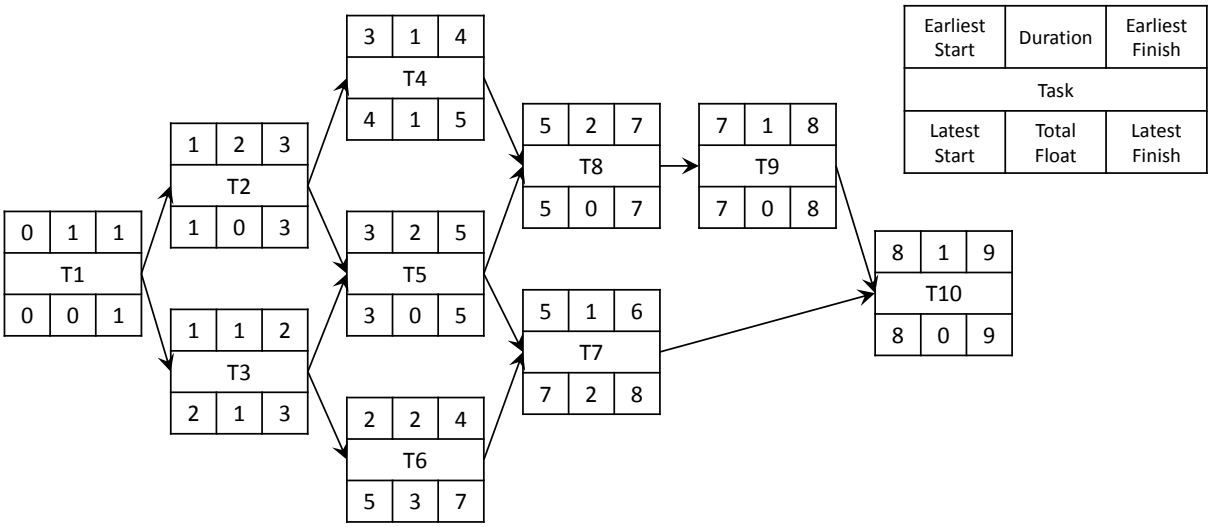


Figure 2-7: Activity-on-Node representation of the introduced construction project in Figure 2-1. Boxes: tasks, Arrows: precedence relationships. Information about the tasks like earliest or latest start time are represented inside the tasks box.

There are three similar and commonly used scheduling techniques that are based on the aforementioned two network representations. Two of them, *the Program Evaluation and Review Technique* (PERT) and the *Critical Path Method* (CPM), are based on the AOA network and the *Precedence Diagram Method* (PDM) is based on the AON network. The former two techniques (PERT and CPM) were developed at the end of the 1950´s and aimed at solving large scheduling problems where the Gantt chart seemed to be inapplicable. An important feature of the AOA network representation is that interdependencies between activities can easily be visualized and modified. A further feature of these both methods is the determination of early and late start dates for each task. Via these parameters the float time for individual tasks, and so the critical path (see Section 1.3.5), of the project can be determined. Thus, temporal bottlenecks and critical processes within the project can be identified. Before introducing these three network scheduling techniques in detail, the general approach of how to determine a schedule with the network scheduling techniques will be discussed.

2.2.5.2 Determining a schedule with network scheduling techniques

To determine a schedule, all three aforementioned network scheduling techniques use the same methodology. After fragmenting the project into the necessary tasks and determining their predicted duration, a *forward pass* is carried out, where the earliest possible start date of the tasks is determined (see Figure 2-8). To keep the explanation of the method simple, the author will not use any complex interrelationships or delays between the tasks, just the common finish-to-start precedence relationship. The forward pass always moves forward in time. Therefore it starts with the first tasks of the project that have no predecessors and schedules them at the start date zero. The earliest termination date of a task can be calculated by adding that task´s duration to the earliest start date of the task. The earliest start date of a task is defined by the latest start date of all its predecessors´ earliest termination date, since a task cannot be started before all its predecessors are completed. When the earliest finish date of the last task in the schedule is determined, the forward pass and so the schedule is complete.

The key advantage of the network scheduling techniques over the Gantt chart method is the capability of determining not only the earliest start and termination dates of an activity, but also the latest ones. This can be achieved by the *backward pass*. The backward pass works in a similar manner as the forward pass, but it proceeds backward in time and therefore it starts with the last task in the project (see Figure 2-9). After the forward pass is carried out, the earliest termination date of the last task in the project is detected. This spot forms the start point of the backward pass. To avoid delays in the project, the latest termination date of the last task of the sequence equals with its earliest termination date[5]. The latest start date of a task can be determined by subtracting its duration from its latest termination date. The latest termination date of a task is defined by the earliest of all its successors´ latest start date, similar to the earliest start date determined for the forward pass. The calculation continues until the latest start time for every task is determined and the initial point of the project is reached (O'Brien and Plotnick 2009).

[5] Otherwise the makespan of the project would be extended

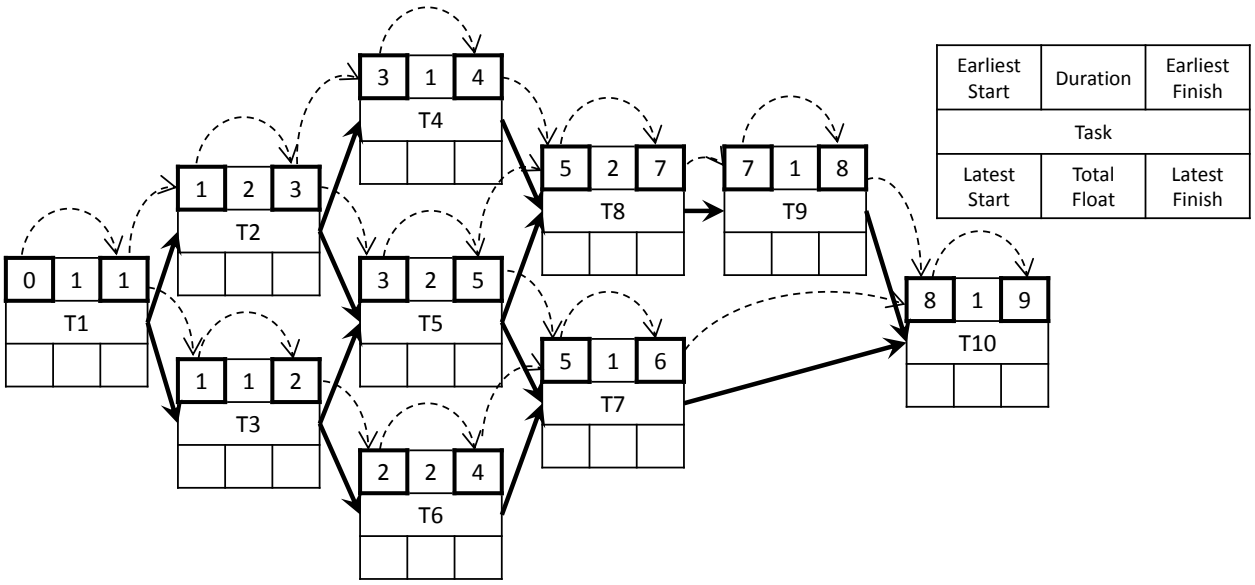


Figure 2-8: Schematic representation of the forward pass for the construction project introduced in Figure 2-1. Determination of the earliest start and finish date of the tasks. Boxes: tasks, arrows: precedence relationships, dashed arrows: steps of the calculation.

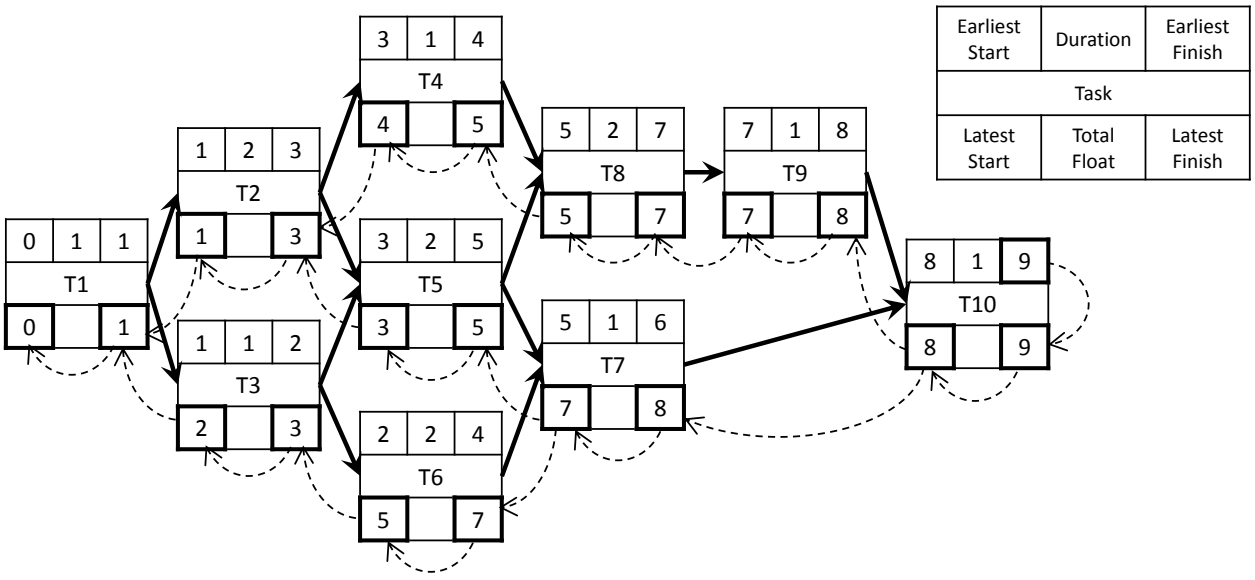


Figure 2-9: Schematic representation of the backward pass for the construction project introduced in Figure 2-1. Determination of the latest start and finish time of the tasks. Boxes: tasks, arrows: precedence relationships, dashed arrows: steps of the calculation.

The difference between the latest and the earliest start dates of the tasks define the *total float* of the task which represents how much time reserve exists without there being an impact on the makespan of the project. The tasks without a total float time constitute the *critical path* of the project. A project can have several critical paths simultaneously. A delay in any of these tasks will result in an increase of the project's overall makespan and therefore a delay of the project itself. For this reason, it is important to determine the float time for each task so that it is possible to identify tasks that are part of the critical path and tasks that only have very short float times.

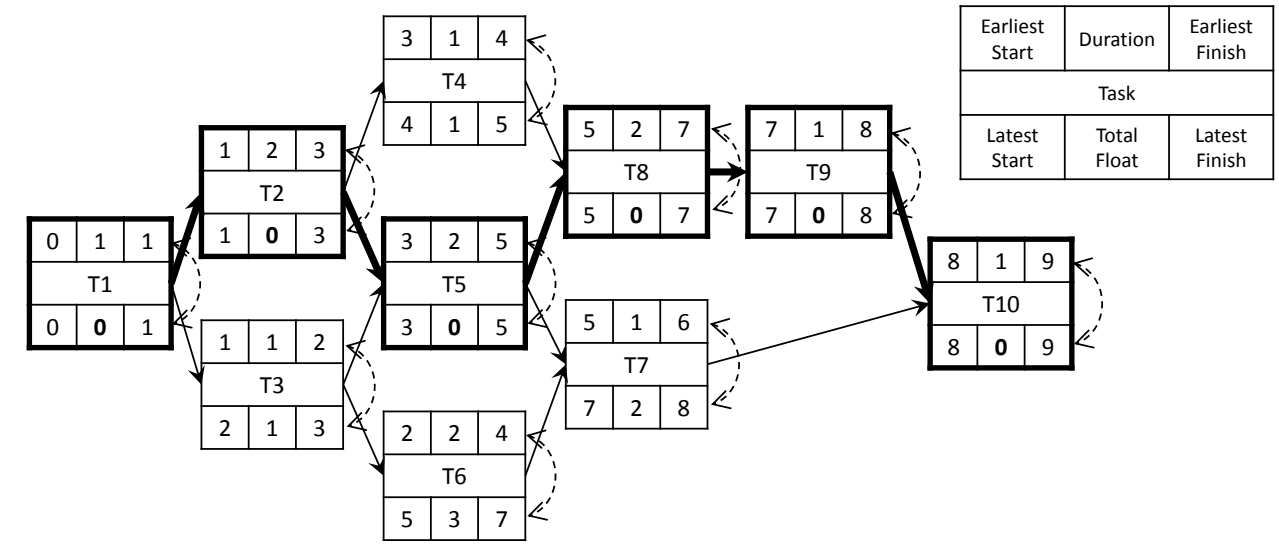
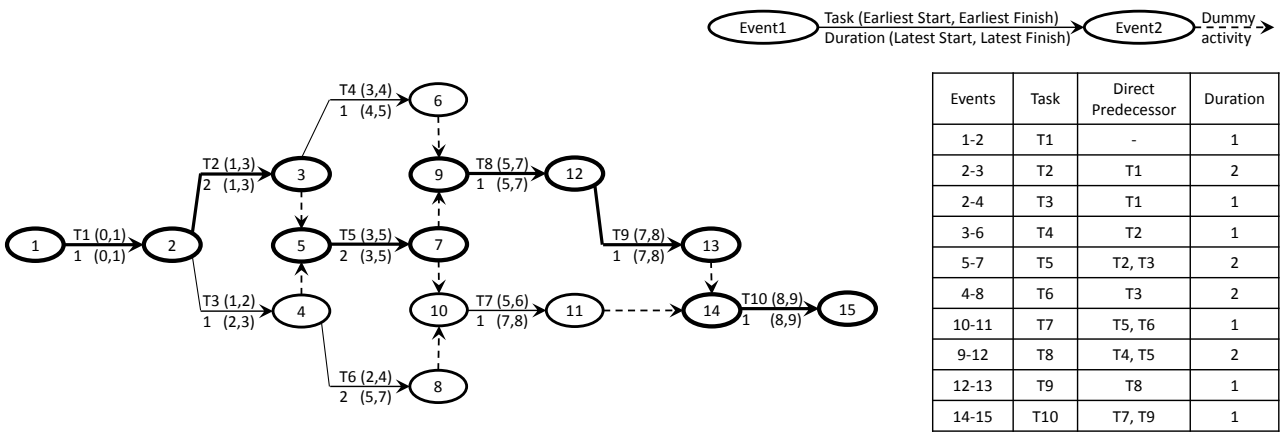


Figure 2-10: Determination of float time and critical path for the construction project introduced in Figure 2-1. Boxes: tasks, arrows: precedence relationships, dashed arrows: steps of the calculation, fat arrows: critical path, fat boxes: critical tasks.

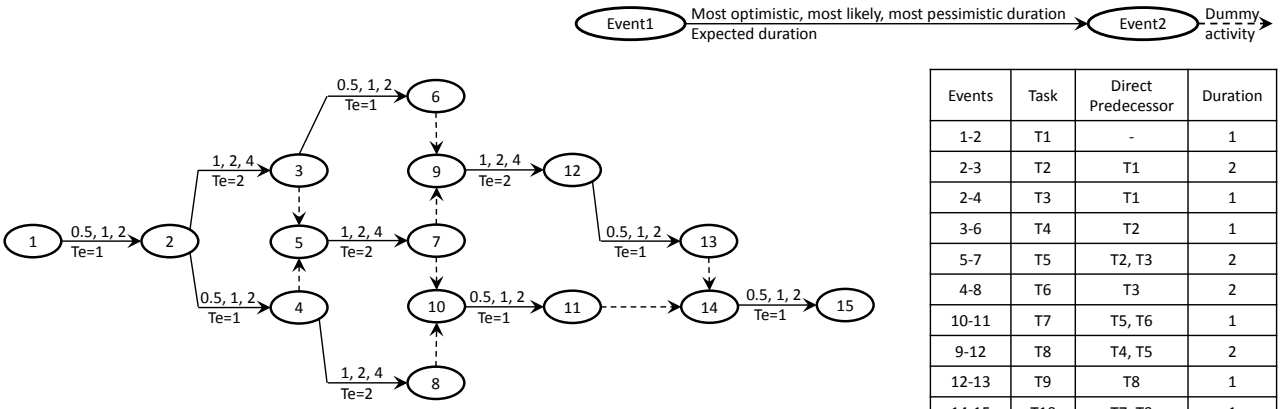


Events	Task	Direct Predecessor	Duration
1-2	T1	-	1
2-3	T2	T1	2
2-4	T3	T1	1
3-6	T4	T2	1
5-7	T5	T2, T3	2
4-8	T6	T3	2
10-11	T7	T5, T6	1
9-12	T8	T4, T5	2
12-13	T9	T8	1
14-15	T10	T7, T9	1

Figure 2-11: Representing the critical path of the construction project introduced in Figure 2-1 on an AOA representation based CPM network. Ellipses: events, thick ellipses: critical events, arrows: tasks, dashed arrows: dummy activities, thick arrows: critical tasks.

2.2.5.3 Program Evaluation and Review Technique (PERT)

The Program Evaluation and Review Technique (PERT) was developed in the 1950's by the US Navy to plan the Polar Missile Program in order to complete it in the shortest possible time (Stretton 2007). To determine a schedule, in addition to the general scheduling technique introduced in Section 2.2.5.2, PERT uses three time estimates for the expected task´s duration: the most optimistic, the most likely and the most pessimistic one. For example, let's assume that the most likely completion time of a task is three days. If everything goes well, the task can be finished in two days. However, if there are unexpected difficulties, it can take up to five days to complete the task. The determination of these time estimates are based on probabilistic distributions which allow the PERT to also calculate the odds of completing the project before a specific deadline (Kerzner 2003). A standard PERT network is presented in Figure 2-12.



Events	Task	Direct Predecessor	Duration
1-2	T1	-	1
2-3	T2	T1	2
2-4	T3	T1	1
3-6	T4	T2	1
5-7	T5	T2, T3	2
4-8	T6	T3	2
10-11	T7	T5, T6	1
9-12	T8	T4, T5	2
12-13	T9	T8	1
14-15	T10	T7, T9	1

Figure 2-12: AOA-based PERT representation of the introduced construction project in Figure 2-1. On the arrows PERT uses three task duration estimates: the most optimistic, the most likely and the most pessimistic one. Below each arrow is the expected duration of the task.

2.2.5.4 Critical Path Method (CPM)

Simultaneously with the development of the PERT, another similar scheduling technique, the Critical Path Method (CPM), was developed by the DuPont Company. Their goal was to create a schedule with a better time-cost trade off (Kelley and Walker 1989), instead of finishing the project as soon as possible. As opposed to PERT, the CPM uses only one time estimate for the tasks (as represented in Figure 2-6) and since this estimate is not connected to any probabilistic values, the method is considered as deterministic in its nature. The determination of the schedule is carried out according to the general technique described in Section 2.2.5.2. Since CPM only uses one time estimate for a task, the estimate must be accurate enough to create a suitable schedule. Therefore, the CPM is primarily used in the construction industry where the companies have detailed databases concerning the completion time of the different tasks with specific conditions. When an accurate completion time of a task is available, an accurate partial completeness of the task can be determined. This can be used to evaluate the schedule during the project as the task is being executed (Kerzner 2003).

In contrast, PERT is used on those projects, such as research and development, where the completion time of a task cannot be accurately predicted and therefore the partial completeness of a task is almost impossible to determine. This kind of schedule can only be evaluated when a task has

been finished and when a milestone[6] was reached. Therefore, PERT can be seen as an event-oriented method and CPM can be seen as an activity-oriented one. Since both methods use the AOA representation of the schedule in order to introduce complex task interdependencies, dummy activities must be used.

2.2.5.5 Precedence Diagram Method (PDM)

The Precedence Diagram Method (PDM) scheduling technique was developed by J. Fondahl and was first published in 1961 (Fondahl 1961). He tackled the same time-cost trade-off problem as DuPont and the US Navy. To address this problem he developed a new representation form and scheduling method called the activity-on-node network and Precedence Diagram Method (Fondahl 1987). The computerized version of the PDM was created by H.B. Zachry Co. of Texas and it was then commercialized by IBM (Snyder 1987, Weaver 2007). The path of identifying the schedule is the same as described in Section 2.2.5.2, but the PDM uses the AON network representation instead of the AOA.

Since its emergence, the usage of the PDM has increased because of its clear advantages over the traditional PERT/CPM. Due to the AON network-based representation that avoids the necessity of dummy activities and allows the usage of complex task interrelationships, PDM diagrams are easier to draw and even complex schedules can be represented clearly with this program. Since it is more flexible than the earlier programs, it can provide a more realistic and accurate modeling of the project (Wiest 1982).

Extensions for the interrelationships are the *lead-lag factors*. These represent a duration that can be attached to the task interrelationships, indicating a minimal time period by which the start or termination of a task leads or lags the start or termination of another task (Wiest 1982, Kerzner 2003). For example, for a concrete structure the task of removing the formwork cannot be started until one day after the concrete pouring task has been completed (because of curing). This is represented by a finish-to-start relationship with a one-day lag.

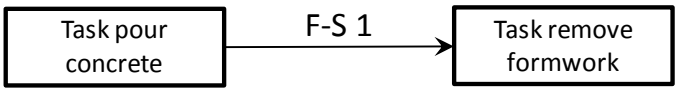


Figure 2-13: Representation of lag in a PDM network: the task of removing the formwork can only be started one day after task of filling concrete has been finished (Finish-Start relationship with 1 day lag).

2.3 Scheduling problems and solution techniques in the manufacturing industry

In the manufacturing industry similar scheduling problems exist to those in the construction industry. This section will focus on these scheduling problems and on different solution methods how these problems can be solved. The designers of the manufacturing industry have developed many different methods to address the specific scheduling problems of this industry (Brucker et al. 1999). In this context the disjunctive graph model and the simulation-based scheduling methods will be

[6] Milestones are special events in the schedule where key segments, packages or phases of the project are completed.

introduced in the next sections in detail, since they are relevant for the author's research work. The simulation-based scheduling technique is the topic of Chapters 3, 4 and 5. Further static methods, like integer programming, will be introduced in Chapter 6.

2.3.1 Specific scheduling problems in the manufacturing industry

The most common scheduling problems in the industrial production process are the shop scheduling problems, such as the *open shop problem*, the *flow shop problem* and the *job shop problem*. These are special cases of the *general shop problem* and are widely used for modeling industrial production processes (Brucker 2007).

The general shop problem is defined by n jobs $i = 1, \dots, n$ and m machines $M_1, \dots, M_m$. Every job consists of a set of operations O_{ij} $(j = 1, \dots, n_i)$ and has a specific duration of p_{ij}. Each operation must be processed by one of the above-defined machines, and precedence relationships must be defined among the operations of all jobs. In the case of the general shop problem the machines are dedicated, which means that each job must be processed on a specific machine. Furthermore, "each job can only be processed by only one machine at a time and each machine can only process one job at a time" (Brucker 2007). The objective of a shop scheduling problem, similar to the RCPSP, is to find an optimal schedule with a minimal makespan.

In contrast to the general shop problem, the RCPSP allows a task to be processed by any resource or any combination of resources that possess the necessary skill to complete the task. This represents the main difference between manufacturing processes and construction industry processes and explains why shop problems do not represent construction projects correctly. Yet another difference between the shop problem and the RCPSP is that for the RCPSP precedence relationships can be defined between every tasks, while for a shop problem precedence relationships can only be defined between operations of the same job.

Based on these similarities and differences, it is recognizable that the shop problems are specific formulations of the RCPSP (Brucker et al. 1999), however these problems are still NP-hard in the strong sense (Blazewicz et al. 1983). Thus, a shop problem can be defined as an RCPSP where a task only needs one specific resource for its execution and the precedence constraints are defined only between tasks that belong to one job. Since the RCPSP is the generalized form of the shop problem, the solution methods of the RCPSP can be applied without restrictions to the shop problems but not vice versa. To apply the solution methods of the shop problems to solve the RCPSP, they must be extended first (see Section 2.4.1).

The first special case of the general shop problem is the open shop problem. In the case of an open shop problem, there are no precedence relationships defined between the operations. Therefore, the objective is to find the best execution sequence of the operations belonging to the same job (e.g. $O_{11} \rightarrow O_{14} \rightarrow O_{12} \rightarrow O_{13}$). Also to find the best order of operations that must be processed on the individual machines (e.g. for M_3: $O_{25} \rightarrow O_{32} \rightarrow O_{31} \rightarrow O_{21}$, see Figure 2-14) while obtaining the minimal makespan of all jobs.

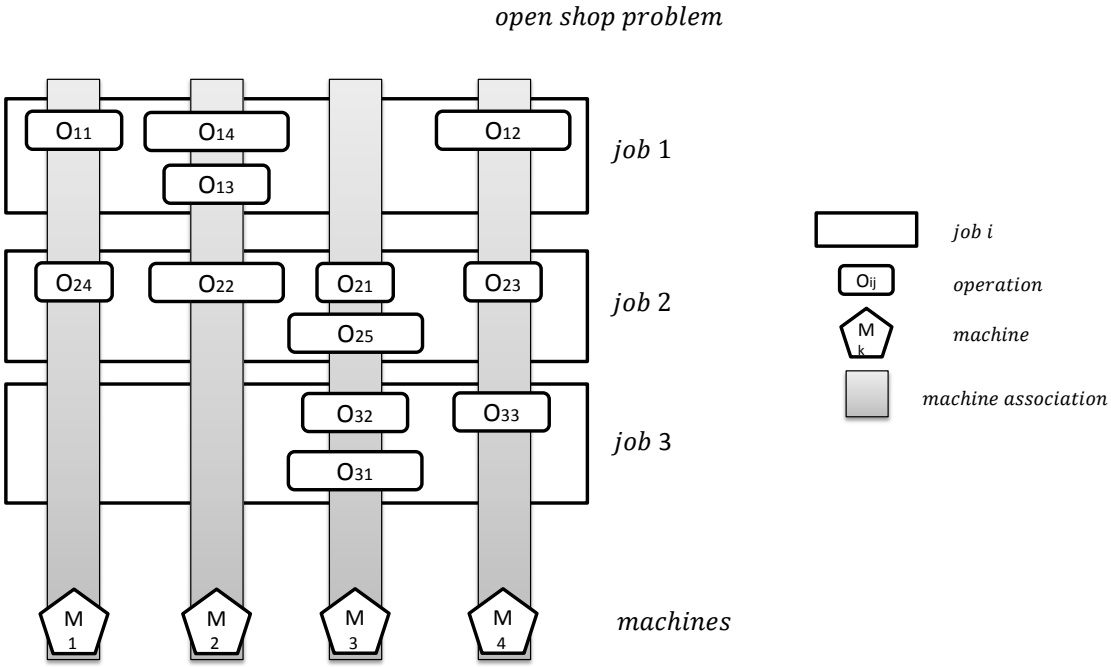


Figure 2-14: Schematic representation of an open shop problem example: the operations (O_{ij}) of a $job\ i$ have no precedence relationships between each other and are associated with one machine (M_k) necessary for their execution.

In the flow shop problem, each job consists of m operations $O_{i,j}$ that must be processed on the machine M_j. The precedence constraints between the operations are defined after the principle $O_{i,j} \rightarrow O_{i,j+1}$ $(j = 1, \dots, m-1)$ for each $i = 1, \dots, n$. Thus, every job must be processed on each machine and the number of operations for every job equals with the amount of machines. The objective is in this case to find the optimal job execution order for every individual machine M_j (e.g. $Job_2 \rightarrow Job_1 \rightarrow Job_3$, therefore the execution order for M_1: $O_{21} \rightarrow O_{11} \rightarrow O_{31}$, see Figure 2-15).

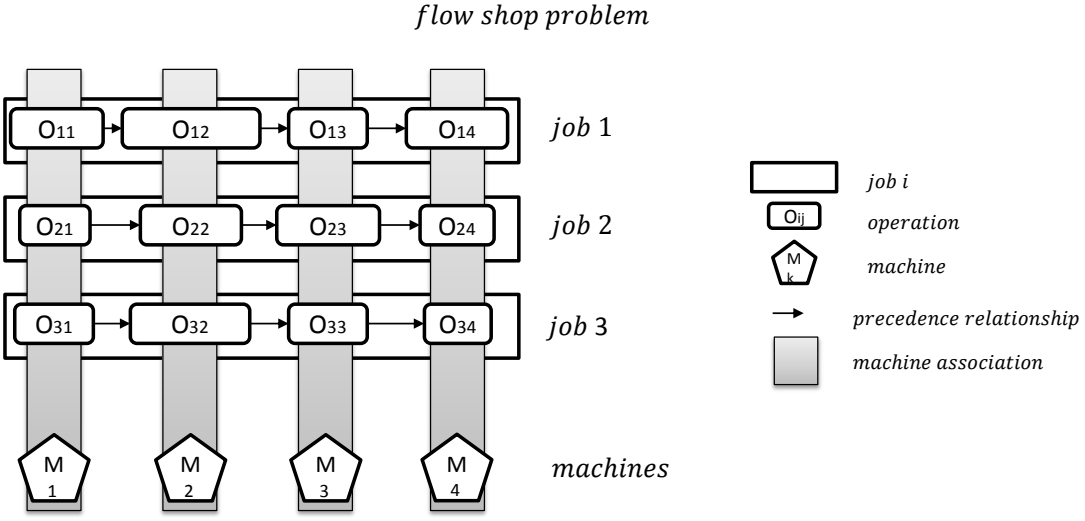


Figure 2-15: Example for a schematic representation of a flow shop problem: the *m* operations (O_{ij}) of every job are connected as a flow with precedence relationships to each other. Every task is executed on the machine corresponding to the position of the operations in the flow sequence. The task is to find the optimal execution order for the jobs.

The job shop problem (see Figure 2-16) is the generalization of the flow shop problem with a modified resource-job/operation restriction. In the job shop problem every job i consists of a sequence of n_i operations: $O_{i,1}, O_{i,2}, \dots, O_{i,n_i}$, which must be processed precisely in this order. Therefore the precedence constraints should be formulated as follows: $O_{i,j} \rightarrow O_{i,j+1}\ (j = 1, \dots, n_i - 1)$. Every operation $O_{i,j}$ needs a resource $r_{i,j} \in (M_1, \dots, M_m)$. Hence, every operation must be processed by one predefined machine, but there is no correlation between the position of the operation within the job to the selected machine position (e.g. in a flow shop problem operation 1-3 must be processed by M3. In the job shop problem it can be any other, predefined machine, e.g. M2). The objective is to find the optimal schedule with the minimal makespan (Brucker 2007).

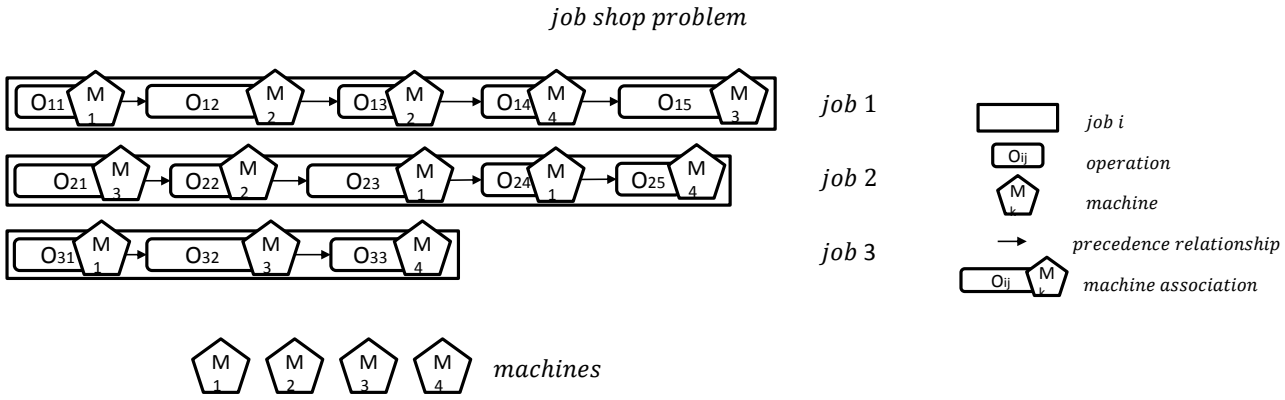


Figure 2-16: Schematic representation of a job shop problem: the jobs are made up of different (n_i) amounts of operations (O_{ij}) that are connected to each other with precedence relationships as described by the flow shop problem. For the job shop problem the ID of the required machine for a task's execution is not necessarily equal to the task's place in the flow sequence.

2.3.2 Disjunctive graph model

To determine diverging feasible schedules for a shop problem, the disjunctive graph model (Roy and Sussmann 1969) is one of the methods used most (Blazewicz et al. 2000). The methods popularity is not only due to its ability to generate feasible schedules, but it can also be used for optimization purposes when the objective function of the optimization is regular (Brucker 2007). A disjunctive graph $G = (V, C, D)$ is defined as follows:

- V: Vertices represent the operation of a job. The operations are weighted according to their processing times.
- C: represents the *directed conjunctive arcs* of the graph that are the precedence relationships between the operations.
- D: contains the *undirected disjunctive arcs*. Such arcs exist between two operations when they are not connected by a chain of conjunctive arcs but need the same machine to be processed and therefore they cannot be executed simultaneously.

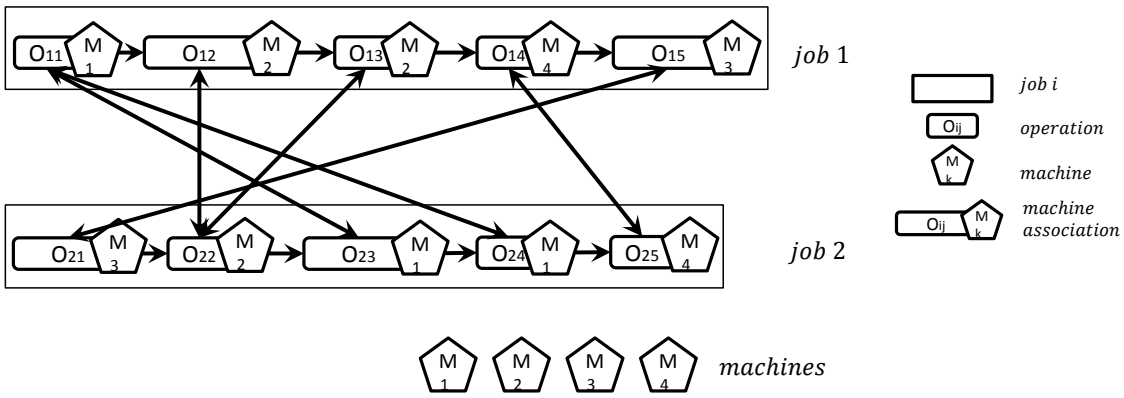


Figure 2-17: Disjunctive graph model of the first two jobs of the introduced job shop problem in Figure 2-16. Simple arrows: conjunctive arcs, double arrows: undirected disjunctive arcs.

The basic idea for scheduling is to turn the undirected disjunctive arcs into directed conjunctive arcs and define straightforward sequences for every machine and operation in this way. The schedule is complete when every disjunctive arc has been turned into a *fixed arc*, and the resulting graph is *acyclic*. By varying the direction of the fixed arcs, diverging schedules might be generated (Figure 2-18 and Figure 2-19). By generating a schedule for every possible fixed arc direction, the complete feasible solution space of the project will be covered and the optimal schedule lies within the determined schedules. This optimization technique is called *total enumeration* which will be discussed in detail in Chapter 6.

A heuristic optimization method for the different shop problems using the disjunctive graph model has been introduced by Mati (2010). He uses a feasible base schedule where all the disjunctive arcs have been turned into fixed arcs. Likewise he attempts to improve the schedule by swapping the direction of fixed arcs that belong to the critical chain. Brucker et al. (1994) developed a Branch-and-Bound method-based (see Section 6.5.1.2) deterministic search tree algorithm. No disjunctions are fixed in the root node. The successor of a node always contains one more fixed arc than the previous node. The examination of a node terminates if all of the disjunctive arcs are fixed or if it can be proven that the node does not contain the optimal solution (Brucker et al. 1994). Abdelmaguid (2009) enhanced the aforementioned Branch-and-Bound method. He introduced the permutation-induced acyclic network (PIAN) where the decision variables are defined as the permutations of jobs instead of the binary decision variables associated with the disjunctive arcs.

Since the disjunctive arcs forbid the simultaneous execution of task pairs that are not allowed to be executed simultaneously and assuming that the disjunctive graph model is acyclic, the method will always result in feasible schedules.

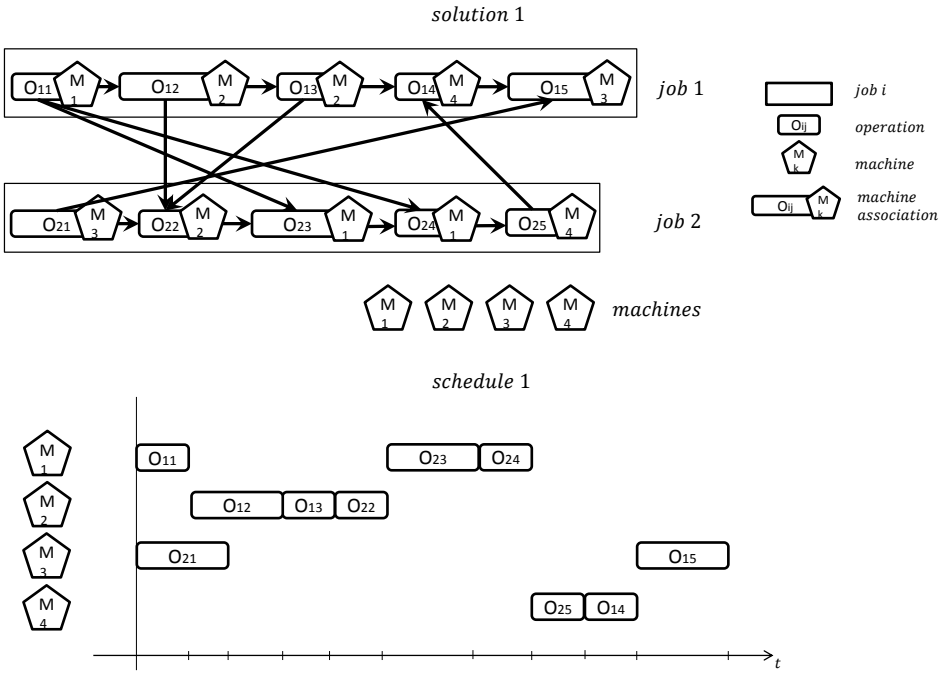


Figure 2-18: One feasible solution for the introduced disjunctive graph from Figure 2-17. Every disjunctive arc has been turned into a fixed arc resulting in an acyclic graph defining exact execution orders of the operations for every single machine.

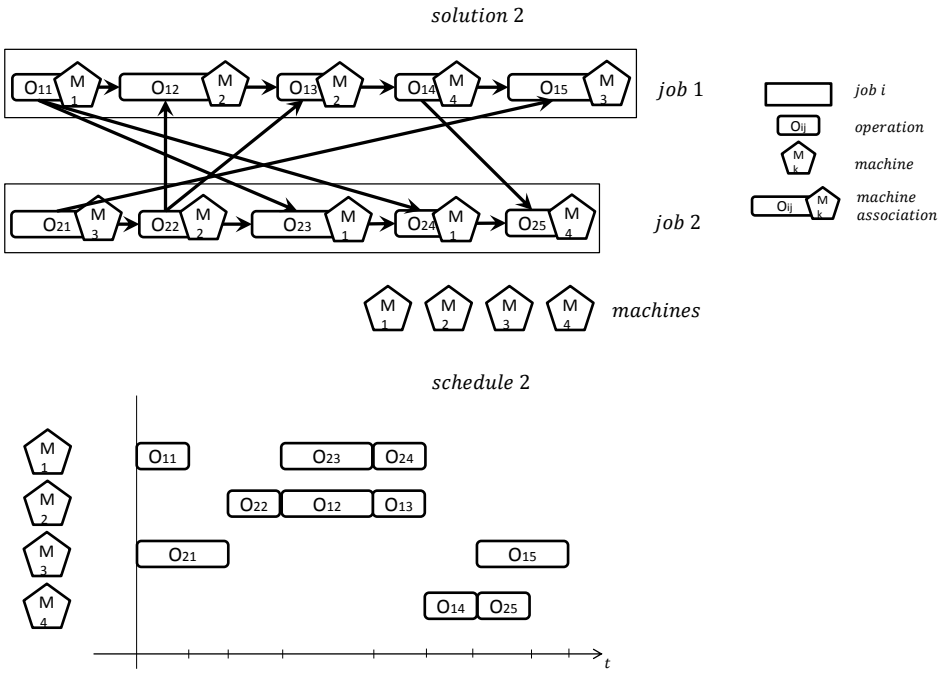


Figure 2-19: Another feasible solution for the introduced disjunctive graph in Figure 2-17. By comparing the resulting feasible schedules of solution 1 and solution 2 it can be seen that schedule 2 is significantly shorter than schedule 1.

The generation of diverging schedules by varying the direction of the fixed arcs is presented with two solutions (Figure 2-18 and Figure 2-19) for the introduced disjunctive graph from Figure 2-17. Solution one (Figure 2-18) results in a relatively long schedule, since the idle time of the individual machines is long. In the second solution (Figure 2-19) the direction of the fixed arcs are directed so that it allows the simultaneous execution of more operations thus resulting in a shorter schedule makespan and less idle time for the machines.

2.4 Improving the scheduling techniques in the construction industry

As described in Section 2.2, the common scheduling methods in the construction industry are not capable of solving problems with resource constraints. The available methods, which have been developed for the manufacturing industry, are not applicable for construction processes, since they do not take the flexible assignment of resources to tasks based on description of skills into account.

In order to improve the scheduling processes of the construction industry instead of using the conventional manually executed methods, computer-aided techniques should be applied. The goal is to improve the scheduling processes so that they can handle large and complex projects, to reduce the calculation period to determine near optimal schedules and to raise the validity of the generated schedules. One promising technique is the simulation-based scheduling technique. However, before introducing this technique it is important to understand the difficulties encountered in applying this technique to the construction industry. Therefore, in the next sections, the differences between the main characteristics of the two industries and the relations between the main scheduling problems (the resource-constrained project scheduling problem and the shop problems) will be discussed.

2.4.1 Comparing the scheduling problems and solution techniques in the construction and the manufacturing industry

Analyzing the scheduling problems in both construction and manufacturing industries while taking also resource constraints into account allows us to identify both some similarities and differences. The most general similarity is that both the construction project-scheduling problems and shop problems under resource constraints are made up of processes (called either tasks or operations) that might have interrelationships with other processes and all of them need resources for their execution. These similar foundations of the problems and the recognition that both problems contain repetitive operations[7] encouraged researchers (Halpin 1977, Martinez and Ioannou 1994, Wales and AbouRizk 1996, Herroelen et al. 1997, AbouRizk and Hajjar 1998, Lu 2003, Blazewicz et al. 2007, König et al. 2007b, Tulke and Hanff 2007, Günthner and Borrmann 2011) to attempt to apply the same methods to solve these problems or to adapt solution methods from one industry to another.

However, an adaption is not always a straightforward process and its difficulties lay mostly in the differences between these problems. In a shop problem "each machine can handle at most one job at a time and each job can be executed by at most one machine at a time. Thus, at any time, the execution of a job is restricted by the presence of a single scarce resource (Blazewicz et al. 2007)."

[7] Such as earth transport in the construction industry or the preparation of a product in the manufacturing industry

In contrast, if the scheduling problem of the construction industry is considered as a resource-constrained project scheduling problem (RCPSP), a task shall be executed at least by one resource (machine and labor) and it can be executed by any of the available resources that have the necessary skills. Thus, dependencies are not defined between the task and a resource itself, but between the task and the skill category that a resource must have in order to execute the task.

This additional freedom creates a mathematically more complex constraint description between task and resource in the RCPSP than in the shop problem, where the connection is defined directly between a task and one specific resource. Since mathematically the foundations of the two problems are the same and the differences are only the boundary conditions (e.g. precedence and resource constraints), it can be proved that the resource-constrained project scheduling problem is the *general formulation* of the shop problems (Blazewicz et al. 1983). Thus, a shop problem can always be formulated as an RCPSP, but not vice versa.

This is achievable by defining the jobs and their operations as independent sequences of tasks, where the tasks within a job are connected to each other with precedence constraints. Every task needs only one skill that can be executed by only one specific resource entity (machine) for its completion.

Furthermore, through this classification, every solution technique that is able to solve the RCPSP can also be applied to solve the shop problems without restrictions, but not vice versa. To solve the RCPSP with methods that have been developed to solve a shop problem, first, the methods must be extended so that they can handle the more general precedence and resource constraints that were introduced above. As a consequence, the RCPSP is mathematically more complex and harder to solve than the shop problems (Blazewicz et al. 1983, Herroelen et al. 1997).

2.4.2 A drawback of conventional scheduling techniques – consideration of resources

A very significant drawback of the introduced conventional scheduling techniques in the construction industry (Section 2.2) is the missing capability to take resource constraints into account. To overcome this drawback, either an extension of these techniques or the usage of a new method is necessary. A simple extension for the common scheduling techniques is to manually push tasks forward in time along the time axis until all resource constraints are met (Figure 2-20). This is a time consuming iterative process that is inefficient in complex cases where the amount of tasks and interrelationships is high. This is due to the many infeasible schedules. Further challenges include the decisions of which tasks to push forward in time, since pushing different tasks forward results in different schedules with different makespans and costs.

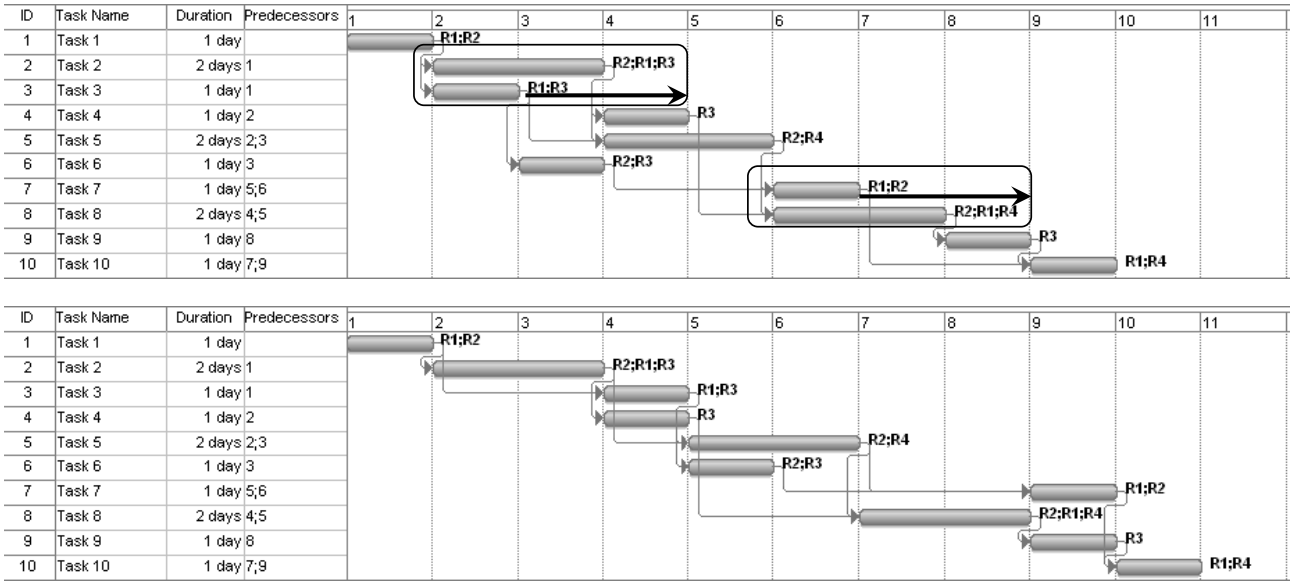


Figure 2-20: Pushing tasks forward along the time axis due to resource shortage. Above: basic schedule, below: updated schedule. There is only one R1 resource available on the site. Therefore either task 2 or task 3 and task 7 or task 8 which form the basic schedule must be pushed forward in time, since they exceed this resource limit. The author's choice was to push task 3 and task 7 forward resulting in a schedule with a makespan that is one day longer than the basic one, but where the resource limits are all met. (Project from Figure 2-1)

Another possible approach to overcome the resource allocation problems while keeping within the predefined resource limits is the application of heuristic methods for the scheduling process (Hegazy 1999, Lu and Li 2003). These algorithms rank the tasks according to certain rules, like priority, earliest start time or other project specific values and then attempt to schedule the tasks in their ranking order while not exceeding any of the resource limits (Willis 1985). As further development of this technique, Lu and Lam (2008) introduced a method that is also able to take resource calendars into account for those occasions when the resources are only available on the construction site within a predefined time window. Although, these methods also facilitate resource constraints into account with common scheduling techniques, due to their complexity and time consuming preparations they are barely used in the industry.

A suitable method for industrial application should not only be able to generate schedules by taking resource constraints into account but should also be easy to manage and should have a low computational time. The approach taken in this thesis follows the principle that such a method should be adapted from another industry, e.g. from the manufacturing industry, where the method has already proven its applicability.

Since the process-simulation technique has already proven its effectiveness and efficiency for schedule generating purposes with resource constraints in the manufacturing industry, an adaption of this method could accelerate the conventional scheduling process in the construction industry. Although the process simulation method is not an optimization method it can be coupled with heuristic optimization techniques, in order to determine near optimal schedules. Furthermore, resources can be defined and handled on a simple and straight forward way. The basic terms of simulation and its applications in both manufacturing and construction industry are introduced in the next chapter.

3 Simulation-based scheduling

3.1 Executive summary

Simulation is an experimental technique that analyzes the reproduction of a real or imaginary process or system to explore its behavior over time. Simulation models are used to observe, predict and study how a process or a system reacts to parameter changes (e.g. performance factors, layout, etc.) and to investigate the reasons behind the specific simulation results. Further applications aim to design new models while exploring alternative configurations or to optimize specific parameters of the model. Since simulations are always abstractions of a real problem, they can be applied for problems with any complexity, where other techniques may fail (Banks 1998, Page and Kreutzer 2005). Therefore, the process simulation is also applicable for scheduling purposes with resource constraints. The first section of this chapter will explain the basic definitions, modeling concepts and components of the process simulation. After introducing and comparing current applications of simulation in the manufacturing and construction industries, a different simulation approach will be presented that is capable of not only evaluating schedules but is also able to generate schedules for construction projects. The advantage of this technique over conventionally used scheduling techniques such as the network planning methods is its capability to take not only the interrelationships between the tasks into account but also to factor into the schedule the applicable resource constraints. Thus, it can overcome the major limitations of network planning techniques. The exact concept and working mechanism of this process simulation will be introduced in detail in Chapter 4.

3.2 Basic terms of simulation

Simulations are created to study the behavior of a *system* with desired *objectives,* e.g. how the system reacts on parameter or layout changes, or how to reach an optimal productivity with the predefined shop layout. A complete simulation-based study, in most cases, is composed of several *simulation runs* or *simulation experiments* that differ from another in some *simulation parameters*. A simulation run is always executed on a *simulation model* that represents the abstraction of a *system to study* and refers to the objectives of the simulation. The computer-based simulations are executed by *simulation programs,* where the simulation models and experiments are prepared and influenced by the *user,* who is also the *observer* of the simulations. Based on the observations about the behavior of the simulation model, conclusions to the real system can be drawn (Figure 3-1).

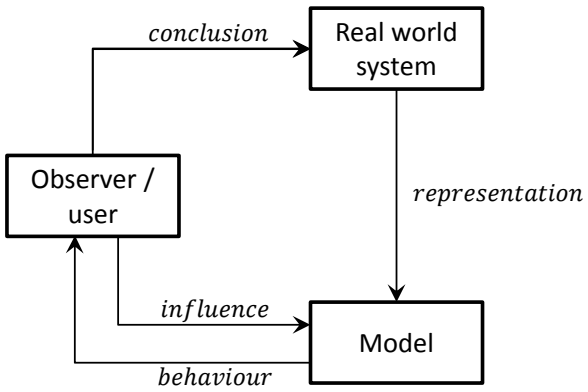


Figure 3-1: Interaction of the user, the system and the model (based on Page and Kreutzer 2005)

3.2.1 The system and its components

A system is a subset of the real world whose behavior is being analyzed. It is composed of a group of objects and elements (e.g. machines, tasks, labor, etc.), which are the *entities*. They might be joined with each other by *interdependencies,* by creating a sequence of operations, which define the *system structure* (Figure 3-2). The *attributes* are the properties of the entities that describe their characteristics such as quality, quantity, capacity, etc. (Banks 1998). In case of a construction project, for example, the entities are the machines, the building components and the construction tasks. The attributes of the building components are its material, its completeness or the deadline for completion.

Changes that can affect the systems behavior might also occur outside of the system. This outside area is called the *system environment* and it is separated from the system itself by the *system boundary*. The position of the boundary between the system and the environment depends on the purpose of the study (Banks and Carson 2009).With a construction project, for example, the transport of materials and building parts is considered as part of the environment and so their influence on the construction itself is considered as low. However, when this influence is significant, it might be part of the system boundary. In this way boundary *input* or *output parameters* might be exchanged within the environment, such as the arrival dates of the materials in the case of the construction project.

A system may not only be described by its components, but also by further characteristics. One such characteristic is the time dependency of the system. When the system does not contain any time

reference, it is called a *static system* (e.g. the system of a bridge, that's static behavior does not change over time). When the behavior of the system is time-dependent, it is called a *dynamic system*. A dynamic system has different *system states* that are defined by the value of all system *state variables* at any given point in time. *System state variables* are the collection of all information necessary to describe the state of the system at a certain point of time. This information can contain properties or attributes of system entities. The necessary state variables for a system are the function of the purpose of the study. So, one variable may be considered as a system state variable in one case but not in the other, even though the physical system is the same (Banks 1998). The *behavior* of the system is defined by a vector of the system states over time.

An *event* is an "instantaneous occurrence that may change the state of the system" (Banks and Carson 2009). Furthermore, *activities* are defined as time intervals with a specific length. They might be triggered and terminated by an event. For a construction project, the states might be represented as the completeness of the construction (state 1: not started, state 2: in construction, state 3: completed), which can be defined by a state variable. An activity might be the construction of the building that lasts e.g. 100 days long. The events that are connected with this activity are the "start the construction" and the "finish the construction". The former one might change the system state variable, for the completeness of the project, from "not started" to "in construction", and the latter one to "completed".

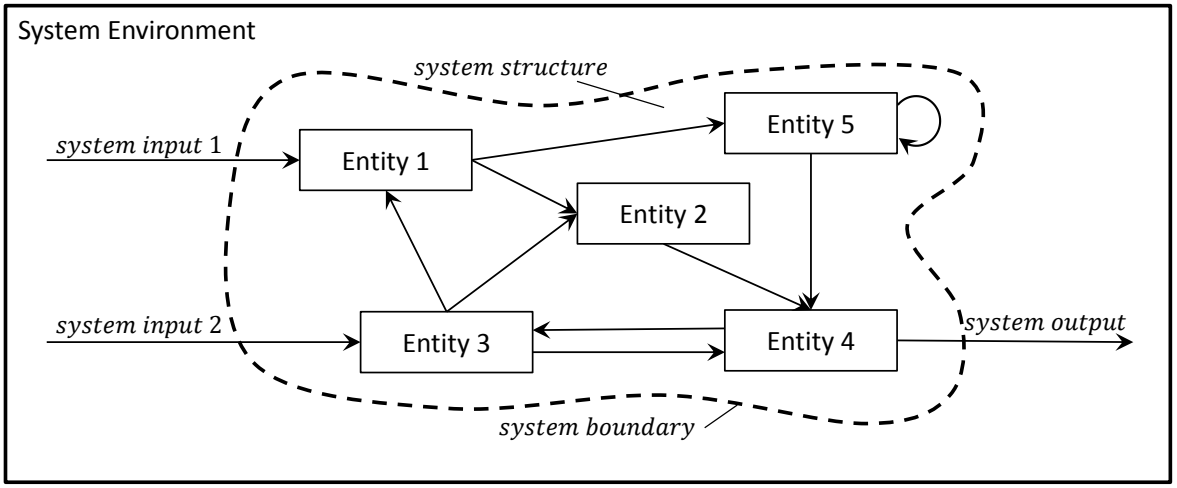


Figure 3-2: Representation of the system environment, the system boundaries and the system with its entities and their interactions. (Based on Page and Kreutzer 2005)

Based on the interaction of the system with its environment two further system categories can be differentiated. While a *closed system* (e.g. an aquarium - Page and Kreutzer 2005 - Figure 3-3) has no interactions with its environment, an *open system* (e.g. a manufacturing plant or construction site) has at least one interaction point (see Figure 3-2 above).

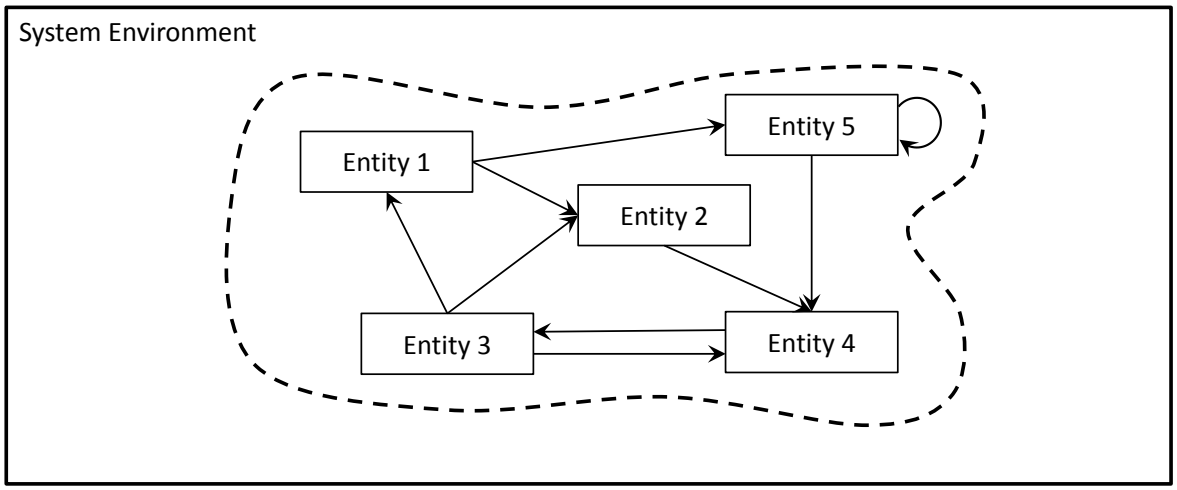


Figure 3-3: Representation of a closed system that has no interaction at all with its environment.

A further categorization of a system is that the system is either discrete or continuous. Discrete systems have state variables that only change their values at a discrete set point in time. For example the introduced small construction project can be considered as a discrete system, since the state of the project only changes when an event occurs (Figure 3-4).

The state variables of a continuous system change, as its name says, continuously over time. An example is the temperature of a machine in a factory. During production the machine warms up and after a temperature limit is reached a cooling process starts until a normal temperature is reached again. Figure 3-4 illustrates the change of the state variable, the temperature of the machine, over time for this continuous system.

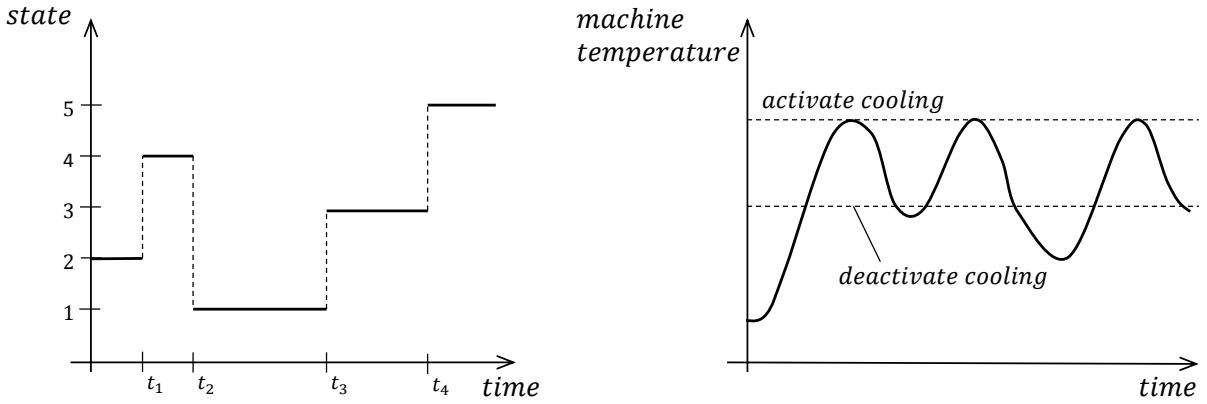


Figure 3-4: Representation of a discrete (left) and a continuous system (right)

3.2.2 Model concepts

A *model* is the representation of the system being studied within a selected examination framework. Models help explain the system's behavior and the effect of the interactions among its components (Page and Kreutzer 2005). The real world systems are mapped into models by *abstraction* and *simplification* of their components and functionalities. Therefore, a model is always simpler than the original system. However, it should always be complex enough "to answer the

questions raised" (Banks 1998) related to the system, but simple enough to understand the system's behavior and to keep the simulation efficient. Which system components are crucial and which are ignorable depends on the purpose of the model. For a construction project, for example, it must be decided at which level of detail the single components of the building are modelled. Is it enough to consider coarse objects, like the abutment or superstructure of a bridge, or should it be more precise and consider a more detailed structure, like the sub-foundation, foundation, the walls of the abutment or the individual girders of the superstructure? It only depends upon what question the model aims to answer. For an approximate calculation in the planning phase of the building the former version might be enough. However, for a detailed analysis or calculation in the latter phase of the building a more detailed model is necessary.

Simulation models can be categorized as static or dynamic, discrete or continuous and deterministic or stochastic (Figure 3-5). Static simulations, as static systems, are independent of time (Banks and Carson 2009). Dynamic simulations analyze how systems behave, respond or change over time. Dynamic simulation is, for example, the simulation of the production line of a factory to find out how many goods it can produce a day.

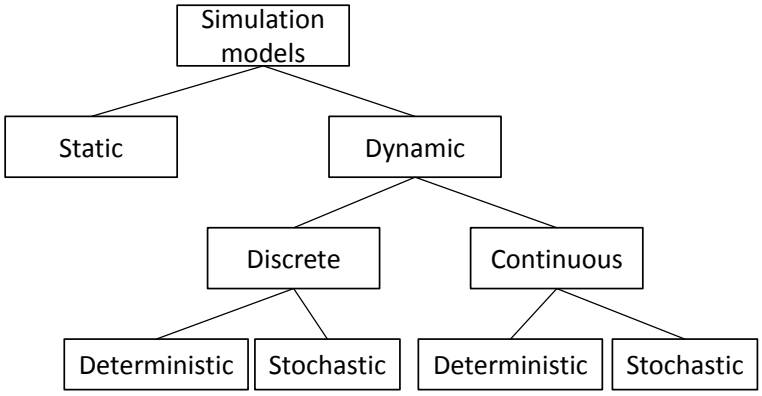


Figure 3-5: Categorization of simulation models (based on Page and Kreutzer 2005)

Deterministic simulations contain no random variables, have known input data and result in a unique set of outputs (Banks and Carson 2009). As an example, a simulation of a factory with input materials and machines is considered deterministic when every arrival date of the materials is known and the processing times of the machines are all predefined. Different results might be determined by diverging factory layouts. Stochastic simulations have at least one random input variable that leads to random simulation outputs. Since the results of the simulation are affected by a random variable that only approximately describes the modelled variable, they can only estimate the characteristics of the real systems behavior (Banks and Carson 2009). For the factory simulation model both the arrival date of the materials and processing time of the machines can be replaced with random distributions, which will result in diverging results for the same factory layout. Results of such a simulation would be the average production rate of goods a day, or the average idle time of specific machines.

Discrete and continuous models can be defined analogous to the discrete and continuous system introduced in Section 3.2.1. However, a discrete system is not necessarily modelled as a discrete simulation model, nor the continuous system as a continuous simulation model. Discrete systems might be simulated as continuous models and continuous systems as discrete models. Which model to choose, depends on the characteristics of the system and the purpose of the study (Banks and Carson 2009). Thus, the material transport on a construction site, for example, might be modelled discretely (states: stored, transported) when the transport itself is insignificant for the whole construction simulation. Although, if this process might collide with other processes and could cause

delays in the system, it is appropriate to model the transport with a continuous simulation model so that the location of the transport goods can be tracked down at any time.

3.2.3 Steps of a simulation study[8]

To analyze the behavior of a real system with simulations, different steps must be followed in order to gain reliable results in an efficient manner. These steps are part of the modeling cycle (Figure 3-6).

The first step of the modeling cycle is the *problem formulation*. Before starting to implement or model anything, a clear statement about the problem to be analyzed must be declared (e.g. a new machine should be inserted into an already working production line, or an assembly line has a lower performance than excepted, etc.). It is important that every stakeholder (policy maker, analyst) involved in the planning process understands and agrees with the statements. The problem formulation might change during the study when some of the results are available.

After declaring the problem, the *objectives* of the study and an *overall project plan* should be set. For the aforementioned assembly line, an objective might be to find an answer to the question of how to insert the new machine into the production line so that the performance increases. Or to figure out how the performance of an assembly line that is not performing as expected and cannot keep up with the plan could be increased. At this point in time it should be decided whether the simulation is the appropriate method to reach the stated objectives or whether other methods such as analytical methods, should be applied. Next, the alternative systems should be considered and the overall project plan should be set down. It should include all aspects such as how many people will work on the project, costs, deadlines and expected results (Banks 1998).

When the problem and the objectives of the study are identified and the application of simulation techniques have been selected to analyze the problem, a *simulation model* can be built. For that, first the system, including its components and their interconnections, the system boundaries and the system environment should be stated. This identifies, for example, which part of an assembly line should be part of the system to be analyzed and which part only interacts with the system and serves as input data for it. To create an appropriate and efficient simulation model for a system is a complex and difficult task. Since the fields of application, systems to analyze and purposes of the study differ from case to case, it is not possible to give a set of instructions that will always lead to an appropriate and effective model. However, some general guidelines might be followed (Banks 1998). As Banks and Carson describe: “The art of modeling is enhanced by an ability to abstract the essential features of a problem, to select and modify basic assumptions that characterize the system, and then to enrich and elaborate the model until a useful approximation results” (Banks and Carson 2009). Thus, it is recommended to start with a simple model and extend it toward a higher complexity. When the necessary complexity of the model has been reached depends upon the purpose of the study.

Parallel to the construction of the simulation model a *data collection* that defines the necessary input and output data for the model should be set up. Since the complexity of the model might change with time, the necessary data for the simulation should also be revised, and if necessary, changed. This results in a constant interplay between the construction of the model and the definition of the necessary data collection.

Once all of the necessary data is in place, the *implementation* or *modeling* of the simulation model can be started. For that either a simulation language or a special purpose simulation software is

[8] Based on the description of Banks and Carson (2009) and Page and Kreutzer (2005)

necessary. Simulation languages provide a powerful environment for simulations for the high qualified user, who has experience in programming. Some well-known simulation languages are GPSS, Arena, Simio, AnyLogic, Modelica, Matlab and Simulink, etc. These languages provide a great flexibility for users to set up and improve their models, but it is also very time consuming. Special purpose simulation software, such as Simphony (AbouRizk and Hajjar 1998) or Stroboscope (Martinez and Ioannou 1994) used by the construction industry provide an environment within which to model specific construction problems, like tunnel or bridge construction. These software systems offer the user predefined model component templates that can be used to establish a simulation model. These software systems do not require knowledge in programming and therefore can only be used for the selected specific purpose. However, they frequently give the user the opportunity to extend the predefined templates or create new templates and components by using the underlying programming language.

When the simulation model is complete, it must be *verified*. The verification process should prove that the implemented simulation model is an accurate reflection of the designed model (Page and Kreutzer 2005) and that the computer program performs properly (Banks and Carson 2009). For that, the input parameters, the logical structure of the model and the results should be reviewed. For a complex model this is difficult, if not impossible. Therefore, the modeling strategy introduced above, of starting with a simple model and enhancing it further when it performs well, also simplifies the verification process of the model. In this case, only the new components and their logical structure shall be verified, since the existing model is already verified.

After the verification, the simulation model must also be *validated*. Validation compares the simulation model to the real system (Kamat and Martinez 2003). This is often an iterative process that takes until the simulation results adequately represents the examined real system. For the assembly line, for example, the simulation results must match the performance of the real system in order to be validated.

When the simulation model performs correctly (verified) and represents the real system accurately (validated) the experimental phase can be started. Depending on the purpose of the study, first, the *experimental design* must be stated. Diverging simulation scenarios (alternatives) should be defined that correlate to the initial purpose of the simulation. Alternatives might differ in model layout (different sequence of machines), components (further machines), input data (the arrival of 10% more material to work on), attributes (performance factor, capacity), etc. During the experimental design, different decisions have to be made depending on the kind of simulation, such as the length of the initialization period, the duration of individual simulation runs and the number of replications for each run.

During the *analysis* the different scenarios are executed to estimate measures of performance for the ongoing simulation scenario (Banks and Carson 2009). To draw conclusions from the analysis results the user himself must detect patterns that are related to the purpose of the simulation (Page and Kreutzer 2005). Typical examples are that the performance of machine 3 is much lower than the performance of the other machines, or by switching the sequence of machine 1 and 2 the performance of the factory increases by 2% in average.

When the analysis is completed the analyst decides based on the results of the simulations whether *additional simulation runs* are necessary or whether the available ones provide enough information on the behavior of the system and such that the initial question can be answered.

Since simulations generate a huge amount of data, the documentation and clear representation of this data is a very important task (Page and Kreutzer 2005). Good representation and documentation is important for the analyst to find the above-mentioned interesting patterns and for external people to understand the working mechanism and characteristics of the examined model. For such purposes

graphical representations such as diagrams, bar charts and UML diagrams are usually used. Since different scenarios were simulated, it is important to document not only the results, but the model itself as well, so that a correlation between the model and the results can be deduced.

The final step of the simulation modeling process is the practical application of the results of the simulation. According to the simulation results, decisions can be made that may change the production layout or may add additional machines to the production line in order to reach a higher production rate, or to avoid bottlenecks within the production line. If the simulation model that was created was used properly such that every necessary component and influence factor has been included into the model, the results will reflect the real behavior of the analyzed system and can also adequately predict changes in its behavior.

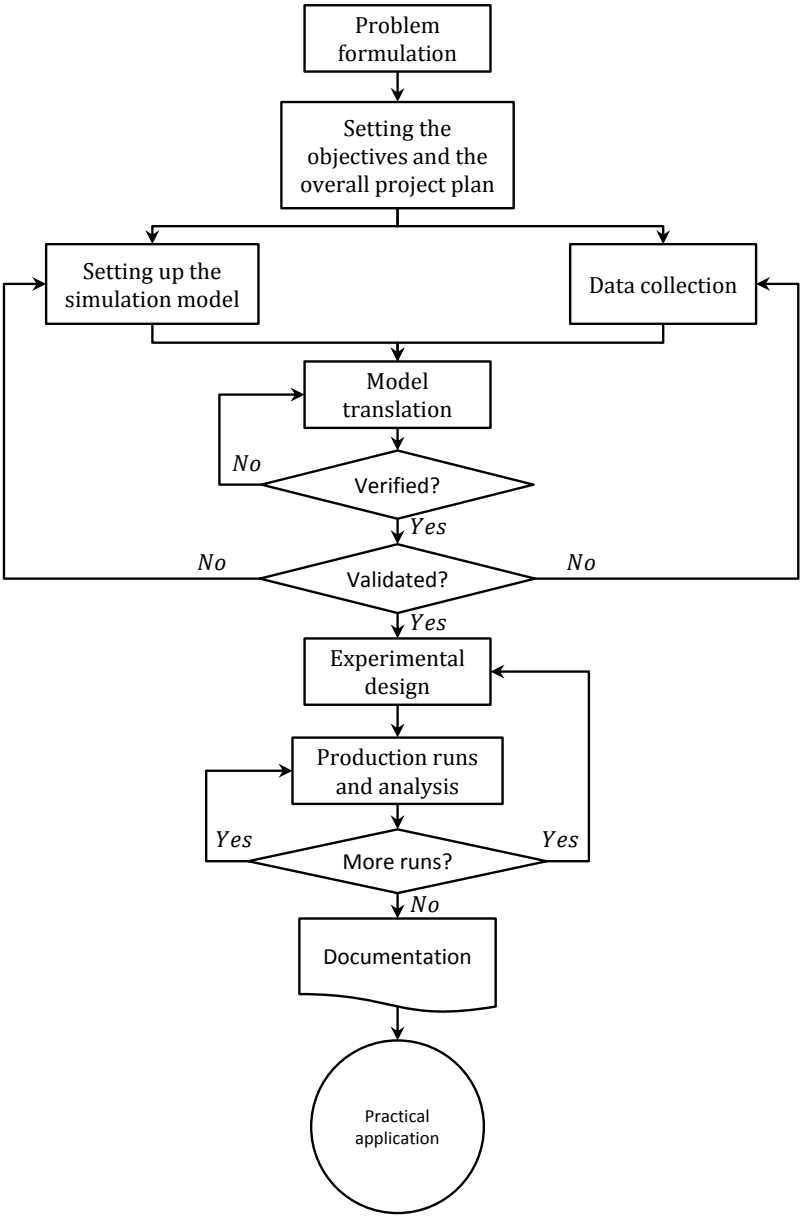


Figure 3-6: Steps of a simulation study (Banks 1998)

3.2.4 Advantages and limitations of the simulation technique

Simulation is a promising technique for a user who tries to study problems of the real world, because simulations are capable of predicting how a system that is still in the planning phase would perform, or enhancing a system that is already in place. Additional reasons as to why the simulation is a favored technique in the manufacturing industry have already been addressed in Section 1.4.1. In this section further advantages and disadvantages of this technique will be discussed to get a realistic overview about this technique.

One important characteristic of the simulation is its capability to model complex systems. Through simplifications and abstractions it can accurately model systems that cannot be analyzed by any other tool (Page and Kreutzer 2005). System modeling requires a high understanding of the system itself, its components and their interactions. Therefore, the understanding of the systems behavior improves, just through the modeling process, even before starting a simulation. Simulation is the perfect tool to experiment with the system. Parametric studies, what-if experiments, alternative scenario tests and bottleneck analyses can be executed before the real system has been built. The behavior of a system can be studied using simulations without impacting the real system. Once a model has been set up, it can be used as part of a bigger model, thus reducing modeling time. Furthermore, in dynamic simulations, the time can be accelerated or slowed down, so that important processes can be analyzed in slow motion, or long-term predictions can be made by simulating years of production in only seconds of simulation time.

Beside these important positive characteristics, as with every other technique, simulation also has some limitations and disadvantages over other methods. The list of these limitations according to Page and Kreutzer (2005) are the following:

- Conceptual gap between model and reality
- Confusing a model with its reality
- Lack of transparency
- Inadequate data
- Propensity of errors
- High degrees of construction effort
- Misplaced confidence in computer generated data

Since a model is always a simplification and abstraction of the real system, the gap between model and system might become too large. Thus, the results of the simulation will no longer represent the accurate behavior of the system. Such a gap can occur due to misinterpreted relationships and working mechanisms, overseen (not modelled) components or inadequate goals (Page and Kreutzer 2005). The more complex the model, the higher the risk of a modeling error. Furthermore, one model represents only one of the many aspects, of how a system can be described or handled. Therefore, the significance of one simulation run should not be overestimated, and the conclusions should also involve different simulation models. In this context the transparency of a model is an important factor that can be lost e.g. by overly simplified assumptions, overcomplicated interactions or hidden parameters.

One further category of risk is related to the input data of the simulation. Empirically inadequately supported estimates as input data may devaluate the results of the simulation. When the necessary adequate data is not available or cannot be obtained, an approach other than applying simulation techniques should be considered to solve the problem (Page and Kreutzer 2005). Furthermore, when the complexity of the model increases there is a higher risk of internal modeling errors that are hard, or in some cases impossible, to detect. To keep the risk as low as possible, the modeling shall be

started with a simple model and enhanced toward the desired complex scenario. Proper validation also guards the models from internal errors and is therefore an important part of modeling. Due to these difficulties that the analyzer has to face while modeling, the construction phase of a model is time-consuming and the analyzer requires increased abilities and skills to avoid the aforementioned mistakes and errors. To support the analyzers and accelerate the construction process of models, vendors of simulation software develop verified packages, also called templates that can be used to build up complex simulation models. Additionally, these packages are set up with output analysis capabilities to verify not only the working mechanism of the model, but also the feasibility of the results. Furthermore, due to the rapid development of computer hardware and specific simulation algorithms, the duration of a simulation is getting shorter and shorter (Banks and Carson 2009). Due to this technical advancement and the aforementioned advantages of the simulation, the application of this technique for scheduling purposes in the construction industry is useful. For such purposes the *discrete event simulation* is the most suitable one, which will be introduced in the next section.

3.3 The discrete event simulation

The discrete event simulation is a type of simulation that models systems for which changes in the system's state, known as events, occur only at discrete points in times (Banks and Carson 2009). The simulation time of the model jumps forward in time between these events and the model´s state between these discrete time steps stays constant. Discrete event simulation has a wide range of application in many fields including engineering, health, management, telecommunication and transportation sciences (Fishman 2001, Banks and Carson 2009). Since it is capable of handling complex systems and providing information about its behavior, both researchers and industry apply this simulation technique to support their decision making processes. The next section will introduce the basic concept and different components of the discrete event simulation and explain how this technique works. After that, different simulation model designs will be introduced and compared to each other.

3.3.1 Discrete event simulation model components and simulation concept

The discrete event simulation, like other simulations, consists of *entities* that model the components of a real system and may interact with each other. Entities are represented by their *states* and methods or rules called *events*. Events are responsible for changing the entities' states over time (Page and Kreutzer 2005). These events are associated with a *time property* that is responsible for triggering the state transformation at a specific point in time.

Since the entities have different states it is important to illustrate how they change over time. For simulations two interpretations of time can be distinguished. The first one is the *modeling time*, which is a fictitious time that passes internally during the simulation and is independent from the duration of the computation. The second one is the *real time*, which passes during the execution of the simulation (Page and Kreutzer 2005). For example, a production simulation of a factory might take several seconds in real time, but during this time several years of modeling time might have passed. To model such a time a *simulation clock* is used (Banks 1998).

To execute a simulation run, first, the modeling time is set to zero and the state of the model is set to the initial one. The simulation clock jumps to the first event and changes the state of the

corresponding entity according to the rules of the event. Many events, e.g. those that are responsible for starting an activity, might introduce further events into the simulation. Those events will be registered on the time line by the *simulation controller* for future consideration. When an event is executed, the modeling time jumps to the time step of the next event that occurs and triggers that event. The selected events that are planned to occur in the future are stored by the simulation controller in the *event list*. The event list is often represented by a queue where the events are ordered according to their appearance in time.

Events that have the same event time are called *parallel events* and can be processed in any order unless further dependencies define it differently. The processing order of these parallel events can occur according to different queuing rules (Banks 1998, Banks and Carson 2009). In the *first come first served* rule (FCFS) the event that is placed into the queue first will be executed first. In contrast, with the rule of *first come last served*, the event that is placed into the queue first must wait until every other parallel event that is put into the queue is executed and the first event is executed last. A schematic representation of the two different queuing rules is represented in Figure 3-7. During simulation several operations might be executed with the event list such as event insertion or deletion, reordering or accessing events (Page and Kreutzer 2005).

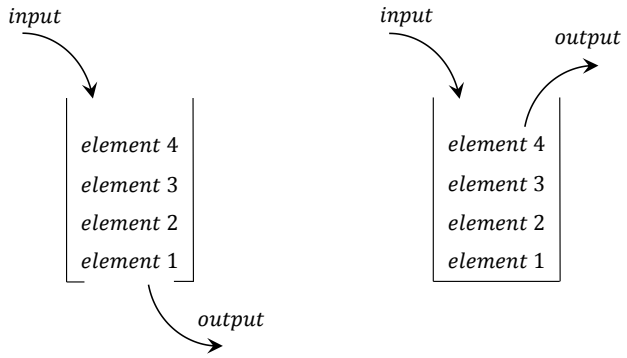


Figure 3-7: The first come first served (left) and first come last served (right) strategy for queuing systems.

To advance time during a discrete event simulation there are two options available: the *event* based and the *fixed time increment*-based time advancement (Banks and Carson 2009). The advantage of using the *event-based modeling time increment* over *fixed time increments* is that it has the more precise representation of state transformations and it avoids the idle times caused by the unfortunate choice of fixed update time step sizes (Figure 3-8).

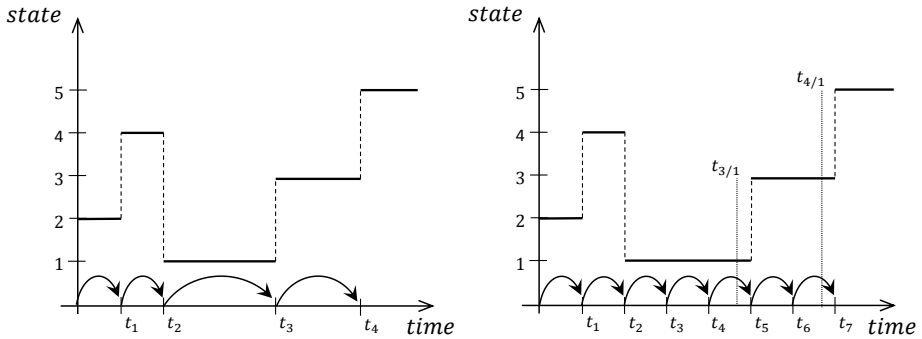


Figure 3-8: Comparison of the event-based (left) and the fix time increments-based (right) simulation clock advancement techniques. Due to the wrong time step size selection the last two events of the simulation must be delayed. ($t_{3/1}$ and $t_{4/1}$ represent the time steps t_3 and t_4 in the first diagram on the left)

The simulation runs as long as there is an event remaining in the event list, or until some other predefined criteria (such as time interval, number of products produced, etc.) terminates the simulation.

3.3.2 Discrete event modeling styles

To synchronize the simulation time passage with the state changes of the model, three different approaches can be used. Page and Kreuzer (2005) describe how to use *events*, *activities* and *processes* for such purposes.

Events describe changes in the model's state at specific points in time. They can be generated either externally or internally depending on what the causes of the event is. Externally generated events are solely determined by the environment and can describe, for example, the arrival of transportations to the construction site. Internal events are caused by state transformations of further entities. For example, when a product has been produced, an event "*load it to the transporter"* might be generated.

The second choice to synchronize time passage with state changes are the activities. They describe operations with specific durations between state changes. Therefore, state changes (events) always occur at the beginning and at the end of an activity. For example, a transporter arrives at the factory and has the state of *"transporter loaded"*. The arrival triggers an activity "*unload transporter"* which starts with the event *"start unload"* and sets the activity state to "*unload in progress"* and ends with the event *"unload complete"*, which leads to the next activity state *"transporter unloaded"*.

As the third approach, processes can be used. They describe a sequence of activities that build the lifecycle of the entity. The lifecycle of an entity might be straightforward, e.g. for the transport of material first it must be transported to the construction site, then unloaded, then used to build the structure and maybe, some decades later, it gets demolished. The lifecycle of an entity can also be cyclic. For example, an excavator might be associated with the activities of "load transporter with mud" and "wait for the next truck to arrive". These activities are repeated in a cycle as long as necessary until all of the mud has been transported.

The relationship between the events, activities and processes is represented in Figure 3-9. Based on these three synchronization forms, different techniques can be applied to model time passage between the state changes. The conceptual details and working mechanism of these three aforementioned techniques will be introduced in the next sections of this chapter. Further, less commonly used techniques such as resource-based simulation or transaction[9]-based modeling styles will be mentioned in passing.

[9] The components of a queuing scenario are servers and model requests. Servers are referred to as resources and model requests as transactions.

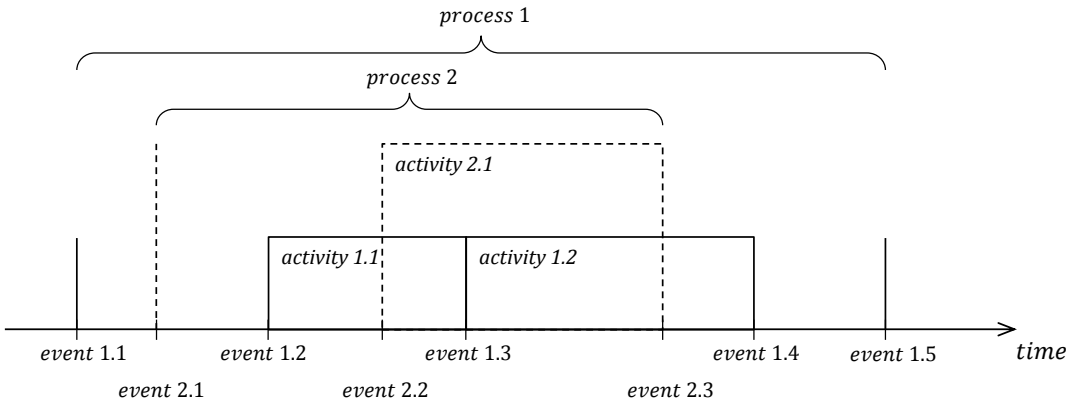


Figure 3-9: Relationship between events (vertical lines), activities (boxes) and processes (curly brackets) represented with an example of two processes. A process might be the complete construction process of a wall. Activities could be assembling the reinforcement and formwork of the wall or pouring concrete. Events might represent the start of assembling the reinforcements, or finishing the pouring concrete activity.

3.3.2.1 Process-oriented simulation modeling

One of the most commonly used modeling techniques of discrete event simulations in the manufacturing industry is the process-oriented approach (Page and Kreutzer 2005), also called process interaction (PI) strategy (Martinez and Ioannou 1999, Mohamed and AbouRizk 2005). For this strategy the simulation model is created from the entities' point of view. Therefore, every entity owns one process that is also called the *lifecycle* of the entity. The lifecycle contains every important procedure that the entity can go through. It includes a sequence of every activity in the order of their occurrence and every relationship to the other entities in the model. Simulation time passes whenever an event triggers a delay in the time instance (e.g. activity duration: one day). The lifecycles of the simulation model entities are executed in parallel (Figure 3-10). The control of execution is supervised by the simulation's *time management executive*. At every point in time it traverses through the processes one after another while passing control to a process to make the next possible steps in the entities' lifecycle. When the process cannot make any further steps in the entity's lifecycle, the control will be given back to the executive, which than gives the control to the next process and so on until the simulation terminates. A process can modify the properties of an entity, generate new entities and lifecycles or activate or deactivate other entities at specific clock times. Furthermore, processes can deactivate themselves and pass control to the simulations time management executive or other entities in the model. Additionally they can terminate themselves and the lifecycle of other entities (Page and Kreutzer 2005).

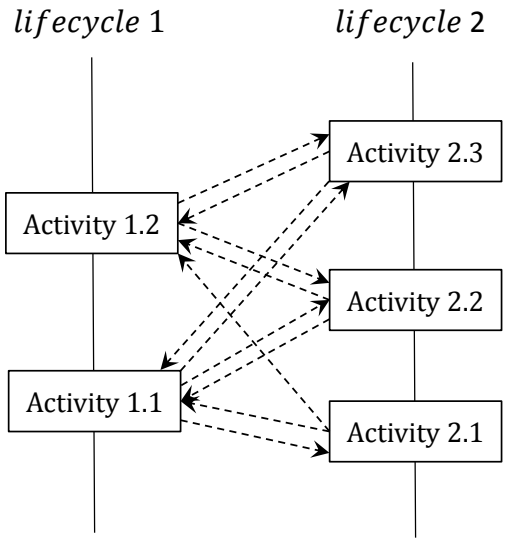


Figure 3-10: Parallel execution of two lifecycles (processes). The arrows represent interaction between the activities of the processes.

The most important component of the process-oriented simulation is the simulation executive. It is responsible for executing the state changes at the right time and in the right order. To facilitate this requirement it can assign two states for every process: *active* or *passive*. An active process is allowed to execute its lifecycle until it gives up the control or inactivates itself. The passive processes are not allowed to execute their lifecycles until they get reactivated. Activation is performed by the executive or other processes. The passive state can last either for an indefinite time or for a predefined time period. By inactivating a process, its current state will be stored and until the process is reactivated at which time the process will continue from where it was stopped. The work of the executive costs real time but no modeling time.

During the execution pseudo simultaneous processes may occur. These are processes that execute a state transition at the same model point in time. Such simultaneous transitions are not allowed in a simulation because they might change the state of the same entity, which would lead to a logical error since the simulation would not know which one is the current state of the entity. To avoid such errors, the pseudo simultaneous processes must be serialized. The activation order of these processes can have a significant impact on the results, therefore a reliable control of the serialization is important (Page and Kreutzer 2005). Reliability can be ensured by queuing systems.

Queuing systems are mainly used for scenarios with resources and transactions that require that resources be processed in a specific order. Since resources have a limited capacity, when this limit is reached further transactions cannot be processed until the resource once again has work capacity. When a transaction cannot be processed it must wait and therefore it will be placed in the queue. Which transaction will be selected when the resource is available again depends on the strategy that is applied for the scenario. The most commonly used ones are the *first come, first served* strategy (see Section 3.3.1), the *highest priority first* strategy or the *shortest completion time first* strategy. The result of the simulation may vary depending on the strategy selected. Therefore, queuing systems are well suited for optimization purposes such as the optimal strategy for allocating transactions to the resources (Page and Kreutzer 2005). Such queuing systems can also be applied for process-oriented simulations to serialize pseudo parallel processes.

3.3.2.2 Event-oriented simulation modeling

Event-oriented simulation is older than the process-oriented simulation and also goes by the name of event scheduling (ES) strategy (Page and Kreutzer 2005). Instead of grouping every activity to one process like in the process-oriented modeling, this technique clusters every event based on the time of their occurrence. One cluster represents simultaneous events that occur at the same model time but are most likely causally unrelated to each other. An example for a cluster could be when a loaded truck arrives to the construction site, an available excavator must be activated and one event must set its state to "in work", while with another event the truck must be set to the state "unload in progress". These events are correlated with two different entities but due to the interaction between them, they can be clustered into one big event.

Due to these differences in the concepts of the simulations, the process for building the model is slightly different for the two simulations. To create a model for a process oriented simulation, the modeler concentrates primarily on one entity at a time, and collects every possible activity and interaction the entity has. During the same process for an event-oriented simulation, the designer must concentrate not just on one entity, but rather must address all the entities. The designer must collect every interaction, transformation and event for all of the entities that can happen at the same point in time. This is neither a more difficult nor an easier process creating a model for the process oriented simulation. In the end both simulations will have the same functionalities within the model. The only difference is that they are formulated from different points of view.

For event-based simulations, similar to process-based simulations, the simulations executive plays an important role during the simulation. The executive processes the list of events that should be executed. The executive then selects the event with the earliest occurrence time. When there is more than one event at the same time to start, a queuing system can be used to order the execution. Every event condition, except for time conditions, must be implemented within the event routine (Hooper 1986). In an event-based simulation the time passes while the executive jumps from one event to the next event, which is located at a later point in time.

3.3.2.3 Activity-oriented simulation modeling

In contrast to the process interaction and event scheduling strategies, models for the activity oriented modeling (also called activity scanning (AS) strategy) are built up from the activities point of view. The model consists of activities that are waiting to be executed. To execute an activity all of its prerequisites, such as resources or logical dependencies, must be fulfilled. Therefore, at every step during the simulation, the simulation executive checks the prerequisites of the activities and starts all the executable activities. When there is no further activity to start, it jumps to the next recorded clock time. When an activity is completed, it releases its resources, which can then be allocated to other activities that have not yet been started. The activity scanning strategy is best suited for systems with highly-dependent components that need many conditions to allow activation (Hooper 1986), such as construction projects. The activity-oriented simulations have been developed especially for the construction industry. These use the modified form of the AS strategy such as the *three-phase* or the *three-stage strategy*. Both of these strategies are used to control the simulation experimentation.

The three-phase strategy combines the AS with the event scheduling strategy and classifies the start and end events of the activities into two types of events. The first class comprises the B events that are bound events. B events are predictable and thus they are schedulable (Lin and Lee 1993). The second class, C events, consist of conditional events. Their start times depend on a fulfillment of certain conditions (Zhang et al. 2005). The first phase of the simulation advances the simulation time.

The second phase initiates the B events and phase three checks the conditions of the C events and initiates the executable ones.

In contrast to the three-phase strategy, the three-stage strategy does not classify the activities. At stage one the simulation scans and checks every activity for their start conditions. Stage two advances the simulation time to the next activity start time and stage three finishes the affected activities and releases their resources. This strategy was developed by Shi (1999) for the activity-based construction simulator that will be introduced in more detail in Section 3.4.

A representation form of an activity oriented model is the Activity Cycle Diagram (ACD), which can represent both the idle (circle) and active (rectangle) states of the simulation entities (Martinez and Ioannou 1999, Zhang et al. 2005). Within the ACD the alternating circles and rectangles are connected with directed arcs that represent the flow of resources (Figure 3-11). ACDs serve as blueprints for the development of AS simulation models (Martinez and Ioannou 1999). Shi (1999) introduces a simplified representation of the ACD, based on the three-stage strategy. He recommends modeling the two states of an activity as an attribute of the activity and to represent the two states only with one element. This is also advantageous for the modeler, since it is easier to model states at the local environment of the entity rather than independent from the activity through further elements (Shi 1999).

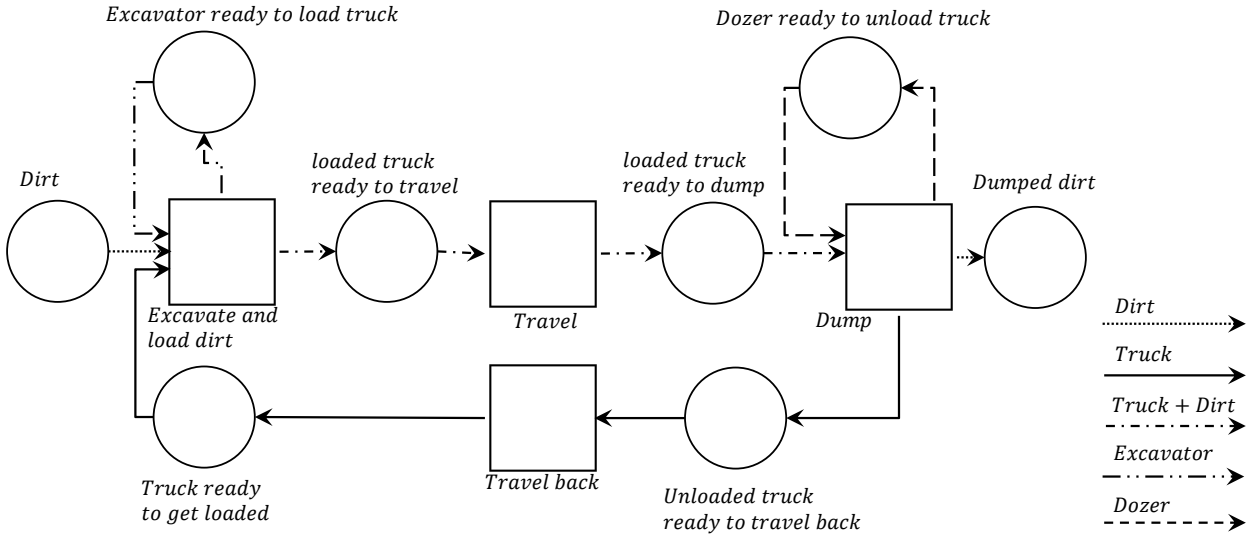


Figure 3-11: Earth moving process represented with an ACD

3.3.2.4 Comparison of simulation strategies

All three introduced simulation strategies view the real system from a different point of view. Therefore, numerous researchers have tried to determine which one of the above mentioned strategies is actually the best for simulation (Hooper 1986, Martinez and Ioannou 1999, Zhang et al. 2005). The researchers have all pointed out that there is no best strategy, however some are better suited to work with one special kind of system than the other ones. It is also evident that the selected strategy will significantly affect the modeling of the system. Therefore, the modeler should decide which strategy best suits the system needs and if he can implement the system in the most effective way. For their

support, researchers like Hooper (1986) collected the characteristics of the three strategies and provided advice for their best applications.

Event scheduling strategy is very flexible in its application, but its model development process requires significant effort. This strategy is efficient for the execution of systems with relatively independent components, such as straight manufacturing lines, but it can become inefficient for models with relatively dependent components (Hooper 1986), such as the construction industry.

In contrast, the activity scanning strategy is efficient for models with highly dependent components, where the components require the fulfillment of many conditions for their start. During the simulation, a considerable amount of work is required by the executive that verifies the conditions of the activities and determines their execution order. Due to these high computational efforts, the activity scanning strategy might become inefficient for models with relatively independent components (Hooper 1986).

For many modelers, an attractive characteristic of the process interaction strategy (Banks 1998) is its close representation of the model to the problem and its straightforward model development and modification process. The logic behind a process is similar to the life cycle of a real system component. Furthermore, the process-oriented modeling collects both properties and behavior of an entity into one process that simplifies the modeling logic by narrowing the gap between the real system and the designed model (Page and Kreutzer 2005). During modeling, the modeler follows the sequence of activities that appear in the lifecycle of a component and when some activity changes, the process can be easily updated to reflect the change. However, this modeling advantage might cause difficulties during the simulation.

The process interaction strategy (PI) -based simulation is the strategy that requires the most support from the executive. The PI-based simulation might have inefficient execution times. Due to the inactivation and reactivation of processes during the simulation, the implementation of the simulation itself with a programming tool that does not support coroutines[10] becomes difficult. Programming languages, like JAVA, that does not support coroutines directly often use thread mechanisms to model the competing processes (Page and Kreutzer 2005). Since threads were originally designed for real time parallel computing purposes and not for the modeling of process oriented simulations, the computational time of such models might increase beyond the simulation time of an event or activity oriented simulation.

According to Hooper (1986) and Martinez and Ioannau (1999) the PI strategy is best suited for systems where the moving entities have many attributes and the machines and resources that serve them have only a few attributes, along with a limited number of states and relatively few interactions. Such systems can be found in the manufacturing industry. On the other hand, Zhang, Tam and Li (2005) identify activity sequencing as the best suited strategy for the simulation of construction processes, where the activities (construction tasks) have many dependencies (resources and technological dependencies). Based on the newer programming techniques, they propose a new form of the AS strategy, the activity object-oriented strategy that will be introduced in Section 3.5.5

A conclusion of the comparison is that the nature of the system could have an influence on the choice of strategy. Therefore modelers should first investigate the nature of the system before choosing a strategy to implement the simulation. It has been also identified that there is a correlation between the “ease of model development and incorporating future changes” (Hooper 1986).

[10] Coroutines are computer program components that enable the suspension and reactivation of the execution at certain locations.

3.3.2.5 General-purpose and special-purpose simulation

Simulation frameworks can be categorized into two classes such as *general-purpose* and *special-purpose simulations*. The general-purpose simulation can be used to model almost any kind of real world system. Its most important feature is its flexibility and comprehensiveness that allows it to suit many modeling requirements (Hajjar and AbouRizk 1996). Through its flexible components and their programmability, the simulation model can be precisely tailored to reach any desired complexity. However, due to this freedom of programmability significant effort must be invested to create even a simple simulation model (Chua and Li 2002). Therefore, general-purpose simulations are useful only for designers and researchers with advanced programming skills and large simulation experience.

To suit the needs of modelers who do not have such advanced programming skills or prior experience with simulations, special-purpose simulations have been developed. This second category of simulation targets to model only a specific domain of the real world such as tunneling, manufacturing, ship building, etc. Such simulations provide the modeler with a large set of predefined model components that are typical for the particular domain. The modeler uses these components to set up the simulation model (AbouRizk and Hajjar 1998). Due to the presence of predefined components, special-purpose simulations are not as flexible as general-purpose simulations. Therefore, to increase the flexibility of the special-purpose simulations, several frameworks allow the user to not only use the predefined set of components, but also to modify and extend them with new elements (Chua and Li 2002).

In the construction industry there are numerous applications of both general and special purpose simulation frameworks. Those will be introduced in the next section.

3.4 Simulation-based scheduling in the manufacturing industry

As it has been introduced in the last sections, simulation methods are powerful tools to solve complex problems such as scheduling under resource constraints. While this technique is already widely used in the manufacturing industry, it is hardly known in the construction industry. The next section will address why the simulation approach is not as popular in the construction industry as in the manufacturing industry and compare its advantages and disadvantages to a static method such as the disjunctive graph model (Section 2.3.2). In Section 3.4.2 the discussion which began in Section 2.4.1 will be continued and further reasons will be introduced as to why the adaption of the simulation technique to the construction industry is a complex process. In the end of the section a simulation technique will be discussed. This simulation technique can efficiently be used in the construction industry for scheduling purposes.

3.4.1 Generating schedules for the shop problems by simulation

An excellent approach to plan or schedule feasible solutions for a scheduling problem in the manufacturing industry is to use computer-based simulation techniques. The first research results of simulation approaches go back to the late 1950's (Day and Hottenstein 1970). The fundamental recognition by Nelson (1958), Jackson (1957) and Evans (1964) to consider the shop problem as a *network of waiting-lines or queues* is still one of the foundations of the modern simulation methods.

Dynamic simulation methods have been applied in the manufacturing industry for system design, operations planning and scheduling for more than 40 years (Smith 2003). Simulation models are capable of modeling complex system operating rules and with the inclusion of a coupled visualization, to visually evaluate the results (Bell and O´Keefe 1986). The model can be updated quickly in situations where there is a sudden change in the employee availability, machine performance, etc. and a new schedule can be generated in a short time (Concannon et al. 2003). Therefore, one advantage of the simulation is the ease of rescheduling the project when conditions change (Drake et al. 1995). Multiple "what-if" analysis and parametric studies can be carried out with the simulation to create efficient schedules.

The advantage of simulation over static methods, such as the disjunctive graph model (Section 2.3.2), becomes significant when the project is made up of multiple operations and machines, and their interactions are numerous and complex. Assessing the applicability of a schedule and identifying inefficient schedules in advance in a short computational time, can only be achieved by applying simulation techniques. A further advantage of simulations is that they are capable of considering complex boundary conditions such as probabilistic distributions for operation durations.

The applicability of schedules generated by static methods can be determined either by setting up a simulation model, or by the actual execution of the schedule in real time in the shop. The former leads to a doubled modeling of the same problem, but allows the prediction and identification of possible bottlenecks in advance, which can save time and costs during the execution. In the latter case, bottlenecks and conflicts appear real-time without prediction, leading to reduced performance and higher costs.

In the manufacturing industry one simulation model represents one possible schedule for the scheduling problem. The schedule is composed of the determined production durations for a product. The schedule can be described by the factor productivity, which indicates how many products can be produced in a predefined period of time with the corresponding schedule, e.g. product/hour or product/day. By changing the order or the performance factors of the machines, a new schedule can be generated that can be described with a different productivity factor. Therefore it is important to stress that one simulation model represents only one feasible solution for the identified scheduling problem. In order to determine an optimal solution, the simulation needs an external algorithm capable of changing the simulation model that can steer the simulation.

3.4.2 The adaption of simulation techniques into the construction industry

The adaption of solution methods from one industry to another is not only encumbered by the differences of the methods itself (as introduced in section 2.4.1) but also by the differences in the characteristics of the industries. While processes of the manufacturing industry are executed mostly in a hall, which is generally not affected by any weather conditions, the processes of the construction industry take place in the open air, where rain and strong wind can have a significant impact on the performance factor of the workers and machines: In the worst case weather can even result in the need to stop the construction until the necessary conditions are in place again.

Additionally, the production lines in the manufacturing industry have a static layout, which means that the products are moving along a conveyor belt and are prepared with machines that are stationary. This structure also represents the layout of the simulation model, where the products are moving from machine to machine. In contrast, the product of the construction industry, i.e., the building, is stationary and the machines and other resources are mobile. Thus the layout of the construction site is dynamically changing with time.

In addition, in the manufacturing industry one operation can only be executed by one specific machine. In contrast, in the construction industry one construction task might be executed with different resources without impacting the schedule, hence an explicit connection does not exists between resource and task. To define every possible resource-task connection, dynamically changing connections are necessary. Hence the layout of the simulation model for the construction industry is not as straightforward as for the manufacturing industry.

Based on these theories Beißert et al. (2007b) developed a new constraint-based simulation method that has also been used by the author for scheduling purposes and will be introduced in the next Chapter in detail.

3.5 Existing simulation-based scheduling approaches in the construction industry

Similar to manufacturing engineers, construction engineers face the problem of developing and efficiently designing productive construction schedules. However, while manufacturing production plans contain many repetitive processes that are used to create the identical product many times, one right after the other, the construction of a building is a unique process and the result is always only one "product" (as described in Chapter 1). Even when constructing the same building at another place there will be differences in the boundary conditions, such as ground material, logistics, the surrounding neighborhood, weather conditions, etc. and therefore each building will be unique. AbouRizk, Halpin and Lutz (1992) name this uniqueness and the lack of repetitive processes as possible reasons why the study of work processes in the construction industry did not get much attention until the 1960's. At that time the researchers realized that although construction projects are unique, they contain repetitive processes such as earth transport, tunneling, road construction, etc. and a closer investigation of these processes begun (AbouRizk and Hajjar 1998). With the emergence of computers, new computer-based techniques such as simulations appeared to solve problems related to scheduling in the construction industry. According to AbouRizk et al. (2011), Teicholz was the first who applied simulation methods to study the complexity of earth hauling systems. Based on the work of Teicholz with the "link-node" methodology, Gaarslev compared the results of queuing theory (see Section 3.3.2) and simulations with the use of a simple construction system (AbouRizk et al. 2011)

Like in the manufacturing industry the schedule of the project is determined according to the results of a simulation run. During a simulation run the execution dates of an activity (start event and end event equals start date and termination date of the activity) will be saved and merged with other activities to create a schedule. The schedule can then be visualized for example with a Gantt chart (see Section 2.2.3).

3.5.1 CYCLONE

The first implemented general-purpose simulation framework that has been applied to simulate construction operations was introduced by Halpin at the beginning of the 1970's (Halpin 1977). CYCLONE is an acronym for CYClic Operations Network and is a logical extension of the network-based scheduling methods (see Section 2.2.5) to provide a better understanding of resource interactions on the process level (AbouRizk et al. 2011). The newest versions of CYCLONE (from

the 1990's to the present) use an activity sequencing strategy-based discrete event simulation. Therefore, to develop a model one should focus on the involved resource and their interactions (AbouRizk et al. 1992). Resources are modeled as entities of the simulation and can have two states: active or idle. The former one is represented in the simulation environment by a rectangle and the latter one by a circle. Active and idle states of a resource are connected to each other with arcs in the simulation model. Events move the resource along the arcs and change their state according to their position (e.g. state 1: truck loaded (idle) → event 1: truck loaded for trip to dump zone → state 2: truck traveling (active) → event 2: loaded truck arrives at dump→ state: truck waiting to dump (idle) etc. see Figure 3-13). Thus a resource moves along the arcs and executes one activity after another. CYCLONE uses six different elements to build up a model: the normal element, the combi element, the queue, the function, the counter and the link or arc. The formal representation of these elements in CYCLONE is depicted in Figure 3-12.

Normal elements are activities that have no constraints, such as travelling, dumping or spreading earth material. When a unit arrives to this element the process begins immediately without delay. In contrast, combi elements require more than one type of resource for their execution. The process only starts when all of the required resources have arrived to the element. Queue elements represent waiting areas for resources. A queue element is always followed by a combi element and a resource in the queue element must wait until the following combi element is ready to process it and picks it out of the list. Function elements are used to create new resource units for the simulation. Counter elements are used for statistical purposes to track the passage of time or amount of unit that have passed the counter. Passing a counter element for a unit does not cost any model time. The nodes are connected by links or arcs. Arcs represent the flow direction of the entities between the connected elements.

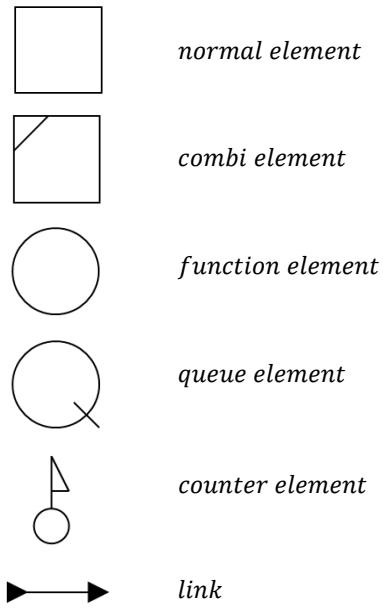


Figure 3-12: Modeling elements of CYCLONE

Since CYCLONE has been developed for cyclic processes in the construction industry, the application of this simulation will be introduced using a simple earth transportation example. For earth transportation the necessary resources include excavators, trucks, dozers and of course the earth material that is to be excavated, transported and dumped. The simulation model is represented in Figure 3-13. It starts with a queue that defines the initial amount of dirt to excavate. In the next queue,

the initial number of excavators can be defined. This is also the queue where the idle excavators will return after they have loaded a truck. Similar to the excavators, another queue will be defined for the dozers at the other end of the model. The only missing resources are the trucks that will transport the excavated dirt to the deposit site. For that another queue is defined (Trucks waiting to get loaded) where the available amount is initialized and they can wait to be served by the excavator. For the loading activity a combi element is used (Excavate and Load dirt) that has three required components: an available excavator, available dirt and an empty truck. When this activity is complete the truck with the dirt is sent through an event to the next normal element: Travel. After the predefined length of time the truck will arrive to the queue "Trucks waiting to dump". Meanwhile the excavator jumps back to the excavator queue and is ready to serve another truck.

For the dumping activity, another combi element is defined that requires a waiting loaded truck and a dozer to start. When they are both available, the dumping starts and the truck and the dozer will be released again after the process is complete (or after a predefined amount of time). When the process is complete, the truck goes back to the queue for trucks waiting to get loaded through the normal element: Travel back. Simultaneously the dozer goes back to the queue for the dozers and is ready to unload another loaded truck.

There have been three counters introduced to the model. One to measure the amount of dirt that has been loaded onto the trucks, the second measures how much dirt has been unloaded and the third measures the cycle time, or production rate, of the trucks.

After completing the simulation model, a simulation of this simple scenario can be achieved. For example, having 20 units of dirt to excavate, with two excavators that can excavate and load one unit of dirt to a truck in 10 minutes, with a traveling time of 30 minutes to the dump site where one dozer is waiting to unload the trucks in 5 minutes of time, using five trucks that can transport one unit of dirt, all the dirt can be transferred to the dump site in 290 minutes. If ten trucks are available, the same scenario can be finished in 165 minutes. Further parametric studies could be performed by varying the performance of the excavators and the dozers or by changing the available number of them.

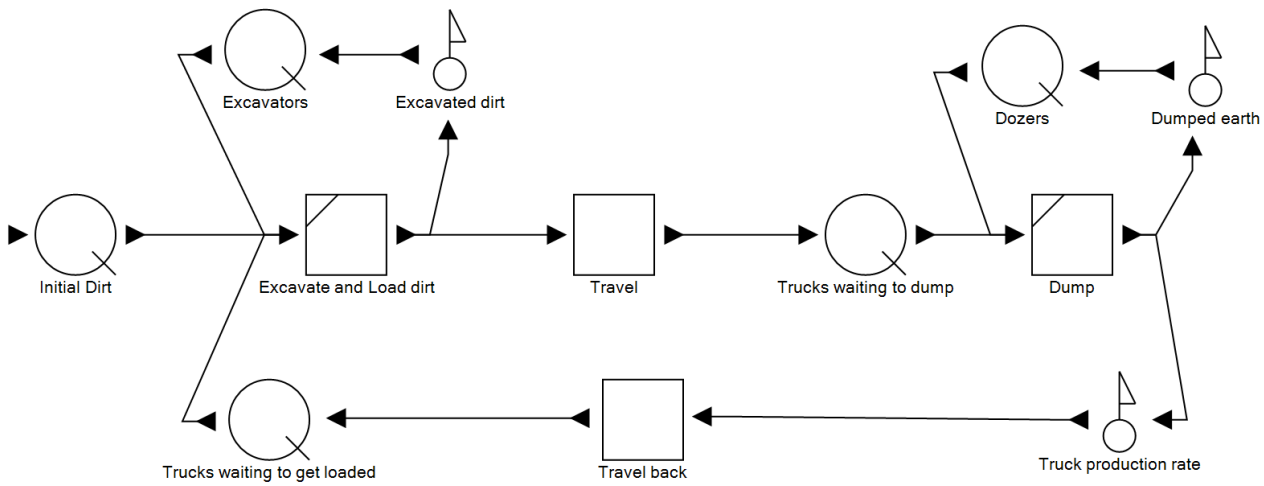


Figure 3-13: Simple cyclic earthwork simulation model with CYCLONE.

3.5.2 Simulation frameworks inspired by the success of CYCLONE

Due to the success of CYCLONE, at least three additional standalone implementations have been developed introducing new features and capabilities to the simulation (Martinez and Ioannou 1999). INSIGHT (Kalk and Douglas 1980) adapts the simulation to microcomputers and uses predefined graphical elements to construct simulation models at a faster pace. UM-CYCLONE, developed by Carr and Ioannou (Ioannou 1989), is the third implementation of CYCLONE. Ioannou provides a comparison between the performance of UM-CYCLONE and INSIGHT using two examples. In both examples UM-CYCLONE turned out to be more efficient. The fourth implementation was created by Halpin (1990) for microcomputers, thus the name microCYCLONE. The simulator has been enhanced by process cycle and stage buffer monitoring and a statistics collection mechanics to collect information between processes by Lutz (1990).

CYCLONE uses the ACD representation of the system (see Section 3.3.2.3) not as a blueprint for the model but as the model itself. This representation form brings both advantages and limitations for the modeling. While using the ADS as a simulation model makes the modeling process simpler and easy to learn, but complex logics and interconnections that cannot be modelled by CYCLONE must be simplified (Martinez and Ioannou 1999). After the comparison of microCYCLONE and SLAM II, another well-established general-purpose simulation language, Gonzalez-Quevedo, AbouRizk, Iseley and Halpin (1993) concluded that the microCYCLONE system, in general, works well for the construction problems it was developed to address. Major drawbacks include the inability to recognize resource attributes, allow user-defined functions and to combine simulations.

Chang (1989) tried to overcome the drawback with the resource attributes with RESQUE (RESource based QUEuing network simulation system) by including an overlay that introduces flexibility for the simulation by resource distinction and increased simulation control (Martinez and Ioannou 1999). However, RESQUE still had its limitations in resource representation and resource assembling and dissembling (Chua and Li 2002).

Most network-based simulation languages represent the real systems by graphical models which require a separate input file to describe the system to be simulated. Liu introduced a new simulation system, COOPS (Construction Object-Oriented Process Simulation System), based on the concepts of CYCLONE enhanced with an object-oriented simulation design (Liu and Ioannou 1992). The integration of interactive computer graphics facilitates building simulation models within the simulation framework by graphical elements without the need of an input data. Due to the object-oriented representation, every entity of the simulation becomes traceable and so the generation of simulation statistics such as which resource visited which activity at what time also become available. By introducing resource calendars, the working time of machines and labor can be restricted to predefined periods in time, e.g. laborers work only on weekdays. Liu and Ioannou introduce a new node: the router to the simulation that represents a binary random selection of possible ways (Liu and Ioannou 1992).

CIPROS (Tommelein and Odeh 1994) also has an object oriented system that extends the resource representation of RESQUE by allowing multiple properties for the resources. Its hierarchically ordered expandable knowledge base of resources (e.g. labor, machines, equipment, material, etc.) and construction tasks accelerates the building process of simulation models.

The former simulation frameworks were all general-purpose simulation frameworks in the construction industry. In contrast, STEPS (STructured Environment for Process Simulation) (McCahill and Bernold 1993) is a special-purpose simulation for planning and controlling basic earthwork projects. It is based on ACDs and was developed for the U.S. Navy Civil Engineering

Laboratory. It supports the rule-based release of resources and different resource sizes in the same queue, but lacks a graphical interface (Chua and Li 2002).

3.5.3 STROBOSCOPE

STROBOSCOPRE is an acronym for State and ResOurce Based Simulation of Construction ProcEsses (Martinez and Ioannou 1994). It is a powerful general-purpose simulation framework, designed for modeling complex construction operations and for the development of special-purpose simulation tools (Martinez and Ioannou 1999). It supports the three phase AS strategy and the ACDs (Martinez and Ioannou 1999). In contrast to CYCLONE, the ACDs of STROBOSCOPE only represent the simulation model at the conceptual level. The detailed model is defined by a series of programming statements. Therefore, model specific details can only be found in the source code of the model. The elements of STOBOSCOPE are the supersets of those in CYCLONE, represented by five nodes and four special type of links (Martinez and Ioannou 1999). The Combi node is represented in STROBOSCOPE ADS by a rectangle cut-off in the top left corner. The further nodes correspond to the ones in CYCLONE (Figure 3-14). Beside the basic elements of CYCLONE (normal, combi, function and queues) the fifth node is the enhancement of the router node introduced in COOPS and it is called the fork node (smaller circle with an inscribed triangle).

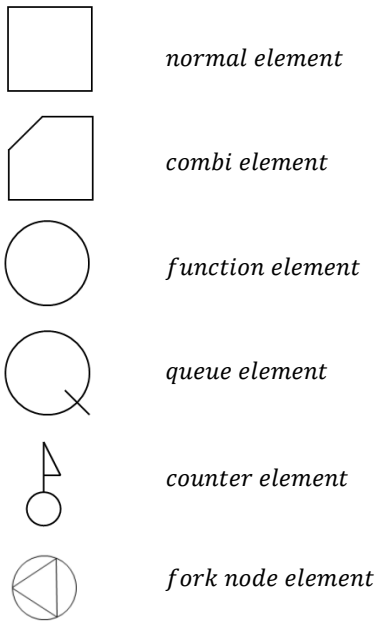


Figure 3-14: Nodal modeling elements of STROBOSCOPE

Links, such as Normal link, SameLink, Outlet and Inlet, are modelled as objects and so they also have attributes like strength and DrawAmount (Chua and Li 2002). The links within the model are named after the resource that flows through them followed by a number as identification. The simple earthmoving model that was introduced in Section 3.5.1, is represented with the STROBOSCOPE ACD in Figure 3-15. The model has been extended with a fork node that allows the possibility for trucks to select an alternative route for their return.

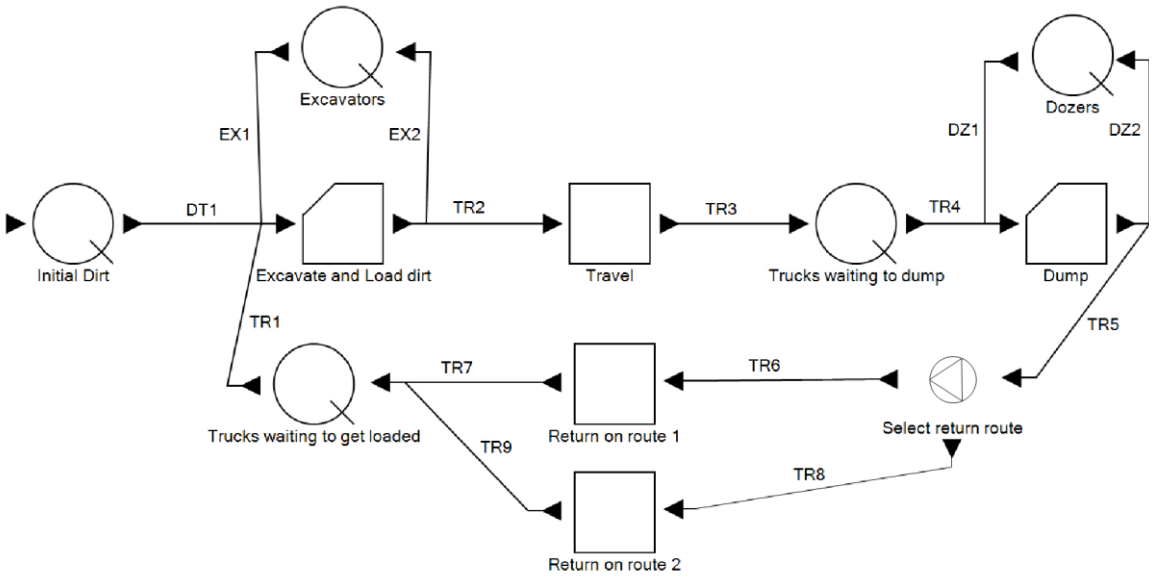


Figure 3-15: STROBOSCOPE ACD for the simple earthmoving system that was introduced in Figure 3-13 extended by a fork element allowing the trucks to select a return route. The meaning of link names: DT stands for dirt, EX for excavator, DZ for dozer and TR for truck.

Every element in the ADS is connected with a method that describes its behavior. This method can be modified by rewriting or extending the code to reach the desired behavior of the node. The power of STROBOSCOPE has been proven by many researchers in many fields of construction industry, such as lean construction (Tommelein 1998), earthwork operations (Martinez et al. 1994), tunneling operations, quarry operations and concrete block manufacturing (references and further examples in Martinez and Ioannou 1999).

In the 1990's, many frameworks were available for the users to simulate construction operations. However, they were not widely known within the industry. This may be explained by the complexity of the construction operations themselves, the complexity of the simulations and primarily by the serious effort required to build up a simulation model. Therefore, researchers concentrated on developing simulation approaches that were more attractive for the industry. To accomplish this goal research has primarily been aimed at making the modeling process easier, less abstract and instead of making additional general-purpose simulations, making more special-purpose simulation (SPS) packages that completely model specific fields of the construction industry (earth work, tunneling, bridge construction, etc.). A goal of the research was also to lower the complexity of the simulations themselves, so that the program is easier for the user to manage.

As a result of such research a simplified version of STROBOSCOBE, the EZStrobe, has been developed for users with less simulation experience providing the same power as the original program (Martinez 2001).

3.5.4 Simphony

The three main standalone SPS tools used in the construction industry in the 1990's were AP2-Earth (Hajjar and AbouRizk 1996), CRUISER (Hajjar and AbouRizk 1998) and CSD (Hajjar et al. 1998). The first tool was designed for large earth moving processes, the second tool to model aggregate production plants, and the third to optimize construction site dewatering operations (Hajjar and AbouRizk 1999). Based on the experiences with these three tools, Hajjar and AbouRizk (1999) collected a broad set of requirements that are necessary to adapt a simulation framework in the construction industry.

Simphony was introduced by Hajjar and AbouRizk (1999). It was designed to fulfill the following requirements:

- A user-interface that supports the graphical representation and manipulation of the simulation model.
- The possibility for advanced users to bypass the graphical system and enhance their models by code.
- Hidden abstract underlying constructs for non-expert users.
- Support of reusability of models and project tools.
- Support of combination of models.

Simphony is a discrete event simulation environment both for users with few simulation experiences and for developers. Developers can use Simphony to create new SPS templates[11] that can be used by the normal users of the software.

SPS templates can be created by the Simphony Designer, which provides the necessary programming environment for designers to implement their new elements. Once a new element is complete it will be stored in the Modeling Element Library and can be used for simulations.

Normal users only have access to build models and execute simulations using the Simphony Editor. Simphony Editor provides users with a graphical interface containing multiple windows where different views of the model can be represented. For example, one window shows the layout of the model, another shows the project navigation tree, a third shows the available modeling elements, and the fourth shows a tracing window for the results. The created simulation models with their results are stored in the Construction Simulation Project Database. The User Model Library is a database similar to that found in prior simulation models in that it can be reused or combined with further models. The user interface of Simphony is depicted on Figure 3-16.

[11] Collection of modeling components targeted for a single domain (Hajjar and AbouRizk 1999)

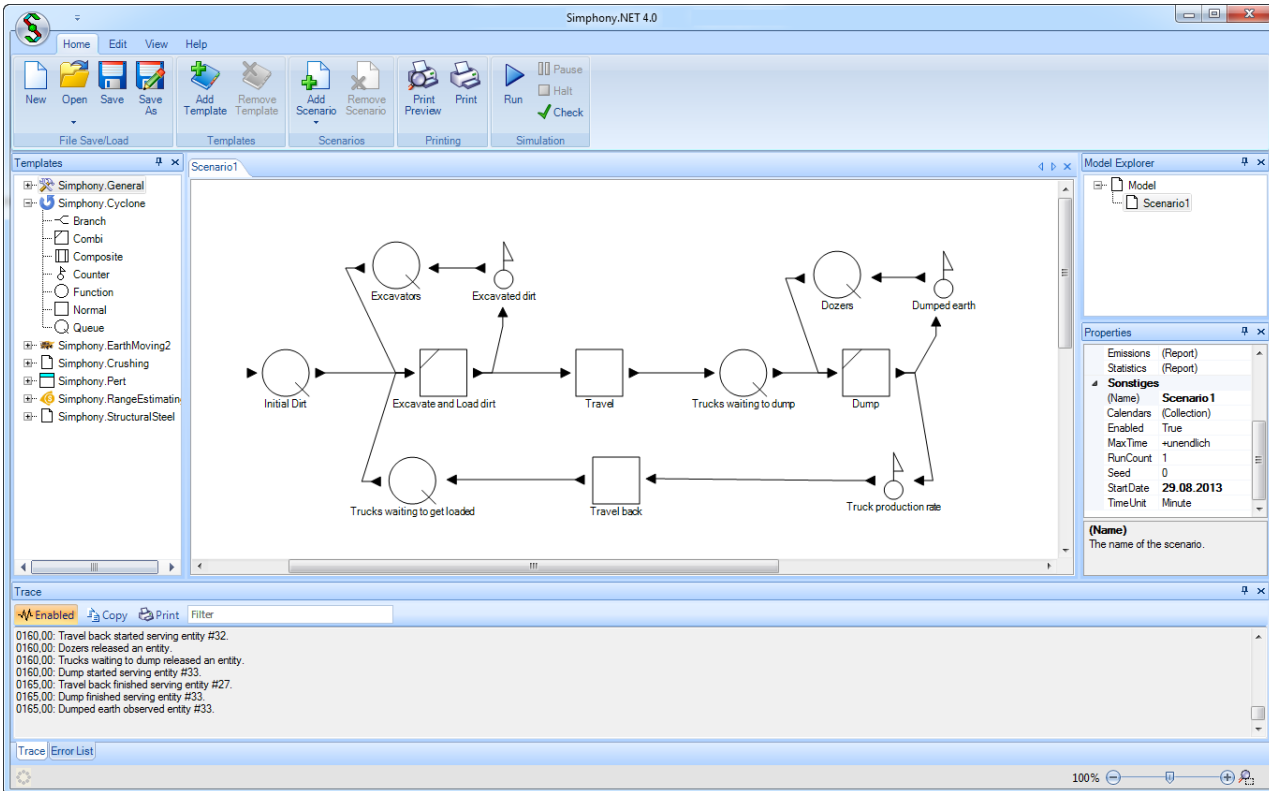


Figure 3-16: The user interface of Simphony.NET 4.0

Simphony has been adapted to the .NET framework by AbouRizk and Mohamed (2002) and the new simulation engine Simphony.NET has been introduced. Further developments evolved with Simphony will be presented in Section 3.5.8.

3.5.5 Activity-based construction and activity object-oriented simulation strategy

The Critical path method (CPM) is a well-known and simple scheduling method that is widely used in the construction industry (Section 2.2.5.4). To narrow the gap between the academic and industrial use of construction simulations Shi (1999) raised the question: “Can simulation be made as simple as CPM without sacrificing its functionality?”. To answer this question, he developed a new simulation method called the activity-based construction (ABC). Since CPM uses only activities as planning foundation, the ABC technique simplifies the ACD representation of the model by introducing a single activity standing for both the “idle” and “active” state of the activity. Thus, the new model representation contains only activities. The state of an activity is treated as a new attribute which can be either active or idle. The necessary resources and further conditions for the activities can also be stored as attributes. The ABC model representation of the simple earth moving system that was introduced in Figure 3-13 is presented in Figure 3-17.

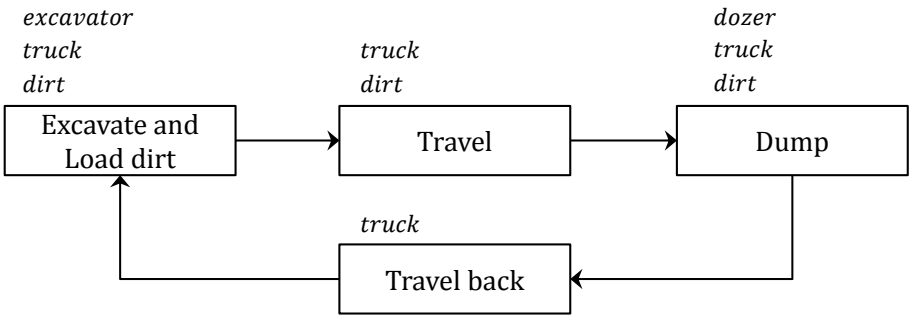


Figure 3-17: ABC representation of the simple earth moving system that has been introduced in Figure 3-13

Based on the new modeling representation, Shi (1999) also introduced a new simulation strategy: the three-stage strategy. As discussed in Section 3.3.2.4, the first step of the three-stage strategy is to select the activity to start with by checking all the constraints it has (resources or logical constraints). After there are no more activities to start at the current point in time it moves to the second step whereby it advances the simulation to the next marked point in time. As step three, when an activity terminates, it releases the entities that were necessary for the execution of the activity, such as resources or reserved working area.

Zhang, Tam and Li (2005) criticize the three-stage method because when there are a large number of activities and the activities have many dependencies and constraints, the simulation speed may slow down. Furthermore, allocation of entities (e.g. resources) occur only at the beginning of the activity and cannot be changed during execution. Even when further resources will be available during the execution they cannot be picked up by the activity. Therefore allocation policies must always be predefined for the three-stage strategy.

Rahm et al. (2012) introduced an approach that is able to suspend activities in a discrete event simulation in the case of malfunction of the resource or disturbance and then continue the activities where they had been suspended after the resource is operational and the disturbance has gone. A similar interruption of activities when the finish event of another related task is triggered provides the possibility for the ABC method to allocate further resources to the suspended tasks.

To support such dynamic decisions during simulation in allocating limited resources for multiple competing activities they developed a new strategy (activity object-oriented strategy), similar to the activity sequencing, enhanced with a further step. In this extra simulation step when an activity terminates it releases its resources and, based on different rules, these resources can then be picked up by activities already in active state (Zhang et al. 2005). To speed up the simulation, the constraint check of activities is only executed when a resource is released and not at every time step. To gain additional speed, only the activities that might need the released resources are checked instead of the program checking all of them.

3.5.6 Further simplifications of the discrete event simulation approach

To further simplify and speed up the model development cycle, Chua and Li (2002) proposed a new simulation approach: the Resource-Interacted Simulation (RISim). By focusing on the resource and process level of the system, users without much knowledge about simulation can easily generate new simulation models (AbouRizk and Hague 2009).

To extend the previous work of Shi (1999) and Hajjar and AbouRizk (1999), Lu (2003) developed the Simplified Discrete Event Simulation Approach (SDESA). This new approach simplifies the

activity sequencing strategy by removing the test head for checking the event conditions from the "begin service" and merges the "arrival" event into the "begin event" (Lu 2003). The new executive is represented in Figure 3-18.

The SDESA distinguishes between disposable and reusable resources. In the case of a disposable resource, when an activity is complete it will not be released but rather remains unavailable for further activities. Disposable resources include reinforcement cages and specific materials. Reusable resources will be identified as available after the completion of an activity and can be picked up by other activities. Reusable resources include labor work forces and machines. All resources are stored within one dynamic resource entity queue during simulation.

The SDESA reduces the queuing structure into a single dynamic queue of flow entities. Flow entities are passing through a sequence of activities in a process, and are interacting with resources at each activity for a certain duration (Lu 2003). Flow activities only contain a time cell to track arrival, waiting and departure times at activities.

Hence, the complete simulation process is simplified to the interaction of two dynamic queues: the flow entity queue and the resource entity queue.

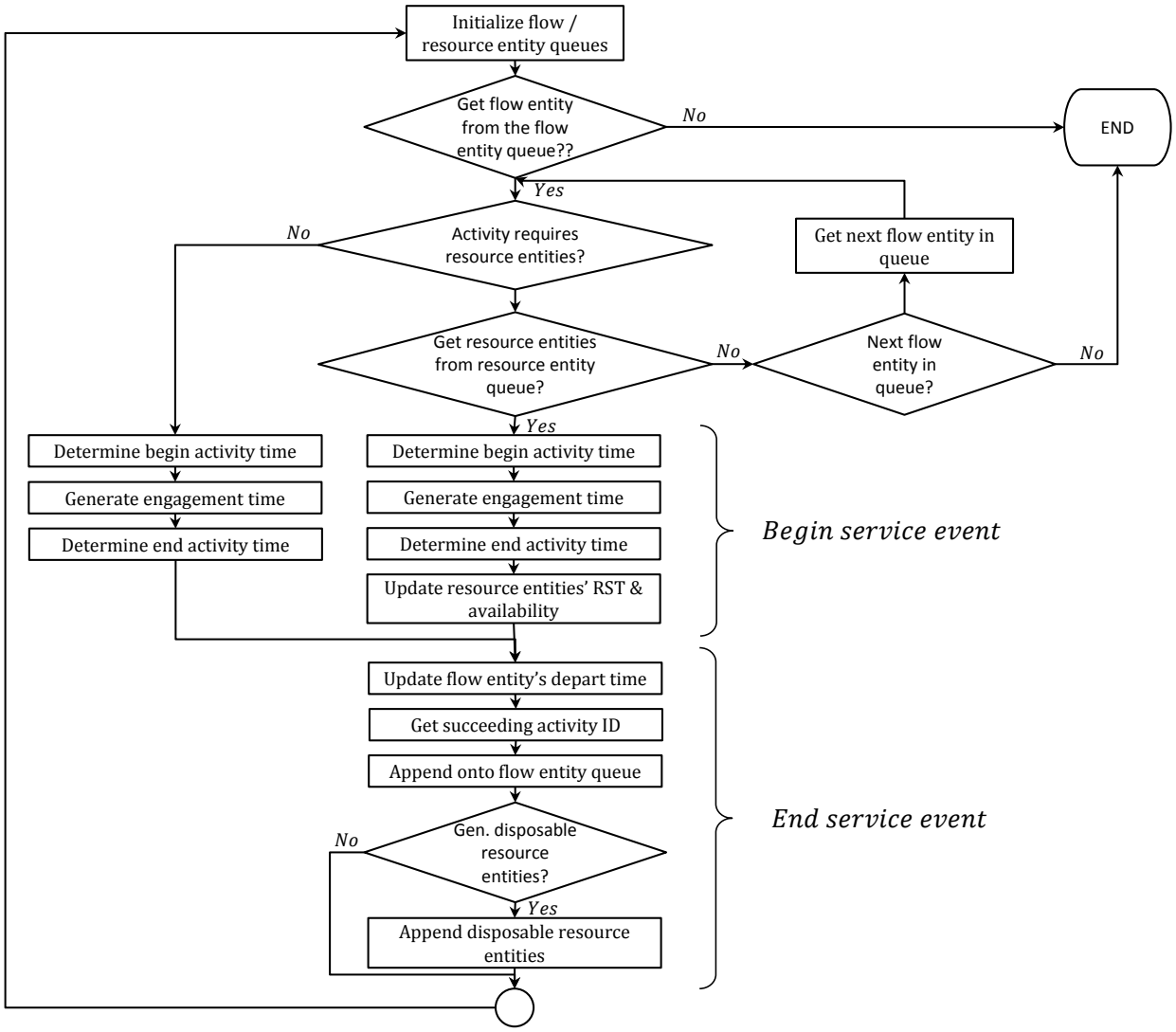


Figure 3-18: Flowchart of the SDESA executive (source: Lu 2003)

3.5.7 Integration of other tools into the simulation framework

With the increased popularity of Computer Aided Design (CAD) software in the construction industry such as AutoCAD or MicroStation, toward the end of the 20th century researches realized that integrating such a tool into the simulation framework could be beneficial. Using visualization software to aid the simulation model declaration, information such as physical product attributes, geometric data and volumetric data can be transferred automatically to the simulation. Furthermore, site layout and additional geometric data of the construction project are mostly stored in CAD drawings. These are very important inputs to create an accurate simulation model and can be imported from CAD drawings.

To provide a more effective definition of project and simulation models Xu and AbouRizk (1999) integrated AutoCAD models into Simphony. The CAD model of the system is embedded within the product hierarchy of the simulation model. It is used to feed the simulation with information about geometrical and volumetric data and characteristics of the site.

3D visualization tools cannot only be used to provide input data for the simulation but also to visualize the results. By integrating a time element to the visualization, a 4D representation of a schedule can be established. Zhang (2000) introduced the 4D Graphics for Construction Planning and Site Utilization (acronym 4D-GCPSU) platform for planning construction projects. This framework not only facilitates the planning of building construction but also the resource management and the site space utilization. It has been enhanced by Wang et al. (2004), who extend the framework by site layout assessment, dynamic resource management and cost control so that project managers can update their resource plans according to the changes in their schedule.

Dawood et al. (2003) developed an integrated information resource base for 4D/Virtual Reality construction processes simulation that is composed of a database of building components (as CAD packages from AutoCAD), Project Management package and graphical user interfaces. After the simulation the schedule is presented in a 4D AutoCAD environment.

Kamat and Martinez (2003) introduced VitaScope, a discrete event simulation system based on STROBOSCOPE integrated with a 4D visualization package. The visualization of construction schedules is helpful to identify logical, temporal and spatial errors within the schedule. Therefore a 4D visualization of schedules can also be used to validate the schedule (Kamat and Martinez 2003).

Chahrour (2006) uses AutoCAD to determine the input data for the simulation of earth moving processes. His prototypical simulation framework uses product-modeling concepts as data structure and is capable of integrating simulation concepts with CAD representations of the construction site.

Tulke and Hanff (2007) introduced a new Building Information Model (BIM) based 4D visualization approach instead of the CAD model-based approach. Since a BIM approach contains not only the 3D geometry of the connected building but also further information as to elements such as material, construction method and geometrical data (Eastman et al. 2011), it can be used to determine the duration of tasks automatically by selecting quantities from the model. In the CAD-based approach such a determination has to be done manually. Therefore, the process of setting up a simulation model can be achieved faster using the BIM-based approach than with the CAD model-based approach.

Since the gaming industry is one of the leaders in 3D visualization, ElNimr and Mohamed (2011) utilize the game engine Blender to visualize the results of a simulation run.

Another game engine-based 4D visualization was introduced by the author of this thesis (Dori and Borrmann 2011). The introduced aim was to visualize not only the building phases of the individual building components, but also to create 4D movements of the resources such as labor and machines to detect both logical and spatial collisions on the construction site. To achieve this goal, animation

snippets were introduced, each one intended to represent a change of one geometrical parameter or attribute (e.g. rotation, movement, change transparency). Animation snippets are the basic elements of an animation chain, which describes the complete lifecycle of movements of a visualized entity. The animation snippets and animation chains are created automatically based on the results of the simulation. However, every single path of possible movement must be drawn within the visualization framework (called Animator, which is based on the Irrlicht game engine) and connected to an animation snippet manually before the start of the visualization. This is a time consuming and inefficient process that should also be automated in order to create a visualization package that is useful for the industry.

3.5.8 Distributed simulations, high level architecture and CoSye

The long-term goal of construction simulations is to achieve a fully integrated and highly automated environment that can be used through the complete lifecycle of the building. To achieve this goal, the simulation approaches used in the construction industry must be re-engineered on a foundational level (AbouRizk and Hague 2009). A suitable foundation for such an environment can be established e.g. by distributed simulations, specifically by using *High Level Architecture* (HLA) approaches.

HLA is a standard of the IEEE (IEEE 1516 - 2010) developed originally by the Defense Modeling and Simulation Office (DMSO) of the Department of Defense (DoD) with aircraft and weapon system development purposes. It provides rules to connect simulations with each other to create an even larger simulation. Such is reached through the real-time communication and data-sharing between the *federates* (e.g. a simulation, a database, visualization or further software components). The communication between federates is governed by the *Run Time Interface* (RTI).

The communication between federates and the RTI is established through *ambassadors*. Direct communication between federates is not allowed, so information exchange always must flow through the RTI. The collaboration of two or more federates describe a *federation*. The *Federation Object Model* (FOM) describes the universe of the simulation. It contains every necessary object and data that is involved with the simulation. The federates share all objects and data with each other. The basic functionality of the HLA is depicted in Figure 3-19.

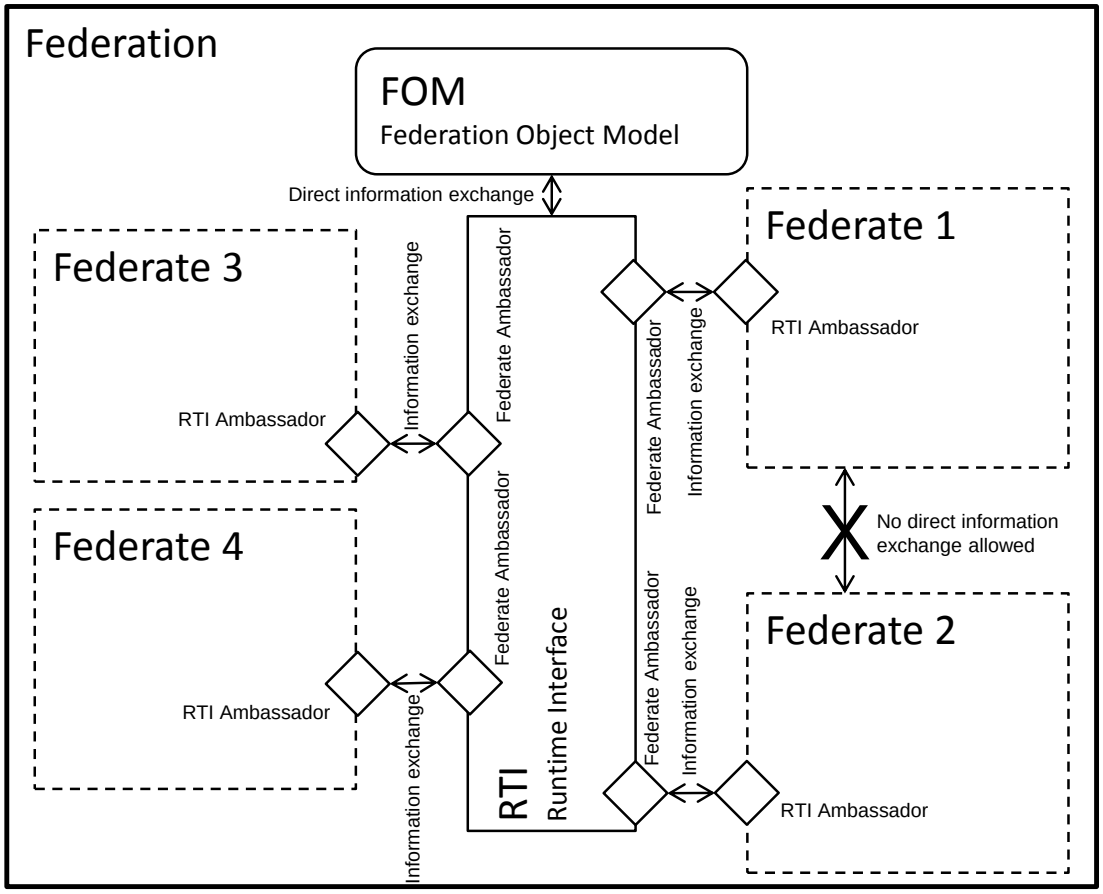


Figure 3-19: Components of the High Level Architecture: The federation, the RTI, the FOM, federates and Ambassadors

AbouRizk and Hague (2009) introduced a new HLA-based simulation environment called *COnstruction SYnthetic Environment* (COSYE) that supports the parallel and distributed simulation of construction operations and development of large scale construction simulations. By using HLA and COSYE, the designer can decompose the originally complex construction system into smaller and simpler components and define them as federates. Federates can be developed independently and connected later with COSYE. Federates can be distinguished as *main federates,* such as the site manager, the process model, fabrication shop, etc., or *supportive federates,* e.g. resource allocation, calendar or visualization (AbouRizk et al. 2010). The application of COSYE in collaboration with industrial partners has been presented by AbouRizk et al. (2010) for industrial construction and Xie et al. (2011) for tunnel construction.

Hollermann et al. (2012) developed an HLA-based distributed communication system for bridge construction. The federates in this case represent the actors of the construction operations, such as the design engineer, site manager or the client. The introduced FOM define classes based on the Industrial Foundation Classes (IFC - ISO 16739:2013) to enhance the communication between the different involved parties and create one common shared model.

3.5.9 Petri Nets

In addition to discrete event simulation, *Petri Nets* (Petri 1962, Petri 1966) are also widely used for general simulation applications. Petri Nets provide a mathematical formulation and graphical representation of the studied discrete system. A Petri Net is a directed, weighted, bipartite graph that consists of four basic types of modeling elements (Sawhney 1997): *places*, *transitions*, *links* and *tokens* (Figure 3-20).

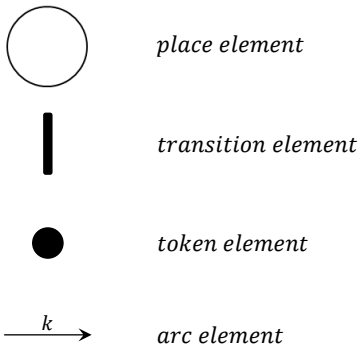


Figure 3-20: Modeling elements of a Petri Net

Places (Circles) represent states of the system. Transitions (squares) are activities that cause a change in the system's state. Links represent connections between places and transitions, and describe the flow direction of the tokens. Tokens are resources that are necessary to trigger or fire a transition (Wakefield and Sears 1997).

The simulation model is made up of alternating place and transition elements connected with links. Initial tokens can be positioned into places. A transition can fire when all necessary places before it are occupied by a token. When a transition is fired it consumes all the tokens in front of it and generates new tokens to put into the follower places. A transition might hold up the tokens for a time period that supports the time passage of the model (Jensen 1991).

Such a description of a network is practically identical with the functionality and appearance of the Activity Cycle Diagram (Section 3.3.2.3 - Figure 3-11) (Lin and Lee 1993). A place is analogous to a queue, or a transition, the tokens to the resources and the holding period of a transition to the duration of an activity (Martinez and Ioannou 1999). Therefore, Petri Nets are often used to study systems formulated as the three-phase AS strategy (see Section 3.3.2.3) and show high potential to simulate construction operations (Lin and Lee 1993). The Petri Net representation of the simple earth moving process that has been introduced in Figure 3-13 at the initial state is depicted in Figure 3-21 and an intermediate state after 15 entities of dirt have been transported is depicted in Figure 3-22.

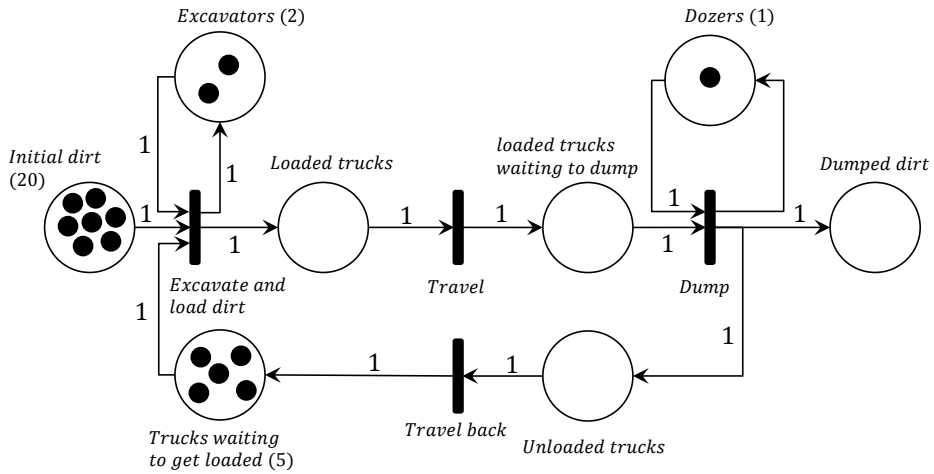


Figure 3-21: Petri Net model of the in Figure 3-13 introduced simple earth moving project at the initial state

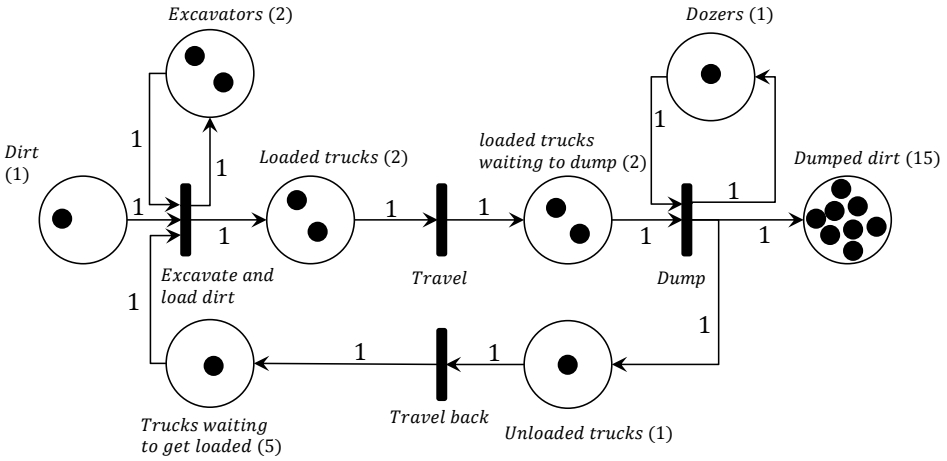


Figure 3-22: Petri Net model of the in Figure 3-13 introduced simple earth moving project in the state after transporting 15 entities of dirt to the dump area.

Sawhney (1997) draws attention to three important requirements which are necessary to allow efficient scheduling of construction operations with Petri Nets. The three requirements are the following: "First, the ability to provide hierarchical breakdown of a complex construction project. Second, the incorporation of risk and uncertainty in the activity time and cost estimates and the network logic. Third, the modeling of dynamic resource allocation and utilization on a construction project".

The main reasons why the application of Petri Nets are favored to simulate construction processes are collected by König (2004) and Berkhahn et al. (2005). These are the following: the graphical representation of the system, the bipartite structure containing places and transitions to model states and activities, the token concept for modeling logical conditions and dependencies and "the mathematical formalism for structural, behavioral and simulation analysis of engineering process models". However, this technique also has limitations, including the inability to represent directly bulk materials. For instance, soil or aggregate that should be transferred from one place to another cannot be modelled directly with Petri Nets (Martinez and Ioannou 1999).

Since the emergence of Petri Nets, the method has been developed and enhanced with further functionalities such as the aforementioned time association with transitions. These are called the Time

Petri Nets (TPN). A time delay for generating new tokens can be either deterministic or represented by any stochastic distribution function. Further information about the TPN can be found in Wakefield and Sears (1997). Another extension is the Colored Petri Nets (CPNs) that allows the association of different colors to tokens, thereby creating distinctions between different resources. Such distinctions can also be used for transitions that need e.g. one yellow and two blue resources to fire (Jensen 1996).

Beside the aforementioned theoretical development of Petri Nets for construction operations, the technique has also been applied to investigate the behavior of real systems such as earth moving operations (Wakefield and Sears 1997, Cheng et al. 2011), concrete placement operations (Wakefield and Sears 1997), bridge construction scheduling (Sawhney and Vamadevan 2000), geotechnical planning process (Berkhahn et al. 2005) and building design process (Cheng et al. 2013). In spite of all these applications and the development of Petri Nets, the discrete event simulation is still the favored approach for both researchers and industry to simulate construction operations. The reason for that is often ascribed to the more complex simulation logic of the Petri Nets, however, a clearly formulated statement does not exists.

3.5.10 Agent-directed simulation of construction projects

Agent-directed simulation is a specialization of distributed Artificial Intelligence (AI) for which the experiment is constructed around a set of agents (individual AI) that are capable of interacting with each other and with their environment (Knotts et al. 2000, Sawhney et al. 2003). *Agents* are intelligent software modules with cognitive abilities such as autonomy, perception, reasoning, assessing, understanding, learning, goal processing and goal directed knowledge processing (Ören et al. 2000, Mohamed and AbouRizk 2005). These agents act as the assistants of the user who can sense the conditions of their environment, assess their situation, make decisions and react according to a set of rules (Sawhney et al. 2003).

Ören et al. (2000) differentiated three categories of agent simulations according to the synergy between agent and the simulation. The first one is the agent simulation, where intelligent entities such as humans or intelligent devices are simulated. The second one is the agent-based simulation in which the agents are used to generate model behavior in the simulation. Model behavior can be defined by the main characteristics of the agents and how they act or react under certain condition changes. The third category is the agent-supported simulation. Here, agents are used to support the simulation operations such as “front-end and/or back-end user/system interface operations as well as activities related with simulation software.” (Ören et al. 2000).

Agent-directed simulations are useful to study problems with emergent complexity. Although the individual agents might be simple in nature, due to the interaction between them the whole system becomes capable of modeling complex behaviors (Knotts et al. 2000). They are described as being more robust, flexible and fault-tolerant than traditional systems. A further advantage of this technique is its instinctiveness which helps to better understand the behavior of complex systems (Knotts et al. 2000). By developing an agent-directed system the issues are identified as the representation of the model including the structure of the agents and the system architecture, the communications, system dynamics, the overall system control and the conflict resolution (Shen and Norrie 1999). Furthermore, the model development is a time consuming process, since every possible interaction between agents, and between agents and the environment must be defined and implemented (Kugler and Franz 2007).

The application of agent-directed simulation in construction operations is advantageous for taking safety or space requirements of resources into account (Sawhney and Vamadevan 2000). In such a

simulation, the resources are modelled as agents that try to find the best working area where the available space and safety requirements are satisfied to determine the most effective safety plan. Kugler and Franz (2007) introduced a concept of a multi agent-based simulation for high-rise buildings where the active components of the simulation, such as cranes, labour and vehicles, are modelled as agents.

A similar agent-based scheduling technique of Knotts et al. (2000) has been enhanced by Horenburg et al. (2012). They applied different agents for both activities and resources and put them into an auction-based environment. The agents negotiate with each other as to which activity should be executed with what resources in order to find a schedule with near optimal makespan. Kim and Kim (2010) developed SIMCON, a multi-agent-based simulation system for construction operations to model the effect of congested flows (such as traffic and trucks) on the construction site while taking the layout of the site into account. Kooragamage et al. (2013) introduced the concept of an agent-based model for planning site logistics by focusing on spatial time clashes between the construction objects and resources.

In summary, agent-directed simulations are powerful frameworks to simulate complex systems with large numbers of interactions such as construction operations. However, much research in this field must be done to establish a simulation tool as usable for the industry as the discrete event simulation. The lack of transparency at agent interactions and decision makings, the restricted control over the simulation of the user, and the time consuming model development are the primary reasons why the author of this thesis prefers to use the discrete event simulation for his research.

3.6 Summary and discussion

In the last chapter it has been shown how the simulation techniques are capable of simulating scheduling problems both in the manufacturing and construction industries. After introducing the basic terms of the simulation, the concept and the characteristics of the discrete event simulation were discussed. The discrete event simulation is a type of simulation in which the changes in the system state, so called events, occur only at discrete points in time. The simulation time jumps forward in time between these events while the model´s state stays constant between these discrete time steps. The DES is a powerful tool to solve complex problems such as scheduling under resource constraints. The advantage of the DES over the conventionally used scheduling tools is that it can handle not only precedence constraints, but also resource constraints thereby generating more realistic schedules then the conventional methods. This technique is already widely used in the manufacturing industry, however, it is hardly known in the construction industry.

The advantages of simulations over analytical solutions, include:

- Simulations can handle complex interactions in a simple manner thereby resulting in a more realistic schedule...
- Simulations provide the possibility to test every small detail of the schedule before starting the actual production…
- These tests can lower the risk of collisions and lower the high idle times for machines by identifying bottlenecks within the schedule…
- Simulation methods are easier to understand and use than analytical methods…

Due to these advantages, researchers have worked to adapt simulation methods to the construction industry since the middle of the 20th century. However, this adaption is not a straightforward process due to the differences between the two industries.

The main reasons for the difficulties in adapting the simulation methodology are the uniqueness of the construction projects and the lack of repetitive processes on the construction site. While manufacturing production plans contain many repetitive processes that are used to create the same product many times, the construction of a building is a unique process and the result is always only one "product".

Furthermore, the production lines in the manufacturing industry have a static layout. The products are transported with conveyor belts between stationary machines where they get prepared. This structure also represents the layout of the simulation model. In contrast, in the construction industry, the building (the product) is stationary and the resources are mobile. Thus the layout of the construction site is dynamically changing in time which is a complex job to model with simulation. Additionally, in the manufacturing industry one operation can only be executed by one specific machine. In contrast, in the construction industry one construction task might be executed with different resources without impacting the schedule. Hence the layout of the simulation model and the connections between resources and construction tasks are more complex for the construction industry than for the manufacturing industry.

The developed simulation frameworks for the construction industry, e.g. CYCLONE, STROBOSCOPE, Symphony and further frameworks, are mainly focusing on modeling repetitive construction processes like earth hauling processes, tunneling, or road construction. To determine a schedule for construction projects with non-repetitive processes such as the construction of high-rise buildings and bridges, the models using the aforementioned simulation frameworks become very complex. Furthermore, due to the applied rigid sequences of activities, these simulation frameworks are not capable of modeling the dynamically changing environment of a construction site.

Based on these theories, in order to develop a method that could handle both precedence and resource constraints in a simple and straightforward way while still being able to simulate non-repetitive construction processes, Beißert et al. (2007b) developed a new constraint-based simulation method that has also been used by the author for scheduling purposes and will be introduced in the next chapter in detail.

4 Discrete event simulation for generating construction schedules

4.1 Executive summary

The optimization of schedules is only possible if the order of single construction tasks can be rearranged within the schedule and the rearrangement results in a feasible schedule. The optimization of schedules is a combinatorial problem due to the arrangement of the single tasks. Combinatorial problems can efficiently be modelled by the constraint satisfaction paradigm (Rossi et al. 2006). A group of German researchers (König et al. 2007a, König et al. 2007b, Beißert et al. 2007b) integrated the constraint satisfaction paradigm into a discrete event environment and with this they created a new constraint-based discrete event simulation approach (CBDES). The overall goal of the group was "to simulate different practicable solutions, which can be analyzed regarding principal guidelines such as time, cost and quality" (König et al. 2007b). Since the CBDES is a method to generate single feasible schedules for the resource-constrained scheduling problem (RCPSP), it is able to generate feasible schedules for the optimization of construction schedules. This will be introduced in Chapter 7. The working mechanism of the CDBES will be introduced in detail in the following sections.

One highly criticized part of the simulation-based scheduling is the time-consuming process of preparing the data. Therefore a new 3D geometry-based method has been developed in order to accelerate these tasks for the simulation and to make the simulation more appealing for industrial use. The methods developed for data preparation before the data is inserted into the simulation will be presented in Section 4.3.

The benefit of the network planning techniques over the simulation is the capability to determine float time. A new technique has been developed in the frame of this thesis that extends the discrete event simulation technique with the capability of determining float time for each individual task. This increases the competitiveness of the simulation approach relative to the network planning techniques

and makes it more attractive for industrial use. A detailed description of this technique will be presented in Chapter 5.

4.2 Constraint-based modeling of construction operations

A constraint satisfaction problem (CSP) is defined as a triple $P = \langle X; D; C \rangle$, where X is a set of variables, D corresponds to the domain of the values and C is a tuple of constraints (Rossi et al. 2006). The solution process of the CSP searches for a value combination of the variables that satisfies all the constraints.

In the case of a construction operation system, activities and resources are represented as variables of the CSP, with resource requirements of activities and their technological dependencies as constraints. Hence, a feasible solution (schedule) will satisfy all of the sequential dependencies between the activities, the resource requirements and the resource limitations.

4.2.1 Elements of a constraint-based discrete event simulation for construction operations

As introduced in Section 2.2 and 2.4, a construction project can be formulated as a resource-constrained project-scheduling problem (RCPSP). The basic components of a construction project are the construction tasks that can be modeled as a pair of events or an activity in a discrete event simulation environment (Section 3.3.1). Construction tasks also build the basic components of the CBDES but they are modelled as simple variables that have certain restrictions and requirements such as technological dependencies and resource needs. However, a task represents the same procedure as an activity within the discrete event simulation environment, because due to the implemented constraint satisfaction methodology it is represented as a variable and not as a pair of events. It is of utmost importance to differentiate between the names. Construction tasks have three states: not started, in progress and finished. In the basic version of the CBDES after a task has been started it cannot be interrupted during its execution. However, extensions have been developed that allow the interruption of tasks during their execution (Rahm et al. 2012).

Resources such as working equipment, man power, material or workspace are necessary to execute construction tasks. Resources can be categorized into two classes. The *non-renewable resources* can only be used once for a construction task at the construction site. After the work is terminated, these resources are not available for other tasks. Non-renewable resources include rebar cages, specific wood elements and precast concrete or formwork elements. In contrast, *renewable resources* can be used to execute several construction tasks, one right after another. After one task has been terminated machines, labor, construction space and other renewable resources will be available again for use on another task. Resources such as labor might have diverging *skills*. Skills are used to describe the nature of tasks a resource is capable of executing. One laborer might be trained in pouring concrete, another one in assembling formwork and a third in both tasks. Resources can have two states: available or in use.

Constraints describe boundary conditions for the construction site that are necessary to erect the building. Sriprasert and Dawood (2002) introduced three class of constraints in the construction industry. The first class is composed of the *physical constraints* such as technological dependencies (e.g. task A must follow task B), space requirements, safety and the environment. The second class

contains the *contract constraints* such as deadlines (time), budget (costs), quality and special agreements. *Enabler Constraints* form the third class of restrictions and involve constraints that pertain to the resources such as requirement, availability, capacity, perfection and continuity. Additionally *constraints about information* such as requirements, availability and perfection are included.

In addition the two types of constraints, *hard-* and *soft-constraint,* can be distinguished. Hard-constraints define stringent conditions for a construction task and must be fulfilled before the task can be started. Technological dependencies, resource needs and safety criteria belong to this type of constraints. Soft-constraints, in contrast, define appropriate conditions, the fulfillment of which is not completely necessary (Rossi et al. 2006). Soft-constraints can be used to implement productivity requirements such as a relation between the workers' productivity and the available work space (Beißert et al. 2007b, Beißert et al. 2010). In this research the author will only use hard-constraints for the developed methods.

4.2.2 Simulation concept of the constraint-based discrete event simulation

As mentioned in the introduction of the chapter, the constraint-based approach is integrated for the new simulation technique into the discrete event simulation. The discrete event framework is used to inspect points in time where a construction task can be started or terminated. The constraint checking algorithm is implemented within the start event of the tasks (Figure 4-1), which searches for construction tasks that can be started. A second event is applied in order to finish a task and release its used resources. The complete simulation is constructed of two general events: the start of construction tasks and the termination of construction tasks (Beißert 2012).

The course of actions for the start of construction task is depicted in Figure 4-1. When a new event occurs, first, all the constraints with the state of "not started" will be checked. These not started tasks are stored in a queue for not yet started tasks. The constraint checking always starts with the first task on the list and checks if all its constraints are fulfilled. If one of the constraints is not fulfilled, the constraint checking process of the task will be skipped and the constraints of the next task on the list will be checked. This process continues until the end of the list is reached. When all the constraints of a task are fulfilled (including precedence and resource constraints) the task will be started, its needed material, resources and spaces will be locked and the necessary new events will be generated. As soon as there are no more executable tasks available, the simulation proceeds to the next point in time where an event occurs (Beißert 2012).

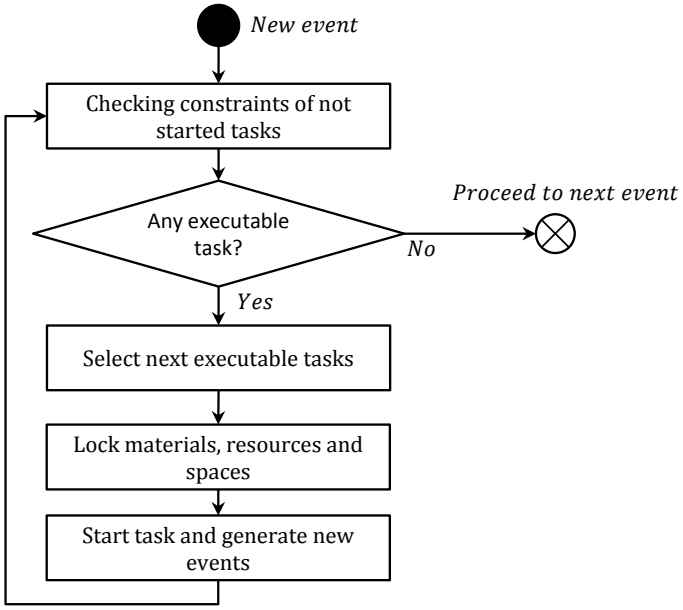


Figure 4-1: Structure of the start event for the CBDES (König et al. 2007b, Beißert et al. 2007b, Beißert 2012)

An important characteristic of the introduced start event is that even when more tasks could be started in parallel, it is always the first task on the waiting list that will get the available resources, while the other tasks have to wait until the necessary resources are once again available. Therefore, the order of the tasks within the list of not started tasks plays an important role for the generation of schedules. Varying the order of the tasks within the list will result in different execution sequence of the tasks. Therefore, different schedules for the same project can be generated with the same boundary conditions (e.g. same amount of available resources).

In the example presented in Figure 4-2 three tasks are waiting to be scheduled: A, B and C. They all need one unit of the same resource from which two units are available on the jobsite. By varying the order of the task list, different schedules can be generated.

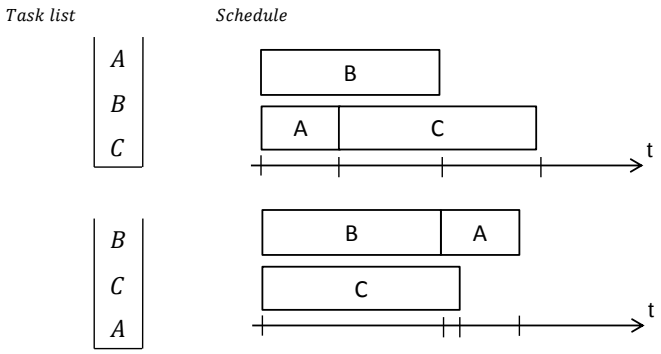


Figure 4-2: Different generated schedules based on the order of tasks within the task list

The ordering of the tasks within the task list can be based on random selection or on different strategies. Strategies can be implemented by soft-constraints (Beißert et al. 2010), which define diverging priorities for the tasks. Depending on the fulfillment of the soft-constraints a task might get a higher priority than others and thus it ascends in the task list. The higher the priority of a task, the higher its position in the task list.

As soon as the end of the task´s execution is reached the "finish of construction task" event will be triggered. It is responsible for releasing the resources (setting the state of reusable resources to *available*) and to set the state of the task to *finished* (Figure 4-3). After tasks have been terminated a start of construction tasks event will be activated to look for tasks that could be started with the new released resources.

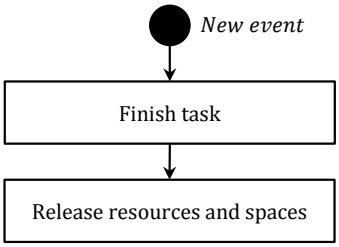


Figure 4-3: Finish of a construction task event in the CBDES (König et al. 2007b, Beißert 2012)

The simulation proceeds until all the tasks are terminated, or until there are no more tasks that can be started. Every event and resource allocation is recorded during a simulation run, thus a detailed evaluation of the results is achievable. One simulation run always calculates one feasible schedule with a corresponding resource utilization. Flexibility in scheduling can be implemented by applying different strategies or modified task priorities. In this case, the order of tasks within the executable queue will be diverging and so, the execution order of the tasks will also be diverging. Thus for the concerned project with the same precedence and available resource configuration diverging schedules can be generated. Further changes can be introduced by varying the input data of the simulation, such as the sequence of the tasks or the available resources. Thus, different practicable solutions might be generated for one specific scheduling problem, which can be analyzed regarding the makespan, the costs of the project and the quality of the built components (König et al. 2007b).

In order to generate a schedule, DESMO-J, a discrete event-based simulation engine (Page and Kreutzer 2005) that has been extended with the constraint-based methodology (König et al. 2007b) is used. The necessary input data is prepared by means of the Preparator (Section 4.3.5 - Dori and Borrmann 2011) and is imported into the simulation engine. This discussed simulation technique is called *forward simulation* and it will also be important for the determination of float time under the consideration of resource constraints (Chapter 5).

4.3 Preparing the necessary data for the process simulation

Before a simulation run can start, every necessary input data must be defined beforehand. This includes information as to the following:

- construction tasks;
- the resources required for each task;
- interdependencies between the tasks,
- available resources;
- performance factors of the resources.

However, the constraint–based discrete event simulation does not need an explicitly defined activity cycle diagram (ACD) like the common discrete event simulation. Instead, it requires a list of

tasks and their constraints (Wu et al. 2010a). Despite this simplification, the input data preparation for the CBDES can become just as time consuming as for the introduced discrete event simulations. Defining the necessary input data is mostly performed manually. This includes tasks and their dependencies. For example, a small bridge construction site contains 32 tasks and 50 precedence relationships. However the same preparation process for the second case study (Section 7.6.2) with 457 tasks and 523 precedence relationships would be a very complex and error-prone task when done manually (Wu et al. 2010a). Therefore half- or fully automated computer aided methods are necessary to support the planner and make this process faster and safer.

4.3.1 Related work in data preparation for discrete event simulation

As already introduced in Section 3.5.7, many researchers integrated 3D modeling software such as AutoCAD or MicroStation into the simulation environment to accelerate the time consuming process of defining input data, to visualize the results and to make the simulation-based scheduling more attractive for the industry (Zhang et al. 2000, Dawood et al. 2003, Kamat and Martinez 2003, Tulke and Hanff 2007and ElNimr and Mohamed 2011). These two methods enhance the data preparation process of the simulation, however, they rely on the utilization of the predefined modeling system components. Tulke (2010) gives an overview of further existing strategies for data preparation for 4D-simulations[12] by assigning construction tasks to 3D-objects. He identified five existing methods. These five methods include the manual linking of every single task to a 3D-object, linking temporal data to 3D-objects within a CAD software framework, creating a consistent hierarchic structure for the process and the product model (Dawood et al. 2003), the use of databases for the selection of building components based on its object type and further attributes and lastly, an interactive preparation of a schedule in a 3D environment with simultaneous specifications for 4D visualization (Zhou et al. 2009).

Wu et al. (2010a) introduced a sophisticated approach that combines some of these approaches and enables schedulers to use any kind of 3D-model to generate input data for simulations, independent from the modeling system. The presented methodology aims to receive the necessary input data for the simulation interactively by assigning construction tasks to 3D-model components. To enhance this quite time-consuming manual process (Koo and Fischer 2003, Tulke 2010), Wu et al. (2010a) applied a hierarchical approach that allows the planner to subsequently refine both the product model (product break down structure) and the process model (work break down structure). Furthermore, they developed a pattern-based approach for the assignment process that collects every task that needs to be performed for the selected building component and merges them into one construction *process pattern.* This approach considers the internal interdependencies and resource needs. Since the author of this thesis used the introduced methodology to define input data for his research, it will be introduced in detail.

[12] 4D-Simulation is a framework where the simulated schedule will be visualized with the use of a 3D-model and a linking of temporal information to the 3D-components.

4.3.2 Levels-of-detail approach

To enhance the manual construction task assigning process to 3D-model components a *levels-of-detail approach* is applied. It allows planners to start their work with a coarse and simple model of the project and refine and decompose its components step-by-step according to the *work breakdown structure* (WBS) of the project. WBS is used to analyze the structure of a project and to show the relationships between tasks and components. Its advantages lie in the systematic representation of the construction project's structure (Melzner et al. 2012). Wu et al. (2010a) introduced a hierarchic assigning process of construction tasks based on the parallel refinement of the process and product model of the project. The advantage of such a model is that general attributes or characteristics of elements on higher levels can be applied to both process and product model elements at the same time. These will automatically be stored in further refined elements. Furthermore, the parallel refinement of the process and product model provides a straightforward connection between processes and geometry at any level-of-detail.

On the highest level of both models are the construction elements of the building and the building itself, which form the coarsest elements. With each step the models will be deconstructed and refined and further information might be assigned to the components, such as the construction method of the building or bridge on the first level (e.g. balanced cantilever, standard false work, etc.). Based on the decision made at the first level the components of the second level can be generated automatically when extended with further information (e.g. amount of spans).

On the second level of the model hierarchy are the *elements* of the building, such as the abutment or a pier of a bridge or the different stories of a high-rise building. The second level of the process model frames the corresponding construction processes, such as the construction of the abutment or the pier. On the second level materials can be assigned to the process model components thereby generating the components of the third level.

The *sub-elements* are located on the third level. These are refined components of the elements, such as the foundation of a pier or a piece of wall on a specific story. The third level of the process model collects the corresponding construction tasks such as the construction of the foundation or the wall.

At this point *construction methods* can be selected that contain single construction tasks that can be generated in the fourth level of the process model, depending on the material of the component. Corresponding product model components will be generated automatically. An example of the refinement process is shown in Figure 4-4. Here a bridge construction project is decomposed to its elementary components based on the methods of Wu et al. (2010a).

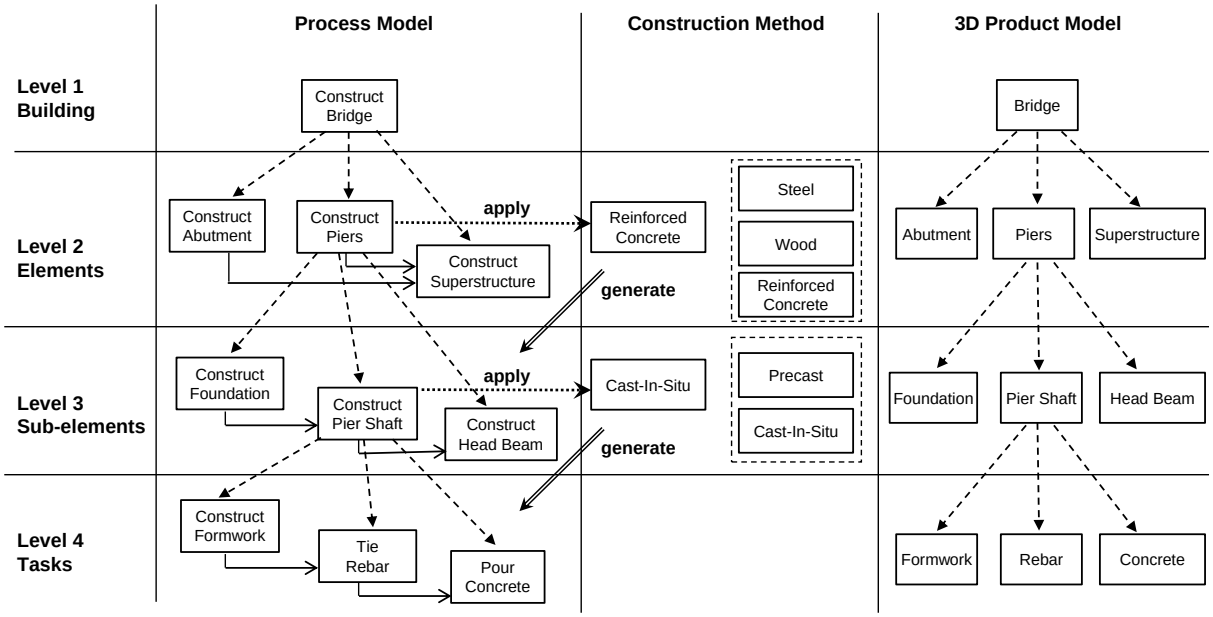


Figure 4-4: Representation of the *levels-of-detail* approach on a simple bridge project. Normal arrows: precedence constraints, dashed arrows: hierarchic decomposition, dotted arrows: construction method selection, double arrows: automatic generation of new components (Wu et al. 2010a).

Although the presented example is made up of four levels, the number of levels can vary with the type of the modelled building and the selected construction methods (Marx and König 2011). However, the finest components of the process model will always represent the atomic tasks which will be used by the subsequent constraint-based discrete event simulation (Wu et al. 2010a).

4.3.3 Process patterns and activity packages

The aforementioned construction methods that can be assigned to the components of the process model are formalized as predefined process patterns. "A process pattern combines a number of process components and their precedence relationships and thus represents a companies' knowledge of how to execute certain construction methods. This is used to generate the process components for the next level-of-detail" (Wu et al. 2010a). Process patterns significantly accelerate the preparation time of process models, since the operator only selects the appropriate method after which further components are generated automatically. In addition, the operator may change or introduce new process patterns that will also be stored in the list of available patterns where they can also be applied in other projects. The author of this thesis was focusing on creating process patterns primarily for bridge construction projects. Process patterns for other fields of construction are presented by Marx and König (2011). A simple process pattern for cast-in-situ built reinforced concrete components is presented in Figure 4-5.

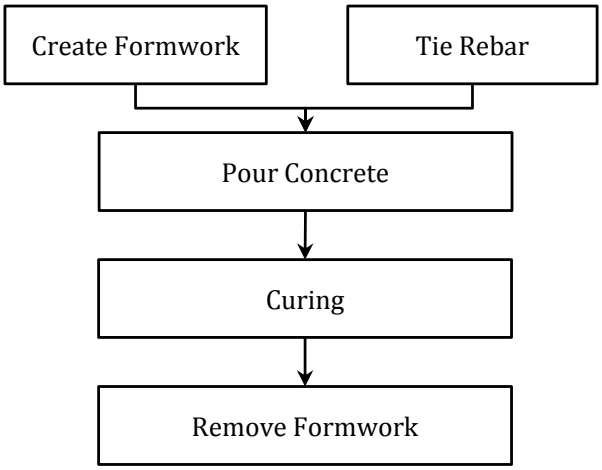


Figure 4-5: Process pattern for cast-in-situ constructed reinforced concrete building component (based on Wu et al. 2010a)

As explained at the end of the last section, the tasks on the finest (last) level of the process model play an important role, since they will be used for the simulation. Therefore they need to store more information than the other components of the model. Hence, the *activity packages* were introduced that store all of the necessary data about the task that is needed for the simulation. Examples of data that is necessary for a task include the name of the task, all pre-conditions, the building component that the task belongs to, its material and all necessary resources (Wu et al. 2010a). An activity package is presented in Figure 4-6. When selecting a construction method for a level 3 process component, the corresponding activity packages will be generated automatically and inserted into the process model.

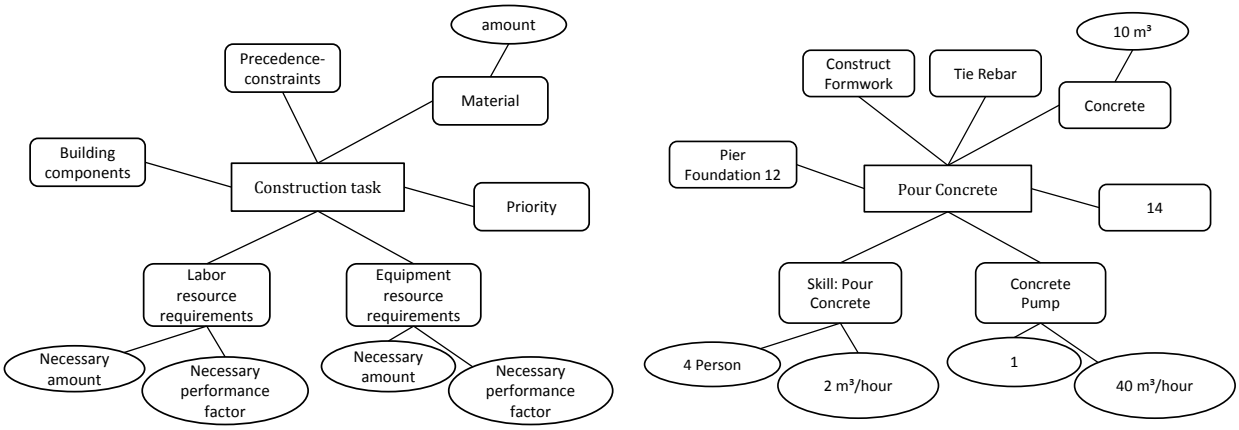


Figure 4-6: Activity package. Left: content, right: example (based on Wu et al. 2010a)

4.3.4 Creating the precedence graph

When the introduced hierarchic deconstruction process is completed, the precedence graph of the construction tasks can be represented. A simple example for the hierarchic precedence graph is presented in Figure 4-7.

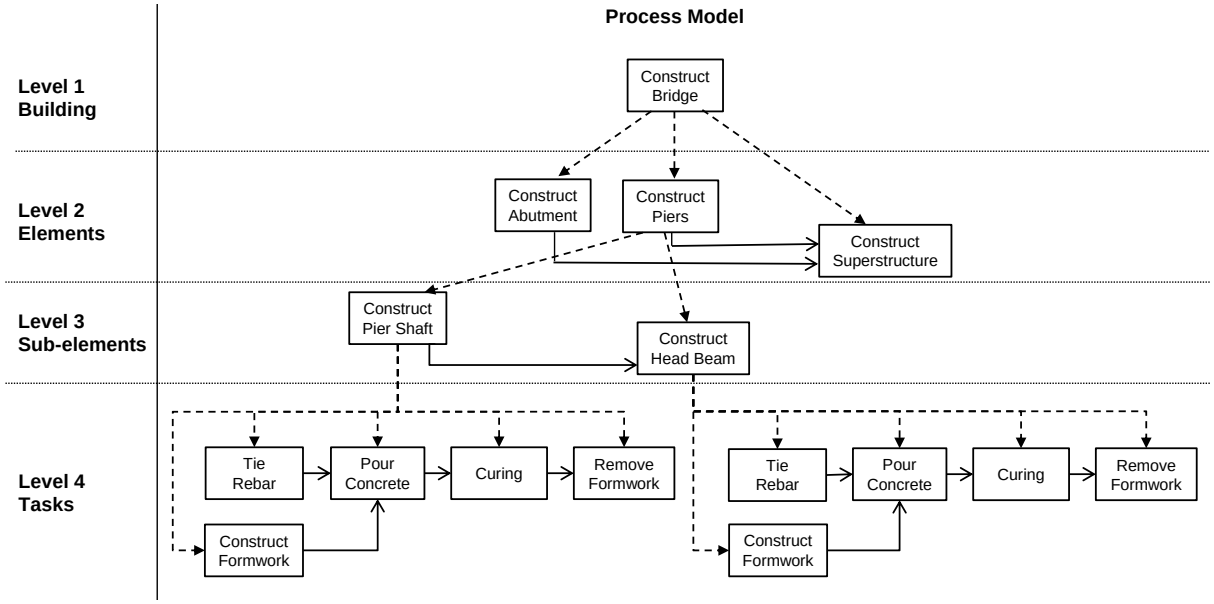


Figure 4-7: Hierarchic precedence graph representation of a simplified bridge construction project´s process model. Arrows: precedence constraints, dashed arrow: hierarchic decomposition

Before starting the simulation of the project, the precedence relationships defined at higher levels (every level except of the finest (last) and the first level) must be mapped to the tasks on the finest (last) level. This step is crucial, because only these tasks are imported into the simulation.

Therefore, the precedence relationships of higher levels will be mapped to the finest (last) level automatically. This occurs step-wise from the highest to the lowest level. By mapping one precedence relationship all the tasks of the predecessor component that have no successor within the component will define a new constraint as precedence. A simple illustration of the method is depicted in Figure 4-8.

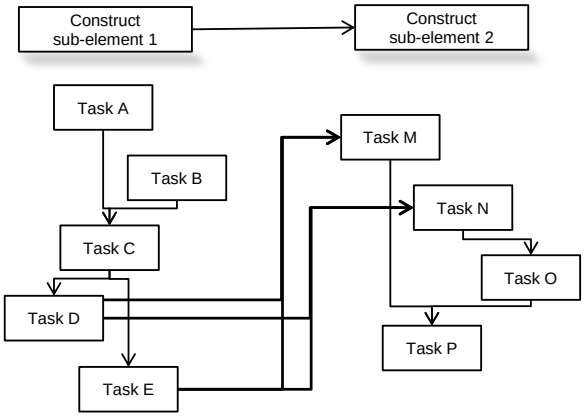


Figure 4-8: The process of mapping high-level precedence-constraints to the task-level: set a new constraint between all the tasks of the predecessor component that have no successor within the component (task D, E) as precedence of all the tasks in the successor component without predecessor within the component (task M,N)

However, the automated mapping of higher-level dependencies might lead to misinterpreted technological dependencies on lower levels, as introduced in the next small example. In this case, the higher level precedence constraint between the two sub-elements of Figure 4-7 (Pier Shaft and Pier Head) define a strict execution sequence of their tasks such as represented in Figure 4-9. However, constructing the formwork and tie rebar of the pier head beam does not require removing the formwork of the pier shaft, because they could start after the concrete is cured. The corrected precedence graph is represented in Figure 4-10.

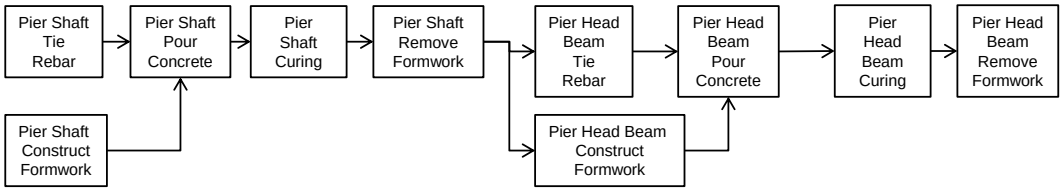


Figure 4-9: Misinterpreted higher-level precedence relationship between the sub-elements of Figure 4-7. The formwork construction and rebar assembly of the pier head beam can already begin when the concrete of the pier shaft is cured.

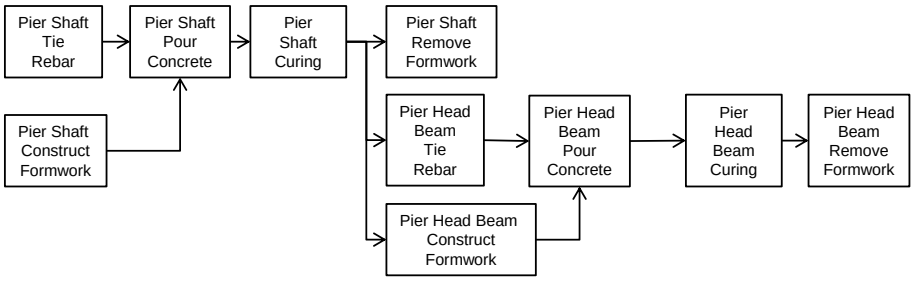


Figure 4-10: Corrected precedence graph of the introduced task sequence in Figure 4-9

To avoid such misinterpretations, the mapping of higher-level precedence constraints always has to be controlled and corrected by the operator before starting the simulation. This can be obtained by a simple graph visualization tool that enables the reconnection of precedence constraints between construction tasks.

Since for the simulation only the construction tasks on the finest (last) level are necessary, their graph will be cut off of the hierarchic tree when it is completed. It is important to mention that, even if this precedence graph (such as in Figure 4-10) looks exactly like the simplified activity cycle diagram (ABC) of Shi (1999) (see Section 3.5.5), it has a completely different meaning. The arcs within the precedence graph for the CBDES represent only precedence relationships between the tasks, while the arcs in an ABC also represent the resource flow between the activities.

4.3.5 Preparator

To support the planner in preparing the necessary input data for the simulation a software called *Preparator* has been developed by Wu et al. (2010a). The Preparator is a graphical user interface (GUI) that is used to establish the precedence graph of a construction project by connecting the predefined process patterns to 3D-model components of the building or a bridge. Connecting it to a 3D-model not only simplifies the assigning process of tasks and the preparation of the precedence graph, but also allows for the control of the results of the task assignment to get data for the quantity take-off for the tasks and to visualize the results of the simulation. When all tasks and dependencies have been created the graph can be delivered to the simulation.

The functionalities of the Preparator software have been enhanced by the author to define all necessary input data for the simulation. The GUI of the Preparator is presented in Figure 4-11. The window for the 3D-model is placed at the bottom in the middle of the screen. On the left side of the window is the list of assigned and non-assigned objects. Above the list of objects there is a list of the necessary construction methods (process patterns) to build up the hierarchic process models. At the top of the window in the middle there is a list of tasks that belong to the selected process pattern. Next to the selected task its precedence relationships are listed. Below the list of tasks is the customization window for the activity packages. With these functionalities the precedence constraints and the necessary resources for a task can be defined. On the right side of the screen, the current hierarchy of the process model is located. When selecting a 3D-object in the 3D-model window, the object will be selected in the object list and a process pattern can be assigned to it. After customizing the attributes of the corresponding construction tasks the pattern can be added to the hierarchy and the next 3D-object can be selected and so on (Dori and Borrmann 2010) until the graph is completed.

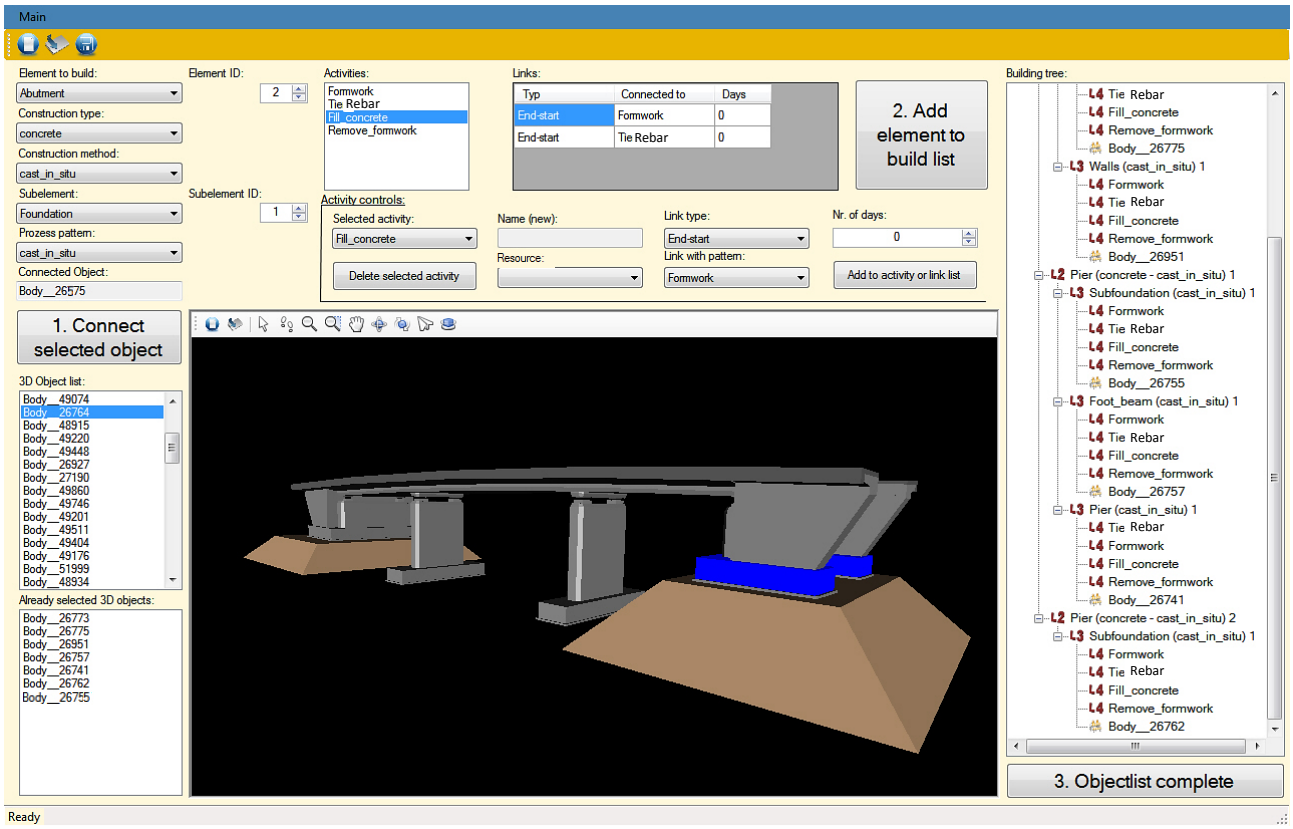


Figure 4-11: The graphical user interface of the Preparator

Higher level precedence relationships can be revised on the main window of the Preparator when the assignment process is completed. A further extension is the *Resource manager*, where the available machines and labor for the simulation can be defined. These further extensions make it possible to join all the necessary data in one tool.

Another software tool to generate input data for the CBDES has been developed at the University of Bochum in Germany. The *SiteSim Editor* is based on the same principles as the Preparator but it has been enhanced with further functionalities. These additional functionalities include the ability to define spatial constraints and strategic constraints (Marx and König 2011). Spatial constraints assign working space to construction tasks, which must be free in order to execute the task. When all the precedence and resource constraints of a task are fulfilled and the working space is also free, the task can be started. Otherwise, it must wait until the workspace is free again. Strategic constraints create grouped assignments for tasks such as pouring concrete, as represented for three columns in Figure 4-12. This kind of assignment is important in order to have both a high degree of utilization and to save costs and time for bigger machines.

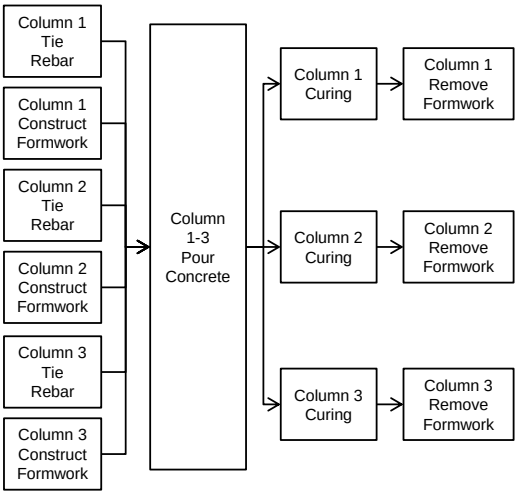


Figure 4-12: Grouped assignment for pouring concrete to three columns

Melzner et al. (2011) introduced a *BIM*- (BIM - Eastman et al. 2011) and an integrated *dynamic data source*-based preparation method that is used to define input data for the process simulation. The advantage of a BIM is that the objects contained in the model provide information about the construction data such as material, time and cost. They can be read automatically for the quantity take-off and to automatically generate construction methods for 3D-objects. The construction methods are similar to the process patterns and they are stored in the dynamic data source.

Tauscher et al. (2007) developed a *Case Based Reasoning* (CBR) scheduling tool called *Alice* that has also been used by Mikulakova et al. (2010) and Hartmann et al. (2012). The software automatically compares building components of a BIM with the construction parts that have already been successfully constructed and stored in a *database*. It then suggests related applicable construction methods for the construction of the building component. Further methods can be added by the operator if the desired method is missing from the database.

Melzner et al. (2012) suggests the standardization of the input data for the simulation in order to reduce the effort of model generation.

4.3.6 Summary of the introduced methods

In the last sections methods have been introduced that aim to accelerate the time consuming input data preparation for process simulations. Since every construction project is unique, in spite of automated hierarchic modeling, process patterns, activity packages and 3D-model connection, the process of creating the task sequence of an entire project is still time consuming due to the large amount of customizable data that is required (e.g. performance factors, necessary resources for individual tasks). BIM and knowledge-based methods such as the case-based reasoning, provide a good basis for overcoming these issues and for an automated precedence graphs generation for the entire construction project. However, further research is necessary to improve the quality of this approach (Mikulakova et al. 2010). Further research should investigate how the creation of custom construction methods and input of custom data could be accelerated. Another important field of research is the automated definition of different construction scenarios for the same project. This would allow the planner to not only compare schedules with different task sequences, but also schedules with different construction scenarios.

5 Determination of float time with constraint-based discrete event simulation

5.1 Executive summary

The creation of a detailed schedule for a construction project that takes into account the available resources and precedence relationships between the various construction tasks is a challenging task for construction managers. The key measure of flexibility within the schedule is the *float time* (Raz et al. 1996). Float time can be differentiated into two classes. The *total float time* describes the time frame within which the execution of a task can be moved, or the flexibility that exists to complete the task without impacting the completion time of the project. The *free float time* describes the amount of time that a task in a project can be delayed without causing delay to the subsequent tasks (Figure 5-1).

The current standard approach to determine float times in the construction industry is to use network scheduling techniques such as the Precedence Diagram Method (PDM), the Critical Path Method (CPM), Program Evaluation and Review Technique (PERT). A detailed description of these methods was provided above in Section 2.2.5. Where material and resource restrictions do not need to be considered, i.e. only precedence relationships of tasks are crucial, the schedule can be analyzed by applying one of the network scheduling methods. In these situations the calculation of float times for tasks is a straightforward procedure. Starting with the first task, the execution time will be added to the start time. The result is the *earliest end time* of the first task as well as the *earliest start time* of the successor tasks. After finishing this process, called *forward-pass*, the earliest possible start time is calculated for each task.

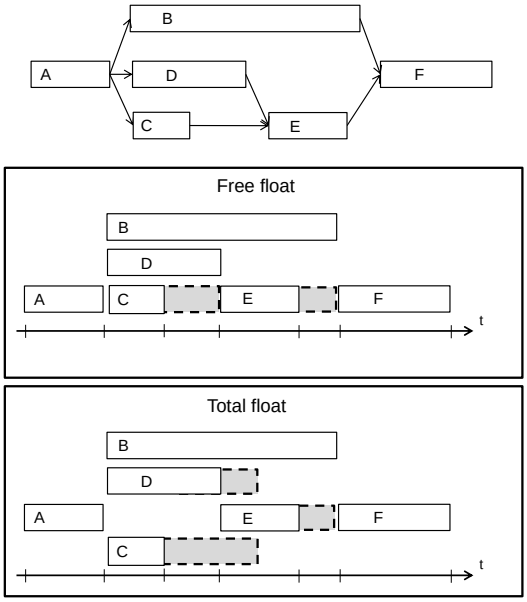


Figure 5-1: Difference between the free and the total float (top: precedence graph of the tasks, middle: free float, bottom: total float, dashed box: float)

Subsequently, a *backward-pass* is conducted, which works in a similar manner. Starting with the last tasks, i.e. those that have no successors, the *latest start and end times* of all the tasks are calculated as discussed in Section 2.2.5.2. The difference between the latest start time and the earliest start time defines the *total float time* of a task. The free float of a task can be determined by subtracting the latest finish time of the task from the earliest start time of the earliest successor of the task. Hence, to determine the free float of a task the backward pass is not necessary. Therefore in this thesis the author will concentrate only on the determination of the total float time of every single task. The tasks without total float time constitute the *critical path*. A delay in any of these tasks will result in an increase in the project's overall time span and therefore in a delay of the project itself. For this reason it is important to determine total float times for each task, in order to identify those tasks that are part of the critical path or have only short total float time.

If the availability of resources is limited for a project, the results that have been calculated with one of the conventional network scheduling techniques can be misleading. Using the PDM/CPM/PERT method alone it is not possible to consider anything other than precedence constraints. Therefore, the network scheduling techniques cannot determine feasible schedules and total float values under limited resource availability. In this section a new approach will be presented that makes it possible to calculate float times for all tasks in a resource-constrained project schedule.

To determine a schedule, the constraint-based discrete event simulation discussed in Chapter 4 is used. This simulation technique is capable of determining the earliest start date for each task while taking into consideration the resource constraints. However, in order to determine total float time the latest start date of a task is also necessary. In the past, this information was not available with the introduced simulation technique. Therefore, a new method will be introduced that extends the constraint-based discrete event simulation so that it is able to determine float time for construction tasks. In a manner similar to the forward- and backward-pass analysis of the network scheduling techniques, this approach combines the forward and backward simulation concept with additional methods that apply restrictions to obtain an identical order of tasks for the backward simulation and the forward simulation.

The following section gives a review of the different approaches analyzed in the available literature. The latest research on the topic of float calculation using network scheduling and further techniques will be presented. The Sections 5.3 and 5.4 describe in detail the concept of calculating float times using combined forward and backward simulation. The limitations of the introduced method is discussed in Section 5.5. A comprehensive case study is presented in Section 5.6, and in Section 5.7 the same case study with the addition of multiple resource needs is analyzed. The final section, Section 5.8, contains concluding remarks and an outlook on future research objectives.

5.2 Related work on float time determination

The most widely applied approach in today's construction-process planning is the CPM. As Kelley (1961) has shown, this approach is optimal for construction schedule planning because it minimizes the overall project duration. However, this approach only works without resource restrictions. Scheduling in situations where there are resource constraints is a much more complex problem. Raz and Marshall (1996) point out that the results can be misleading when the CPM is applied to float determination while taking a limited amount of resources into account. Using CPM alone, the situation can occur where the resource limits will be exceeded (Figure 5-2).

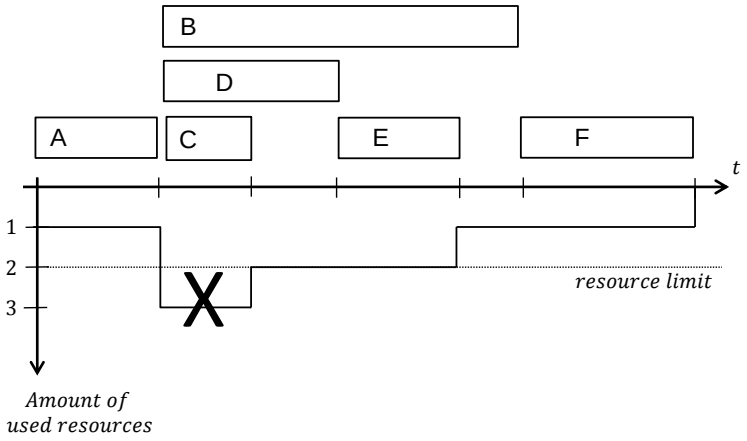


Figure 5-2: Exceeding the resource limits within a schedule determined by CPM. (Precedence graph of the tasks is depicted in Figure 5-1)

To solve this overutilization problem, heuristic methods can be applied to level the resources and decide to which of the competing tasks the resources should be allocated to first so as not to exceed the resource limits (Willis 1985, Raz and Marshall 1996, Hegazy and Menesi 2012). As a result, some tasks may be scheduled later than their actual latest execution dates under the CPM analysis because the required resources are not available. These conflict with the scheduled dates of the tasks and therefore the float times are no longer applicable. Possible alternative solutions are introduced in Figure 5-3.

To solve this problem, Raz and Marshall (1996) propose new float definitions that retain the original meaning of float. They introduce the *scheduled total float* (and *scheduled free float*) which is the difference between the latest scheduled date and earliest scheduled date of a task. Scheduled dates fulfill both the precedence and resource constraints of the task and the resource limits of the project. To determine these scheduled dates a resource-constrained forward- and backward-pass is

carried out using heuristic algorithms for resource leveling. Using their approach it is possible to obtain a more precise and accurate measure of schedule flexibility and risk.

Goldratt (1997) introduces the definition of the *critical chain*, a series of tasks satisfying both precedence and resource constraints (equals critical path with satisfied resource constraints and limits). A delay in one of these chain tasks would extend the total duration of the project. *In this thesis, the author will also use this definition for critical tasks taking resource constraints into consideration*. Bowers (2000) redefines the definition of float time. He calculates the float time as the difference between the earliest and the latest start time of the task of *all possible schedules* with identical durations. The article opened a discussion between Lu and Kim and de la Garza (Lu et al. 2006) where the authors suggested using a common set of examples that the authors could use to test and evaluate the performance of each of their methods.

Lu and Li (2003) introduced the Resource-Activity Critical Path Method (RACPM), in which they apply further precedence constraints to activities that require the same resources (resource-activity combined precedence relationships) in order to determine the tasks float time. These constrains are set between a currently investigated activity and its resource-constrained successor activities. These include the immediately following activities that in part or in total involve the resources used in the currently investigated activity.

Kim and de la Garza (2003) introduced the Resource-Constrained Critical Path Method (RCPM) that describes resource links for the backward pass. The resource links are created between tasks if an activity must be delayed because of a resource constraint. They evaluated their method (Kim and de la Garza 2005) by comparing the results of the RCPS with other studies, among others with the RACPM (Lu and Li 2003) and the methods of Bowers (2000). As a result they summarize that "Bowers' method does not consider technological relationships while identifying resource links" and "Lu's method may generate a large number of redundant resource links…these do not cause any time calculation errors, but the complexity of the scheduling network will be significantly increased." The RCPM does not detect certain resource links when they do not affect the total floats of activities, so that RCPM procedure to find resource links is required whenever the schedule is updated." With the introduced advantages the RCPM could be a more practical tool for solving scheduling problems than the other methods (Kim and de la Garza 2005).

Lu and Lam (2008) state that CPM is not able to handle resource calendars for the total float determination. They propose a new method for eliminating this drawback based on forward-pass analysis alone. They generate a schedule under resource limit and resource calendar constraints and obtain the base project duration. They then select a task and extend its duration iteratively by one day. In every iteration step a resource leveling analysis is run and the project duration is updated. As long as the updated project duration is not greater than the base project duration the total float time of a task gains is extended by one extra day. This process continues until all the float times for all the tasks have been calculated.

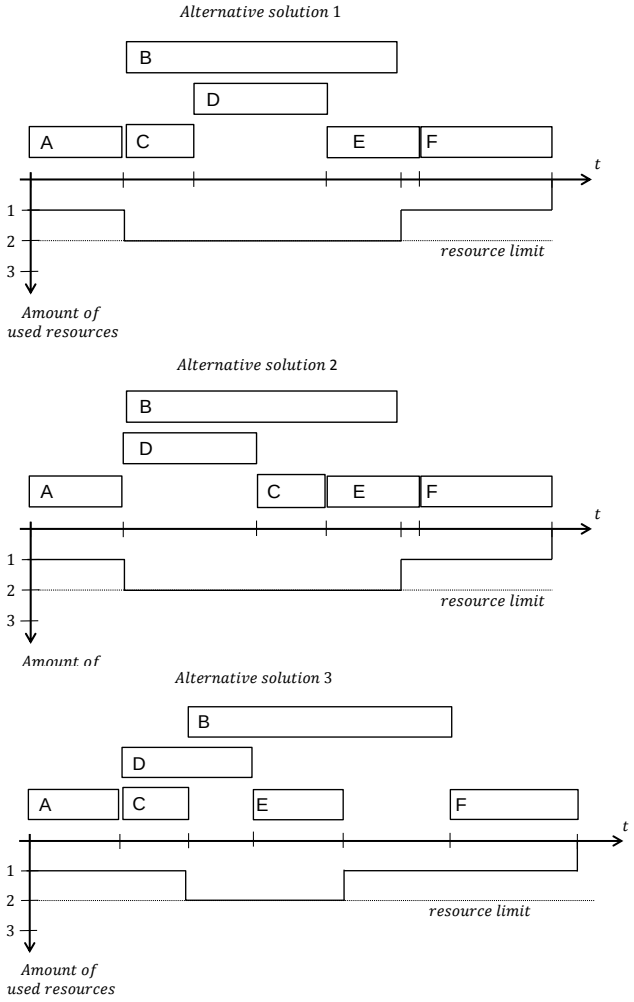


Figure 5-3: Alternative solutions for the in Figure 5-2 introduced resource limit exceeding problem: pushing one of the tasks B, C or D forward in time.

Moreover, Hegazy and Menesi (2010) point out that the CPM algorithm has no mechanism for considering multiple resource constraints, such as deadline and limited resources. It is therefore often difficult to generate schedules that are feasible for all constraints, i.e. one solution in response to one constraint may conflict with another solution for another constraint. Hegazy and Menesi's proposed solution is a mechanism that is based on a finer level of granularity. This is achieved by breaking down the task's duration into separate time segments. As described above, researchers have proposed improvements to CPM, in order to determine more accurate and more realistic information about float time. In addition to the segmentation technique, the researchers have proposed alternative heuristic approaches, e.g. genetic algorithms to generate a schedule that can satisfy both deadline and resource constraints (Hegazy and Menesi 2012).

Lim et al. (2011) discuss the limitations of the currently used approaches in interpreting flexibility of resource-constrained projects. They point out that an appropriate definition of float cannot be based on one single schedule because multiple schedules can exist with the same project duration. Therefore they redefine the notion of critical activity with a broader scope: "*An activity is a critical activity if the maximum of its float values is zero for a specified completion time of the project*."

The total float time definition of the author differs from this last statement. Due to the resource constraints, varying the execution order of the tasks may result in the same project makespan but with

a different critical chain. Such a configuration might conclude that there are only a few or even no critical tasks within the schedule.

To discuss this behavior let us consider an example with three tasks with the same duration, without precedence constraints and with the same resource needs (Figure 5-4). When there are only two units of resources available, it is obvious that one of the tasks must be pushed forward in time so that two of them can be executed simultaneously. Therefore, there are three different schedules that can be generated for the three tasks without exceeding the resource limits. The three different schedules have exactly the same duration since the three tasks have the same duration. Therefore, in order to keep the scheduled time limit, the task that has been pushed forward and one of the tasks in front of it have no float time. Otherwise the makespan of the schedule would become longer.

Since the task that has been pushed forward is always only restricted by one of the other two tasks (if task A is pushed forward either B or C can have float time), for one schedule there are two alternatives as to which task has float time (Figure 5-4). According to Lim et al. (2011), when considering every possible combination of the task sequences it is clear that since all of the tasks has at least in one combination float time, none of them is critical. However, as discussed above, two of them are always part of the critical chain. Therefore, *every single schedule should be treated as a stand-alone result of the project* and the results of diverging schedules with the same makespan and resource configuration should not be combined with each other. This can lead to loose assumptions about actually critical tasks and so the project might only be completed with delay.

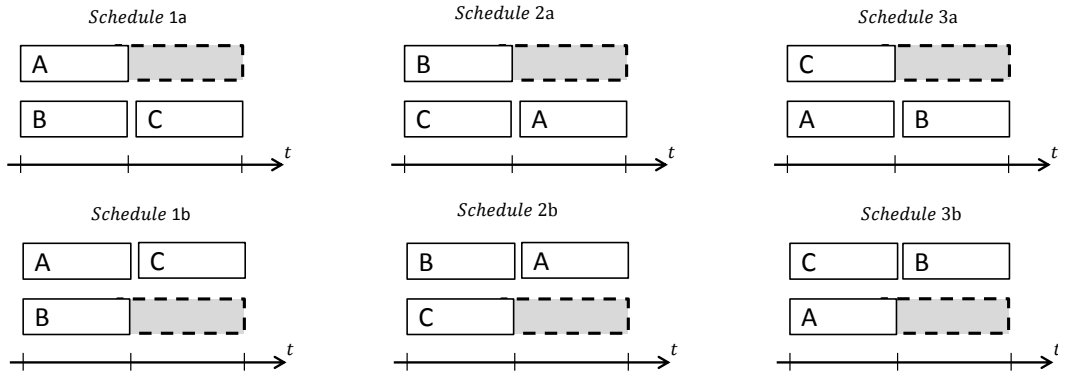


Figure 5-4: Solution combinations for the introduced example (dashed gray box: total float time)

There might exist diverging schedules with the same makespan, but their resource utilization and the execution order of the tasks are different and so the two schedules cannot be converted from one to another without violating one of the constraints. How the execution order of the tasks influences the results of float time and why similar schedules with the same makespan cannot be used for float time detection will be discussed in Section 5.3.

5.3 Concept of float time determination

One of the goals of the author's research was to calculate the *total float time for individual tasks* and to determine the *critical chain* of tasks considering *resource constraints* and *limited resource availability* in a construction project schedule generated using *constraint-based discrete event simulation*. This simulation technique is favored due to the lack of other flexible constraint

satisfaction modeling approaches and because it is a simple way of generating feasible schedules that satisfy both precedence and resource constraints. In addition, in the developed approach the goal was to calculate total float time in a single calculation step without using heuristic or iterative algorithms and the discrete event simulation is able to provide the required results.

In order to calculate total float, similar to the backward-pass analysis used in CPM, a newly developed backward simulation method along with a coupling process extending the common forward discrete event simulation is introduced. Using the forward simulation, the earliest schedule date of all tasks can be determined with the predefined resource and execution order configurations. Accordingly, inverting the direction of the simulation will result in the latest schedule date for all the tasks considering resource availabilities. So, in principle, when the makespan of the schedules and the tasks' execution sequence determined by the forward and the backward simulation is identical, the difference between the resulting schedule date of the backward and forward simulation for the single tasks represents its total float time while taking into consideration resource availability. When the total float of a task is zero, it is a part of the critical chain. The determination of the float time[13] for the individual tasks consists of three main steps:

1. Determine schedule with forward simulation;
2. Determine schedule with backward simulation;
3. Compare the results of the forward and backward simulation and determine float time.

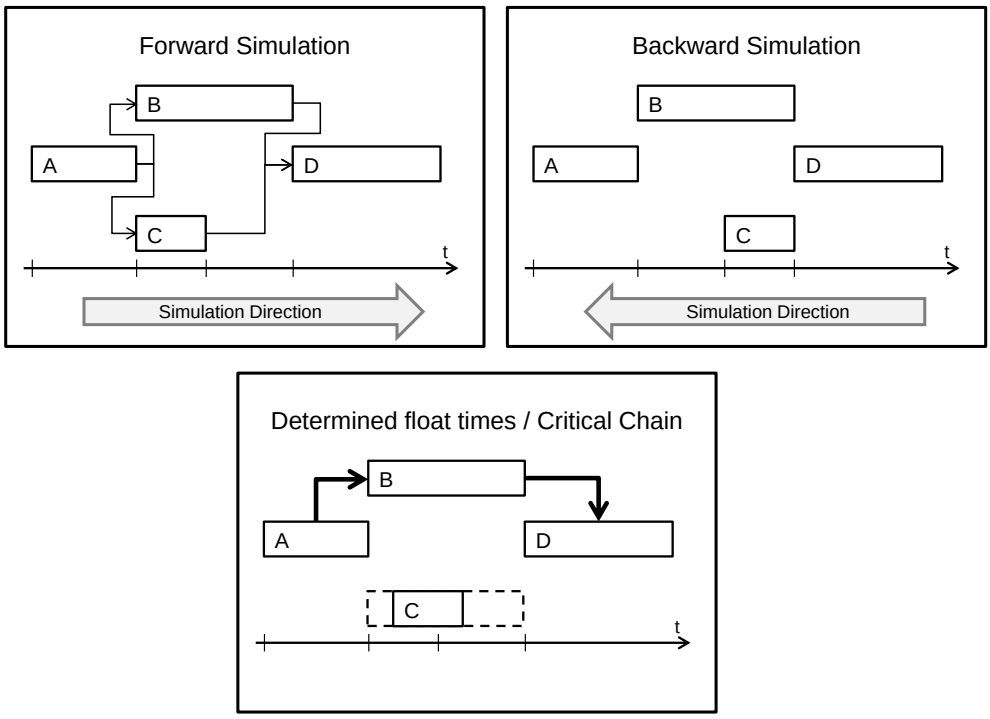


Figure 5-5: Concept of determining total float and critical chain by combining forward (left) and backward simulation (right). (Critical chain: large arrows; float: dashed box)

The principle of simulation-based total float calculation is presented in Figure 5-5. Four tasks are to be simulated: A, B, C and D. B and C must follow A, and C must follow B and C. The execution time for each task is different. When simulating these tasks using forward simulation, B and C will

[13] While mentioning float time in the thesis it always refers to the total float of a task.

start right after A has been finished, and D will start after B has been finished (C finished earlier). If we simulate these same tasks backwards, B and C will be performed right after D has been completed and A will be performed after B has been completed (C finished earlier) due to the reversed direction of the simulation. The two results can be combined with one another because the complete execution time and the order of the tasks are the same. Task C has a float time between the finish of task A and the start of task D. The critical chain of the schedule is therefore A→B→D because they have zero float time.

To determine total float for individual tasks, as in the simple example above, it is important that the *order of the task execution sequence for the forward and backward simulations is identical*. When the makespan of the determined schedules with the forward and backward simulations are the same, however, the sequence of the tasks is not identical. In that situation some tasks may start earlier in the schedule generated by the backward simulation (latest start time) than in the schedule generated by the forward simulation (earliest start time), thus the float time cannot be determined.

In order to determine total float it is crucial that *every single task must start later in the schedule generated by the backward simulation than in the schedule generated by the forward simulation*. To achieve this, a number of mechanisms have been developed and will be discussed in the following sections.

In order to take resource limitations into account not only for the scheduling process, but also for float time calculation, the *resource limitations will also be considered during float time calculation*. Therefore, if the resource limits within a float time range of several tasks will be exceeded, the float time of a task that has been executed earlier with the forward simulation will be shortened. In the example shown in Figure 5-6, task C has been executed before task D in the schedule generated by the forward simulation so the total float of C will be shortened by the execution length of task D (compare results in Figure 5-6 float time without resource limits). Thus the resource limits will not be exceeded when the execution of task C starts only at its latest start time.

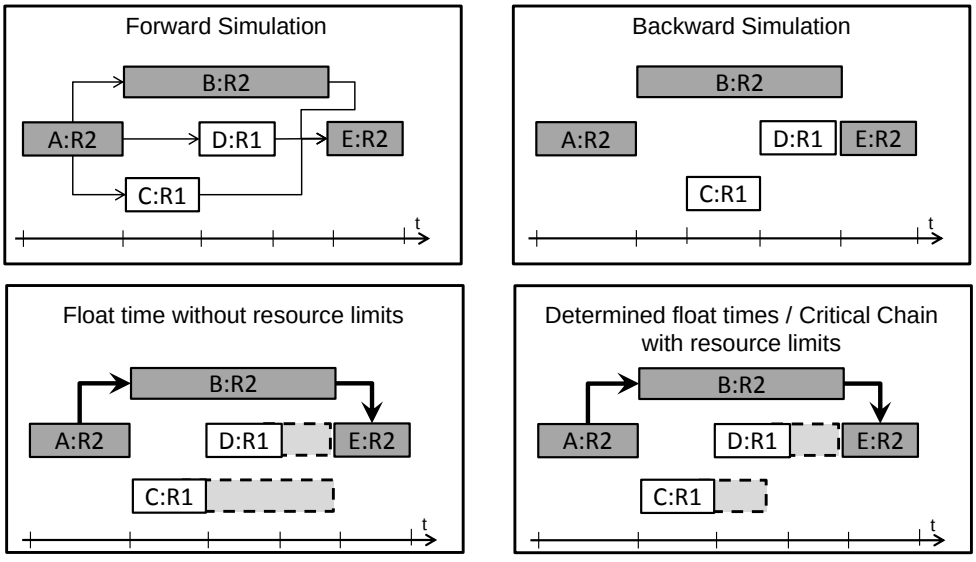


Figure 5-6: Determination of float time taking resource limitations into account (tasks: A, B, C, D, E; resources: R1 (white), R2 (dark gray); precedence relationships: A→B, A→C, A→D, B→E, C→E, D→E (thin arrows); float time: dashed light gray boxes; critical chain: large arrows)

To implement this behavior and obtain reliable results from the connected simulations, conceptual changes and restrictions have to be made in the existing simulation model. The necessary conceptual extensions of the simulations will be introduced in next sections.

5.3.1 Backward simulation

The backward simulation is used to establish the latest execution date of the tasks that can be used to determine total float for each individual task within the project. The concept of backward simulation is basically the same as that of forward simulation (see Section 4.2.2) – the simulation actually runs forward in time but with *reversed execution conditions*. Using this approach, the resulting schedule is calculated based on a simulation that starts from a virtual completion date of the construction project and runs backwards in time until the starting point of the construction process is reached. The reversed conditions mean an inversion of the precedence constraints. For example, for forward simulation *Pour Concrete* is executed after *Construct Formwork*; for backward simulation *Construct Formwork* follows *Pour Concrete* (Figure 5-7). This entails reordering the priorities of the tasks and reversing the resource calendar. To realize this inversion, a function is written which reverses the order of the precedence constraint at the beginning of the simulation, thereby changing the successor into the predecessor and the predecessor into the successor.

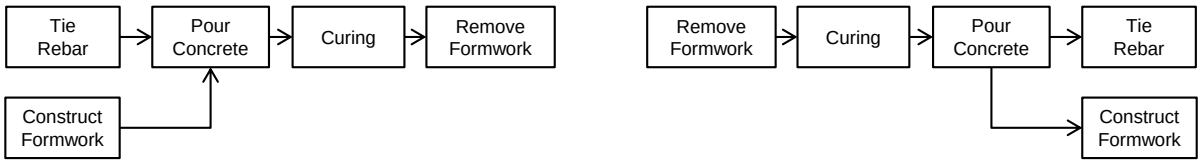


Figure 5-7 Precedence relationships for forward simulation (left) and for backward simulation (right)

To test the applicability of the backward simulation approach, a simple test case was utilized with unlimited resources (see Appendix). This configuration meant that the simulations are independent of resource constraints and task priorities, and the total duration of the project depends only on the precedence constraints between the tasks. The calculated results are identical to the results determined by the CPM/PDM. Therefore the order of the executed tasks of the backward and forward simulation is identical. Furthermore, since the results are identical to the schedule determined by CPM/PDM, the generated execution dates of a task represent its earliest and latest execution dates. Thus the difference between the scheduled dates of a task in the forward simulation versus the backward simulation represents the task's total float. The simulations with unlimited resources essentially reproduce the results of the CPM/PDM.

This served as the verification and also validation of the basic backward simulation concept. However, to determine float time for tasks within a project with restricted resources further restrictions must be applied, which will be introduced in the next section.

5.4 Calculation of total float time

It has been shown that the introduced forward and backward simulation concepts works well for projects with unlimited amount of resources. However, when the amount of available resources is restricted the concept of total float determination becomes more complex.

Executing the backward simulation with restricted resource availability and comparing the results to the results of the forward simulation, as discussed at the beginning of this chapter, may result in a discrepancy. The order of the executed tasks is not identical to the result of the forward simulation so the float time and the critical chain of tasks cannot be determined. This problem occurs because the task execution process of the simulation is based on the order of the task list and, since it is randomly generated, the task execution order is different in the forward simulation when compared to the backward simulation. Hence, the total float time of the tasks cannot be determined.

A first approach to solving this issue might lie in the use of priorities assigned to the tasks. Thus the simulation uses a priority-ordered list instead of the FCFS queue to determine a feasible schedule. However, this list of priorities is not used to implement soft-constraint (Section 4.2.2), but rather to link the forward and the backward simulation with the goal of achieving the same order in the task execution. However, only reversing the tasks' priorities can induce a change into the execution sequence of the tasks (Figure 5-8) and thus the total float still cannot be determined. To illustrate the issue a simple example consisting of four tasks is presented (Figure 5-8). One of the tasks uses a different resource than the other three and there is only one resource unit pro resource category available for the project. In this example with the forward simulation the tasks C and D have been executed before task B. But because of their higher priority for the backward simulation the order of the tasks has been changed and thus no feasible float time can be determined.

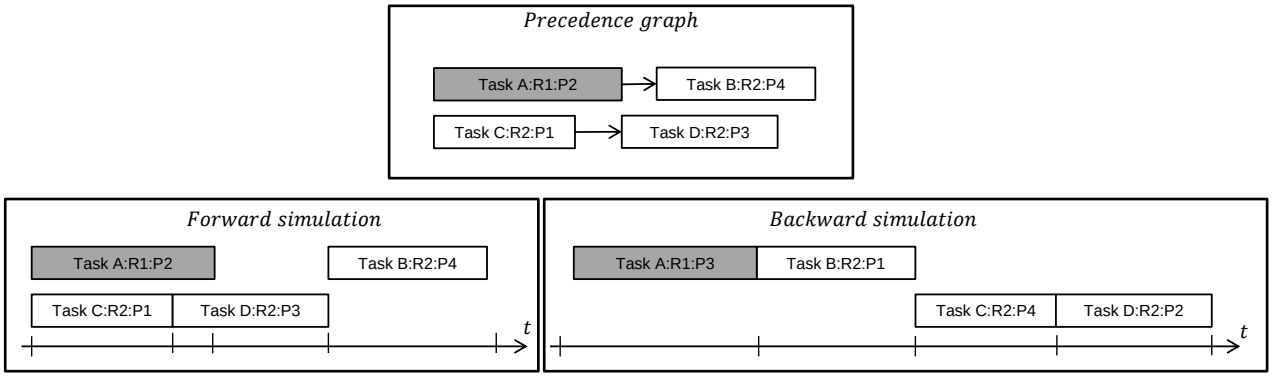


Figure 5-8: Issue with reversed priority settings for the backward simulation: change in the execution order of the tasks (R: resource class, P: priority)

Therefore, to obtain an *identical order of tasks for the backward simulation and the forward simulation*, during the forward simulation each of the executed tasks is assigned a new priority.

Since the end of a task represents the beginning of the task for the backward simulation, the new priorities will be set according to the *finish date of the task's execution* (the later the finish date is, the higher the priority) after the forward simulation has finished (priorities for the forward simulation are assigned randomly). The backward simulation then uses these priorities to replace the FCFS list with a priority-ordered list of executable tasks for every time step when tasks are competing for resources, and then tries to collect the necessary resources for the one with the highest priority. If that is not possible, the next task with the second highest priority is chosen and so on. We call this the *priority-*

by-end-date method. By using this method, the order of the executed tasks is identical for both the forward simulation and the backward simulation (Figure 5-9).

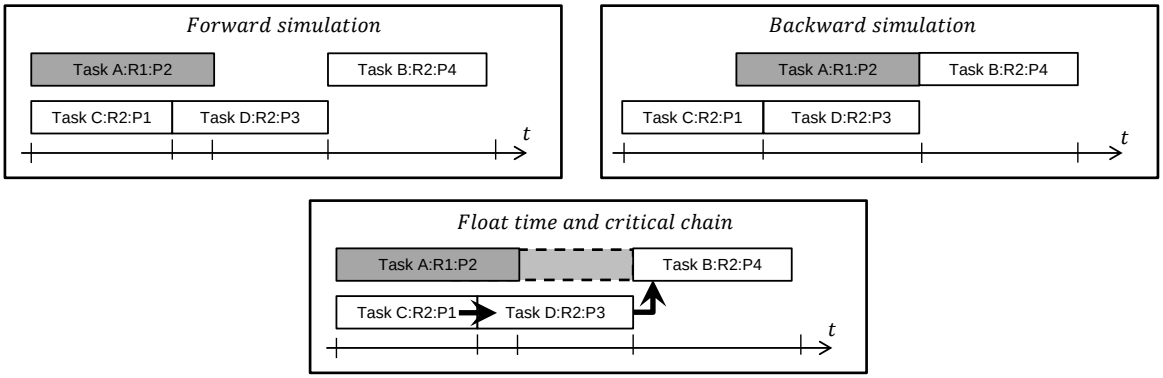


Figure 5-9: Applying the new priorities based on the priority-by-end-date method for the backward simulation. The new priorities are set according to the finish date of the task in the forward simulation. The result is an identical execution sequence of the tasks with the backward simulation, therefore total float and the critical chains are determinable. (Float: dasher gray box, critical chain: large arrows)

However, to extend the application of the priority-by-end-date method to larger and more complex cases than the introduced example with 4 tasks, further mechanisms must be introduced to keep the sequence of the tasks identical within the schedule generated by forward and backward simulation. To understand the developed mechanism, a new term, the *independent task*, i.e. tasks that are independent from each other, will be introduced. The position of these tasks within the execution sequence using only the priority-by-end-date method for the backward simulation might change pushing other tasks forward in time thereby making it impossible to determine feasible total float. To avoid this change within the execution sequence a mechanism has been developed that will be introduced after a further discussion of the role of independent tasks.

As introduced in Section 4.3 the constraint-based discrete event simulation requires a precedence graph to start the simulation. *The independent tasks use resources from the same resource class and are lying on different paths of the precedence graph so that a direct path between the two tasks cannot be found. Thus these are the tasks that might be executed simultaneously with each other within a schedule.* Such tasks will play an important role both at total float detection and later at the optimization of the schedules (introduced in Chapter 7).

The determination of those independent tasks from the precedence graph that is a directed acyclic graph is realized by means of the transitive closure of the graph, which describes whether or not a path exists between two tasks. In a directed acyclic graph when, e.g. task B depends on task A, it means that a path from A to B exists within the graph. Although the dependency is a symmetric characteristic (if task A depends on task B, then task B must also depend on task A) due to the directed precedence graph, the reversed dependency is not part of the transitive closure. To include this reversed dependency within the matrix the transitive closure must be copied through mirroring over the diagonal of the matrix thereby creating a dependency from task B to task A as well. However, it is important to mention that this dependency does not mean a directed precedence constraint between A and B. This only confirms that there is a path existing between A and B. The dependency matrix can be used to determine separated paths between the start and end task of a project or between two tasks (Figure 5-10).

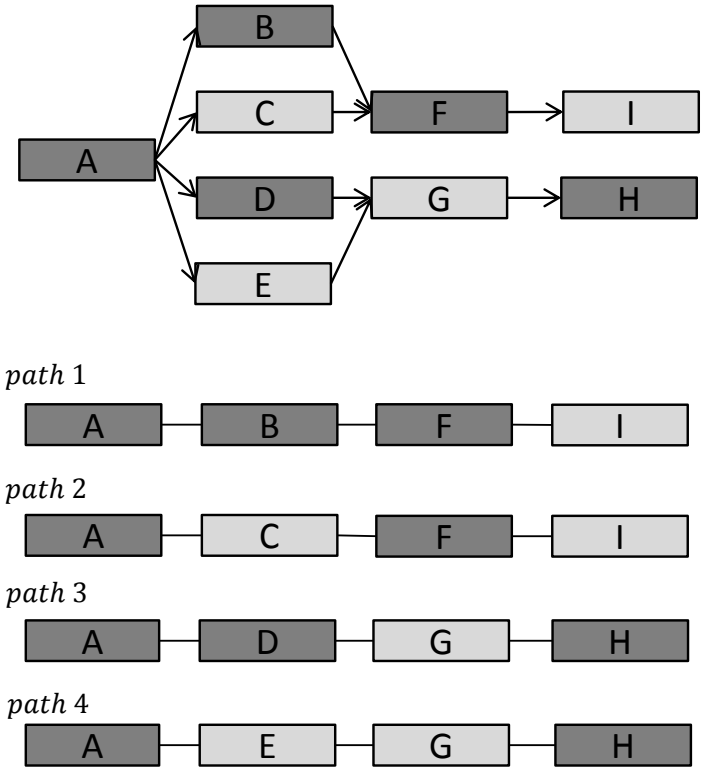


Figure 5-10: Separated paths between the start and the end point of the represented precedence graph in the top. (Different colors: different resource needs)

Thus, the dependency matrix will always be symmetric (if A can be reached from B, then B can be reached from A as well). *The dependency matrix is a symmetric matrix that describes whether a path between two tasks exists or not. A row or column represents all the dependencies of a task. When a path exists between the two concerned tasks the matrix value is set to 1. Otherwise, it is set to 0. Since an examined task is always dependent on itself, the elements of the main diagonal of the dependency matrix are always set to 1.*

Furthermore, according to the definition of the independent tasks, it is important that these tasks use the same resources for their execution. This means that the dependency matrix must be fragmented according to the available resource classes. The tasks and their dependencies that have the same resource needs are selected from the dependency matrix and collected in the *fragmented dependency graph*. When a task needs different resource classes for its execution it will be a part of additional fragmented dependency matrixes. *When an element of the fragmented dependency matrix is 0, the corresponding tasks are a pair of independent tasks.*

To determine the fragmented dependency matrices for a more complex example (Figure 5-10), first the adjacency matrix of the complete project has to be determined (Figure 5-11). This matrix describes the neighborhood relationships of two tasks. If an entry of the matrix is 1, then a precedence relationship exists between them. Using the adjacency matrix, the transitive closure of the project will be determined (Figure 5-11). Finally, the transitive closure will be mirrored and fragmented according to the resource classes (dark grey and light gray) as described in the last section, so that each resource class has its own fragmented dependency matrix with all the tasks that the examined resource require (Figure 5-12). Figure 5-13 represents the different possible paths according to the fragmented dependency matrices of the different resource classes between the start and the end of the introduced precedence graph. The length of a path can be determined by summing up the duration of every task that the path contains.

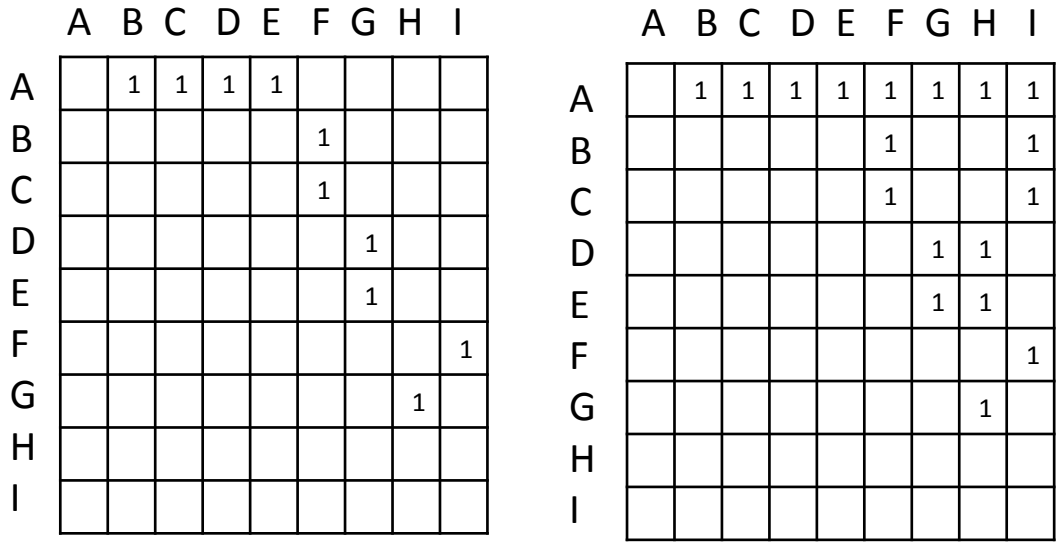


Figure 5-11: Adjacency matrix (left) and the transitive closure (right) of the introduced precedence graph (Figure 5-10)

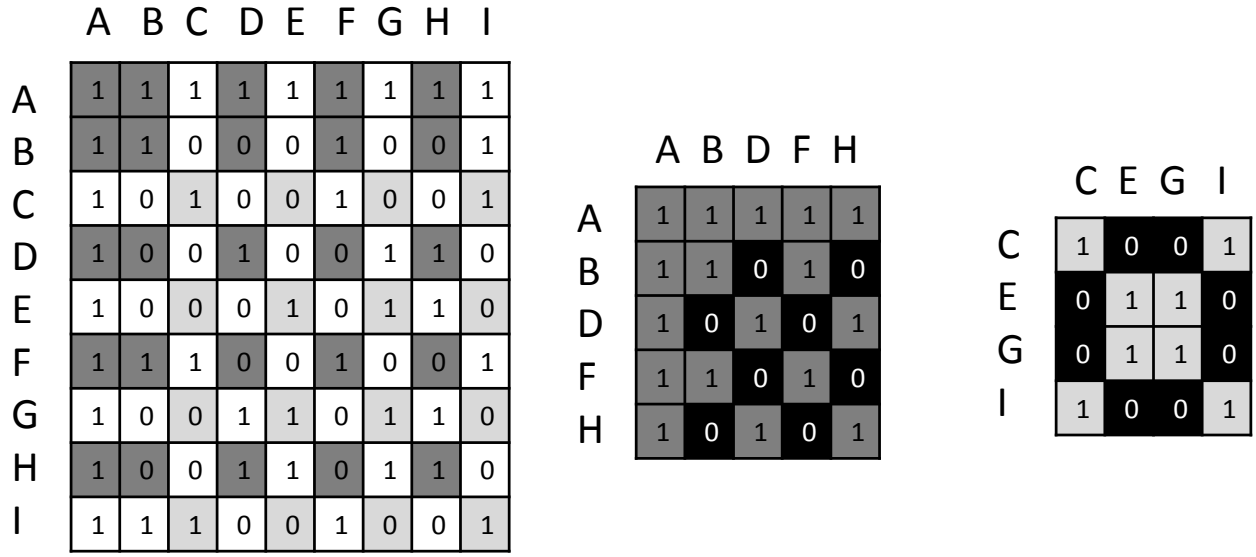


Figure 5-12: The dependency matrix and the fragmented dependency matrices for the different resource classes (dark and light gray). Black elements with white text: independent pair of tasks

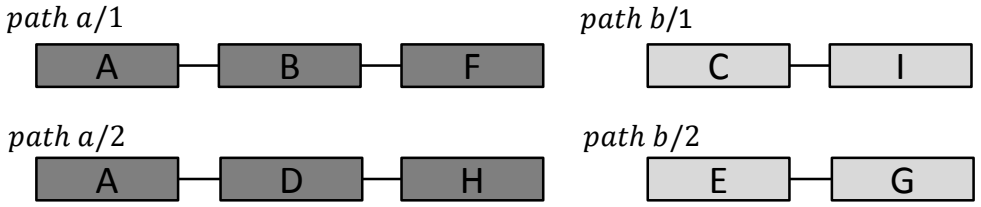


Figure 5-13: Different possible paths between the start and the end of the introduced precedence graph (Figure 5-10) according to the fragmented dependency graph

However, the priority-by-end-date method is an essential element of the float time determination, in complex cases additional mechanisms must be applied to keep the execution order of the tasks identical for the backward and forward simulation in order to be able to determine feasible total float for the individual tasks while also taking resource constraints into account.

As presented in Figure 5-14, in the forward simulation A, B, D, F and E build a path. Task C is executed between task A and task F on a path parallel to the path consisted of task B, task D and task E. C and D use the same resources and in the forward simulation C is executed before D.

Since the backward simulation uses only the priority-by-end-date method, although C was executed before D in the forward simulation it can be started right after E because D still has a successor (F) which has to be executed first. So the execution order of C and D is swapped in the backward simulation. We call this the *unintended swap phenomenon*. Such a swap in the execution order is not allowed by float time determination since it would mean that there is a time period when task C and D can be executed simultaneously. Thus, during this time period the resource limits would be exceeded. Therefore the relative order of task C and D must be kept the same and the total float of task C must be shortened (Figure 5-14). It is important to mention that task C could also be executed after task D, but this would mean another schedule with a different sequence order of the two tasks than the sequence order set forth above.

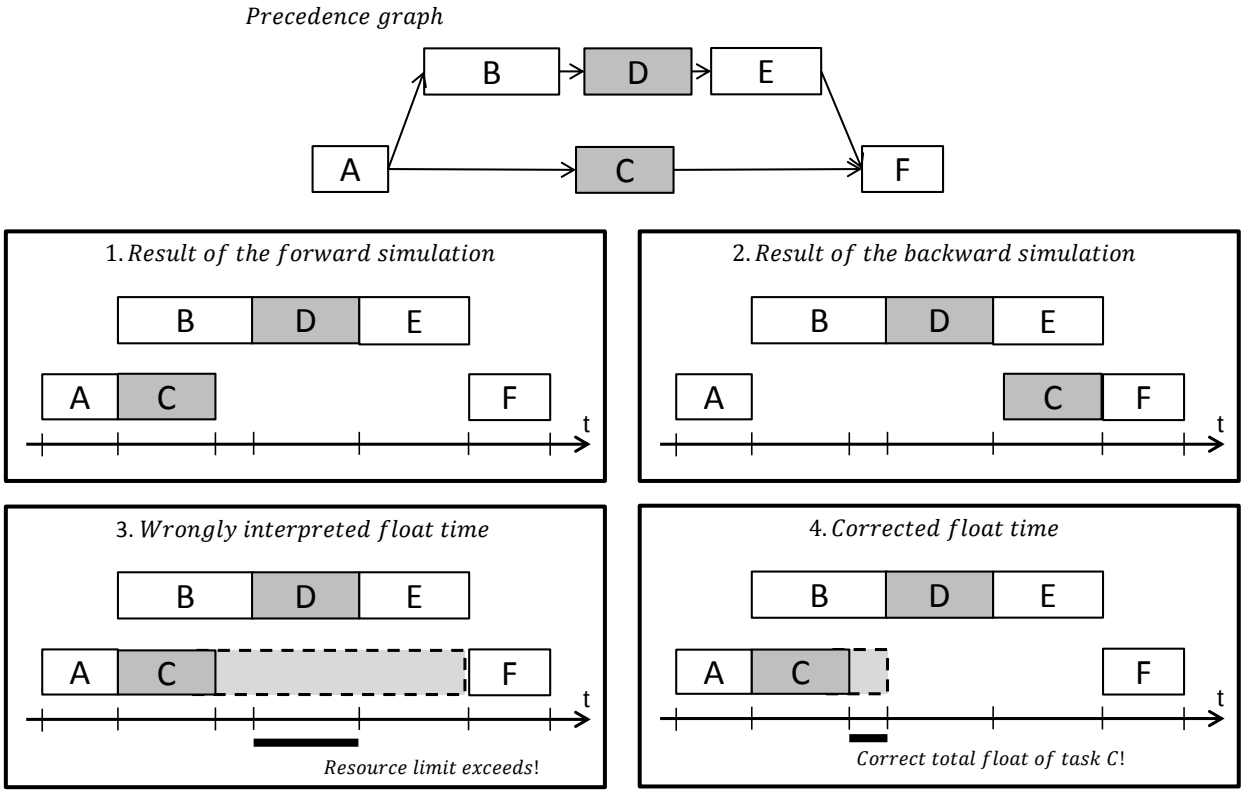


Figure 5-14: Results of the forward (1) and backward (2) simulations. The problem (3): exceeding the resource limits at the time period of float time. The solution (4): maintain the relative order of C and D → shorter total float time for C. (different colors: different resource needs))

An additional extension is therefore necessary to prevent this task order swap in the backward simulation and to be able to calculate the feasible total float of the tasks. A favored solution would be to apply *additional precedence constraints* similar to the resource-activity combined precedence relationships introduced by Lu and Li (2003) between the tasks using the *same resources* based on

the result of the forward simulation. This means that the order of the task execution paths for the different resource types will be kept the same for the backward simulation as it was for the forward simulation. We call this restriction *sequence enforcement constraint*.

If the available amount of resources permits more tasks to be executed in parallel, the problem becomes more challenging and the following rules have to be considered when the additional sequence enforcement constraints are generated:

Create precedence constraints between an examined task and the successors of every independent task that finishes later or at the same date in the forward schedule as the examined task where the sum of the necessary resources of all the independent tasks between the task and the examined task exceeds the resource limit.

The following steps are necessary to determine the sequence enforcement constraints:

1. Determine a schedule using the forward simulation;
2. Identify the independent tasks;
3. For every task in the schedule;
   - 3.1. *Select* first finishing *task* in the schedule;
   - 3.2. *"sum of necessary resources" = "necessary number of resources of the selected task";*
   - 3.3. Store the tasks that are independent from the *selected task* in the *list of independent tasks;*
   - 3.4. Delete every task from the list of independent tasks that terminates earlier than the finish date of the *selected task;*
   - 3.5. While "sum of necessary number of resources" ≤ "available number of resources";
     - 3.5.1. *Select next finishing independent task* from the *list of independent tasks;*
     - 3.5.2. "Sum of necessary number of resources" == "sum of necessary number of resources + necessary number of resources of the *selected independent task*";
   - 3.6. Create a sequence enforcement constraint between the *selected task* and the *successors of the selected independent task*;
   - 3.7. Delete every task from the *list of independent tasks* that are not independent from the *selected independent task*;
   - 3.8. Delete the *selected independent task* from the *list of independent tasks*;
   - 3.9. While the *list of independent tasks* is not empty;
     - 3.9.1. Create additional precedence constraint between the selected task and the successors of the next finishing independent task in the list of independent tasks;
     - 3.9.2. Delete every task from the *list of independent tasks* that are not independent from the *selected independent task*. Delete the *selected independent task* from the *list of independent tasks*.

To illustrate the meaning of this complex rule let us consider an example of nine tasks. Their precedence graph and resource needs are depicted in Figure 5-15. The priority of a task corresponds with its name. For the simulations there are two units of every resource available; hence the given configuration will result in the schedule presented in Figure 5-15 when conducting the forward simulation.

Based on the determination algorithm introduced in the previous sections, the independent tasks within the generated schedule are the following: Task 1 is independent from Tasks 3, 4, 7 and 8; Task 3 is independent form Tasks 1, 4, 5 and 8; Task 4 is independent from Tasks 1, 3, 5, and 7; Task 5 is independent from Tasks 3, 4, 7 and 8; Task 7 is independent from Tasks 1, 4, 5 and 8 and Task 8 is independent from Tasks 1, 3, 5 and 7. Every other combination of tasks is dependent on each other,

so they cannot be executed simultaneously and cannot be considered for sequence enforcement constraints determination.

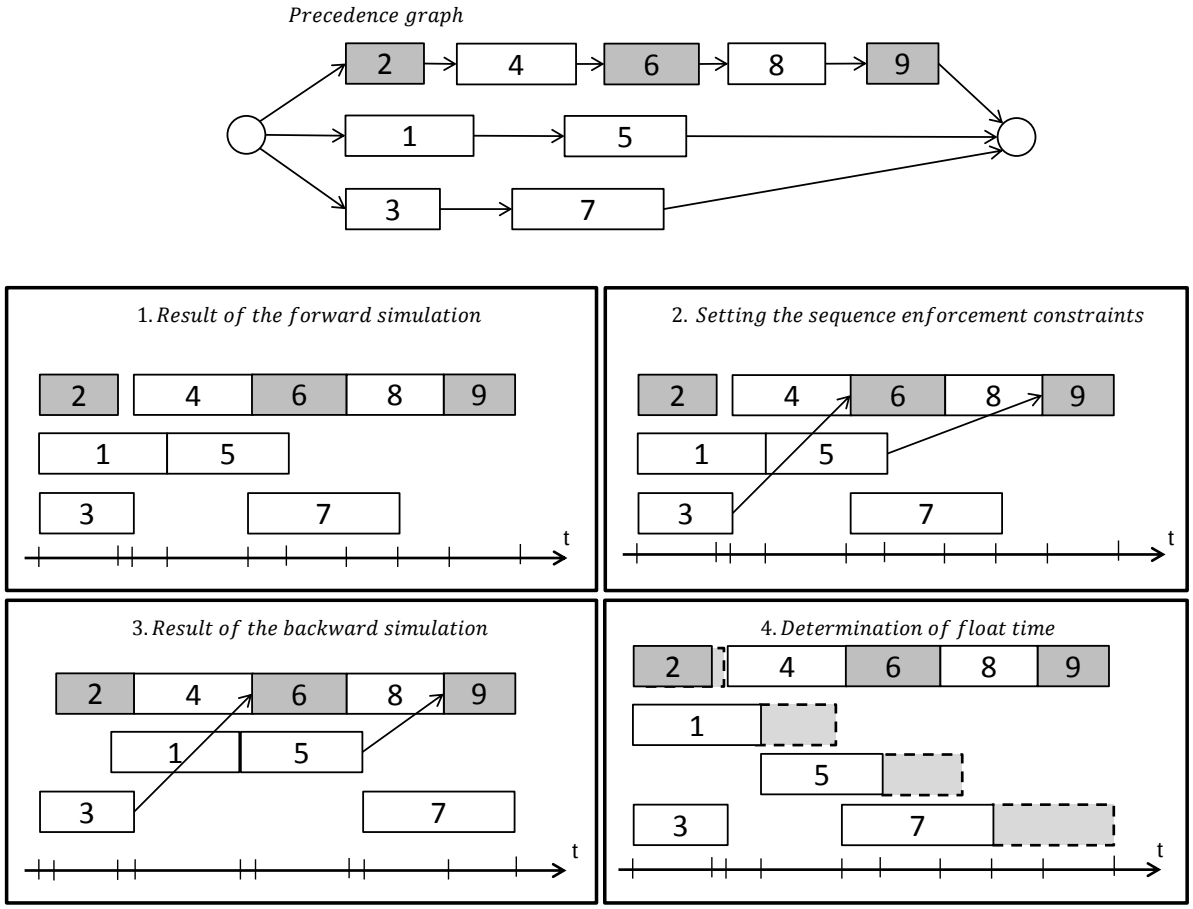


Figure 5-15: Determination of the sequence enforcement constraints for the backward simulation after the forward simulation. (Different colors: different resource needs, arrows: new constraints, dashed boxes: float time)

After identifying the independent tasks, the sequence enforcement constraints must be set for every task in the project. This process goes forward in the schedule and investigates every single task to determine whether a new constraint can be set for the *selected task* or not. First it selects the first task in the schedule: Task 2. Since there are no other tasks which are independent from it the process proceeds to the next finishing task: Task 3. Next, this method performs the same process forward in time through all the associated independent tasks according to their finish date. For example, for Task 3 it selects Task 1 first. Task 1 and Task 3 both require one unit of resource. The sum of the resource units required totals two units, which is equal to, but does not exceed, the limit of two resources. Therefore it advances one step further and selects the next finishing independent task in the schedule: Task 4.

To determine how many resource are used up to this point, the program adds the required amount of resources of Task 4 (1) to the prior total (2). This is 3, which exceeds the resource limit (2) and therefore a sequence enforcement constraint will be set to the successor of Task 4, which is Task 6. It is important to mention that *for the calculation of the sum of the used resources only one task per path will be taken into account, since additional tasks cannot be executed simultaneously*. Furthermore, when a constraint is set to one task, the further independent tasks on the same path will

be ignored as the relative order is fixed in this case. The sequence enforcement constraints for a selected task will be set as long as the *list of independent tasks* is not empty. The only task left in the list of independent tasks is Task 5 (Task 1 has already been taken into account, Task 4 has been selected as the one to connect with, thus Task 8 has also been ignored). Since it has no successor, no additional constraint will be set.

The introduced constraint setting method proceeds to the next finishing task in the schedule generated by the forward simulation, which is Task 1. Since the method is no longer at the beginning of the schedule, the independent tasks that were already completed before Task 1 will not be considered for further constraint determination. Hence, the possible independent tasks for Task 1 are: Task 4, 7 and 8. Following the same procedure as for Task 3, a sequence enforcement constraint can be set to Task 9 (the successor of Task 8). In this case the sum of needed resources were built from the resource needs of Task 1, 7 and 8, which exceed the resource limit. Since every other independent task has been either considered or has no successor, the method proceeds to the next finishing task.

This process is continued until all the tasks have been considered for creating sequence enforcement constraints. The sequence enforcement constraints of the example are also represented in Figure 5-15. The corresponding results of the backwards simulation and the determined float times for the individual tasks are also represented in Figure 5-15. Because of the sequence enforcement constraints in backward simulation Task 5 did not swap positions with Task 8. Therefore the execution sequence of tasks is the same as in the forward simulation and no resource limits are violated.

Using the sequence enforcement constraints, the execution chain of the tasks is identical to that of the forward simulation. The backward simulation follows the task order of the forward simulated schedule which results in all the tasks starting later or at the same time as they do in the forward simulation. Therefore, a feasible total float for the individual tasks can be determined. In order to calculate total float for each individual task, the results of the forward and backward simulation are compared to one another. If a task in the backward simulation starts later than in the forward simulation, the task has a float time. The total float for each task is the difference between the two results. If the results are the same, the task has no float and is therefore part of the critical chain.

5.5 Limitations of the introduced float time determination method

In applying the introduced priority-by-end-date method and the sequence enforcement constraints for the backward simulation, the result will always be feasible float time amounts for the individual tasks within the examined schedule. When the number of available resources only allows the execution of one task at a time (such as the examples introduced in Figure 5-9 and Figure 5-4), the results of the float time determination are correct. The determined total float is the maximum amount of time a task could be pushed forward in time. The sequence of tasks is straightforward and the limit of the total float is either the start time of the next task using the same resource or the successor of this task. In Figure 5-15 an example was introduced where more of the tasks could be executed simultaneously. In this case, the total float time results were also correct. Extending this example by additional tasks as depicted in Figure 5-16, the limitation of the introduced methods can be identified. In this case the length of the three independent paths' are greatly different so that $|\text{Path 1}| >> |\text{Path 2}| + |\text{Path 3}|$.

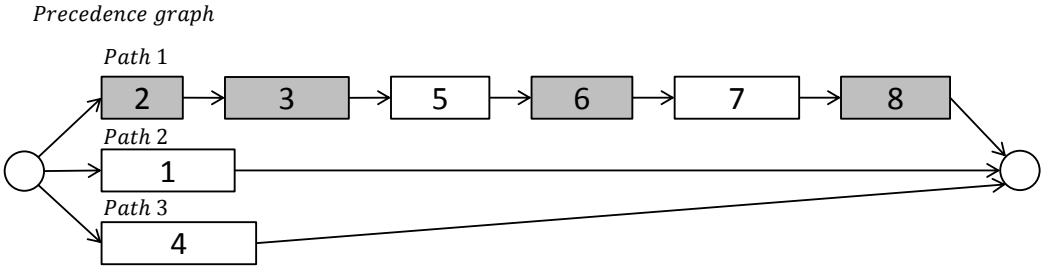


Figure 5-16: An example with large path length differences ($|Path\ 1| >> |Path\ 2| + |Path\ 3|$)

The results of the forward simulation, the applied sequence enforcement constraints and the results of the backward simulation are depicted in Figure 5-17. Due to the sequence enforcement constraints, Task 1 will have float time only until the beginning of Task 6, which is less than the expected amount of time (Figure 5-17). This is due to the introduced sequence enforcement constraints that are designed to always deliver a feasible result which does not exceed the resource limits, even during the time period of the float. Unfortunately, this assumes that tasks that have been executed simultaneously with the forward simulation will also be executed simultaneously in the backward simulation and will be surrounded by the same tasks. This cannot be guaranteed when there are *significant differences between the simultaneously executable paths' length* as in Figure 5-16. Particularly when *a task has a larger actual float than the duration of the simultaneously executed task* (|float of task 4| > |task 1|), a phenomenon called *serialization* of tasks can be observed. When such a situation exists where additional resources are available to execute more tasks at the same time, the execution order of the tasks can be reordered without exceeding the resource limit (Figure 5-18) all the while maintaining the same execution order for both the forward and backward simulation. Thus, by using the introduced determination technique, the determined float time amounts will be however feasible[14], but shorter than actually possible.

[14] Does not exceeds the resource limits during float time

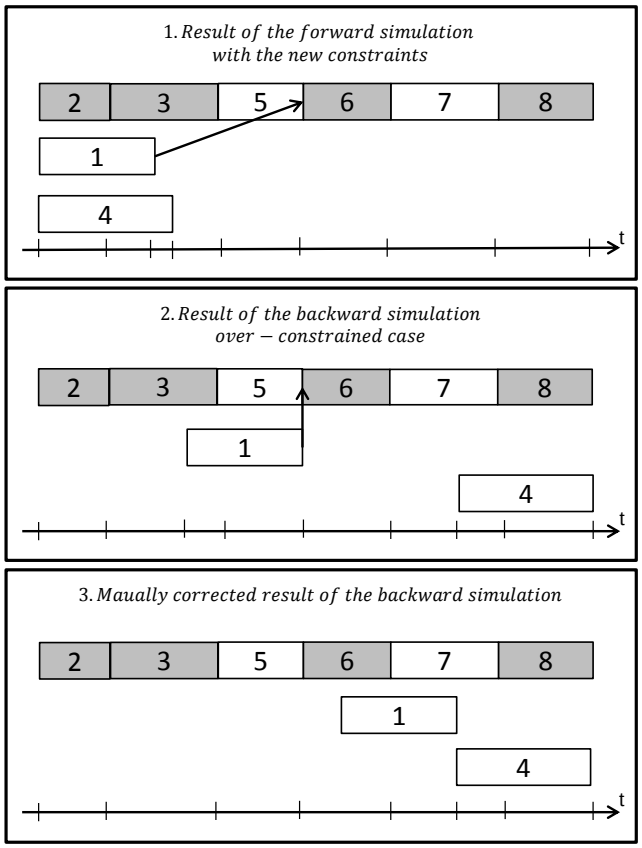


Figure 5-17: The simulation results from the example introduced in Figure 5-16. In the bottom are the manually corrected or expected results.

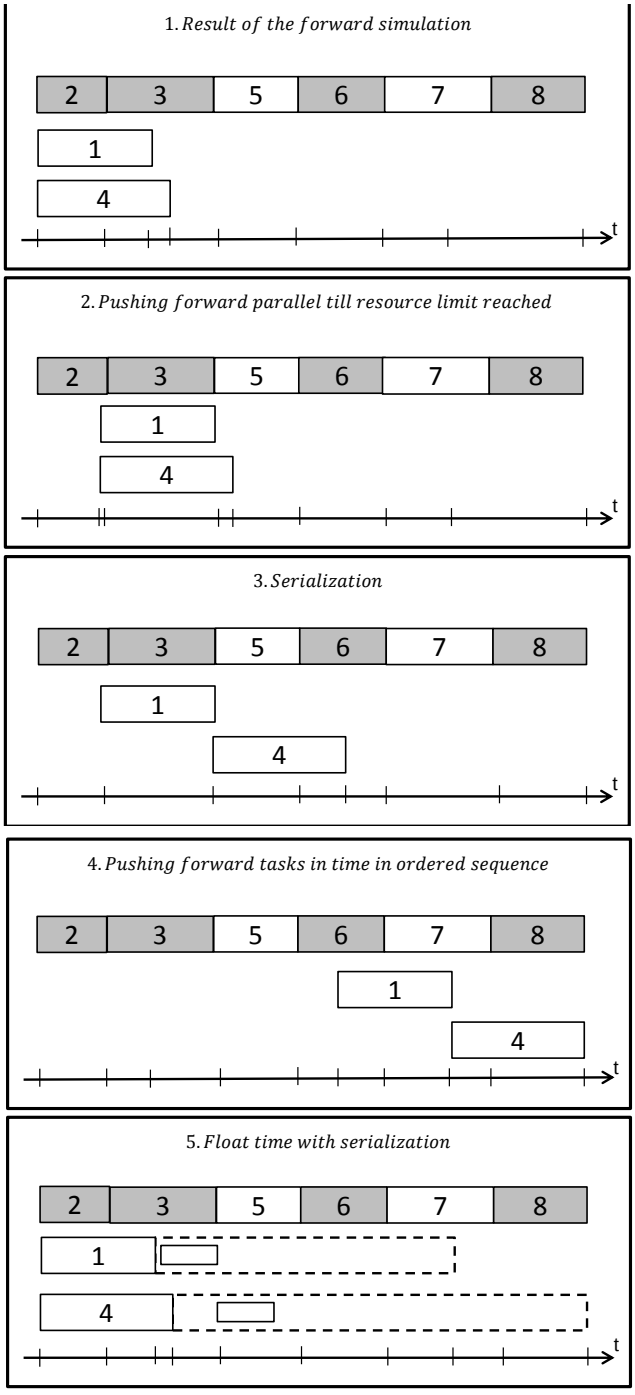


Figure 5-18: The effect of serialization for the determination of float time. (Dashed boxes: float time; boxes in float time: start point and order of serialization)

Further research is required to solve the problem as to how to loosen the restrictions of the sequence enforcement constraint setting method to keep the determined amount of float for every task still feasible but also correct in every case. A case study is introduced in the next section to test the applicability of the introduced float time determining methods on larger precedence graphs.

5.6 Case study

To test and validate the application of the introduced approach, a simplified representation of an already completed bridge construction project will be examined. The bridge is made of two abutments, a pier and two pre-cast girder elements. After the construction of the foundation and the head of the foundation for the pier have been completed, the abutments and the shaft of the pier must be constructed. This requires constructing the formwork, tying the rebar, pouring the concrete and removing the formwork (for simplicity's sake, the curing time for the concrete has been omitted). Before lifting the girders the corresponding abutment and the pier have to be poured with concrete, the racks for the bearings must also be poured with concrete and a scaffold has to be built next to the pier. After the girders are lifted, the concrete for the deck and the girder heads can be poured, and the parapets can also be created. When all of these tasks has been completed, a sealing system can be built for the bridge, and the scaffold for the pier can be removed.

There are three categories of resources used:

1. Formwork-works and others: "casing"
2. Assembly-works: "armouring"
3. Concrete pouring-work: "concreting"

The resources are calibrated so that two tasks from the same resource type can always be executed at a time. The precedence graph of the tasks is represented in Figure 5-19. The mapping between number and task can be found in Figure 5-21.

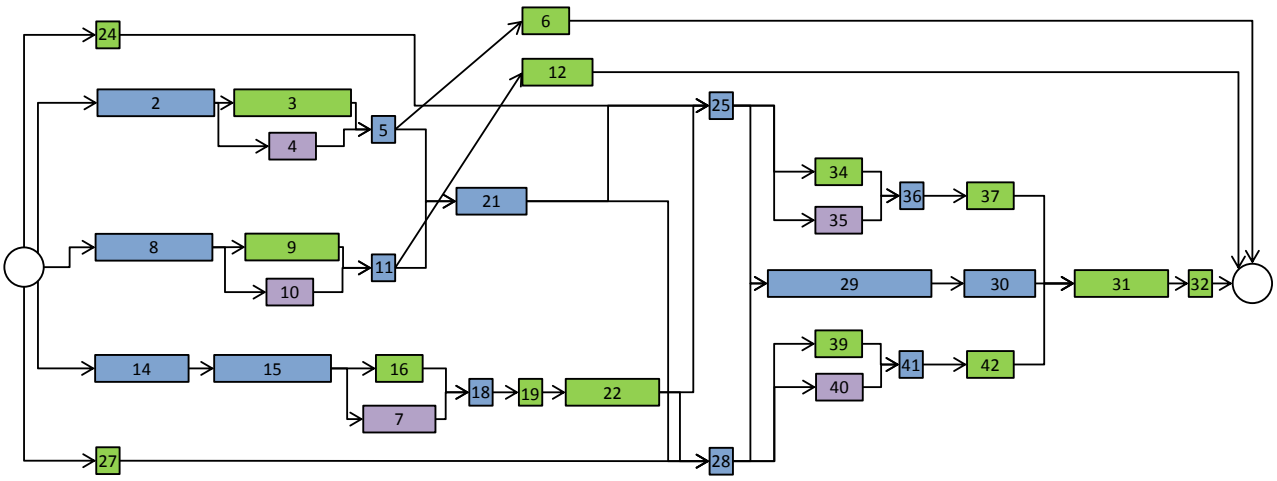


Figure 5-19: Precedence graph of the introduced test case. Colors: resources, such as blue: "concreting", purple: "armouring" and green: "casing".

In order to prevent the unintended swap phenomenon we apply the sequence enforcement constraints to the schedule produced by the forward simulation. The fragmented dependency matrices for the resources are depicted in Figure 5-20. The result of the float time determination without the sequence enforcement constraints is presented in the Appendix. The result of the backward simulation with the sequence enforcement constraints is also depicted in the Appendix.

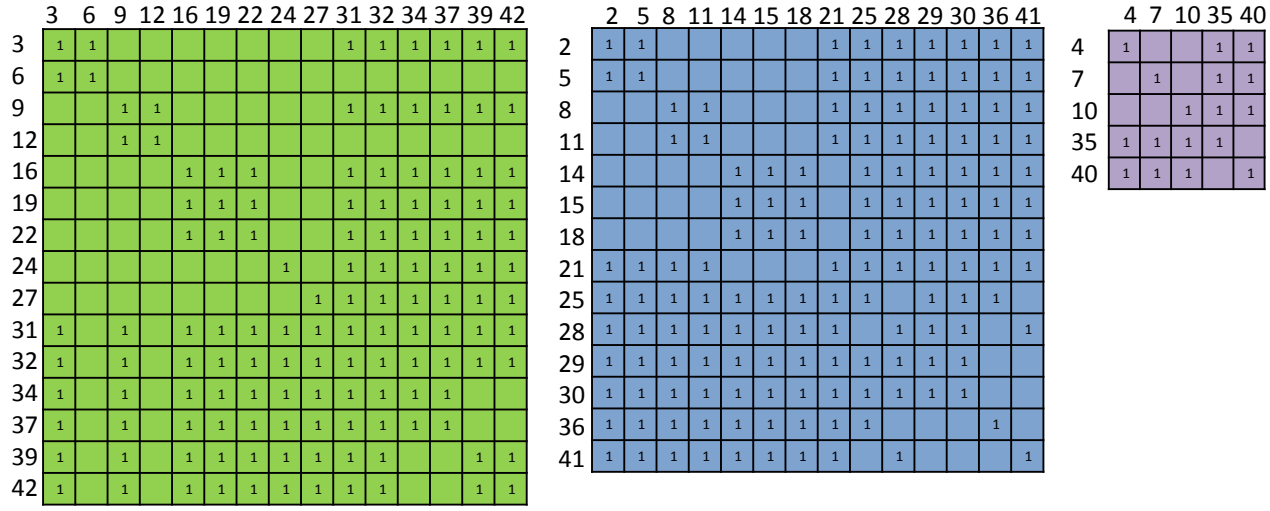
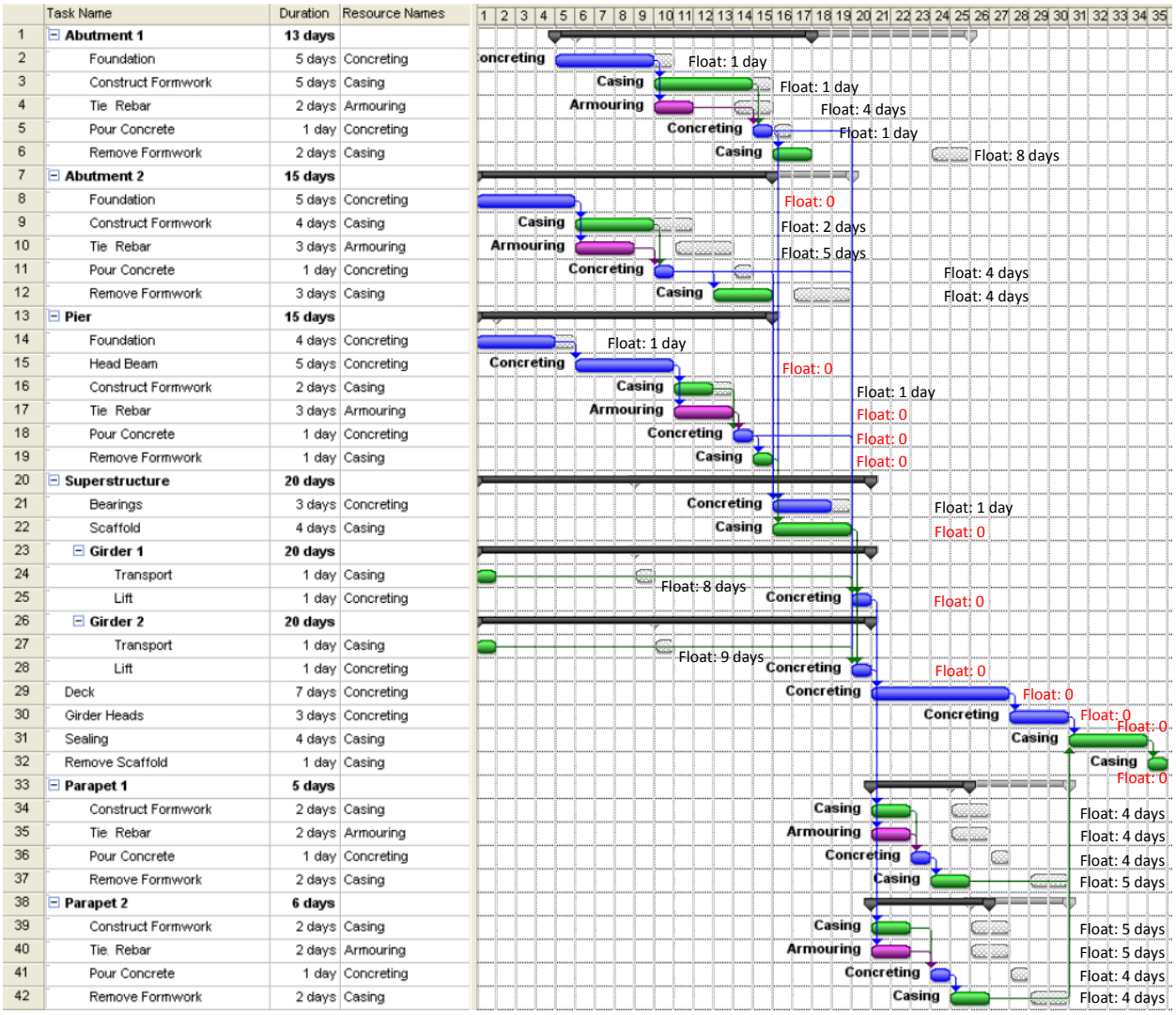


Figure 5-20: The fragmented dependency matrices of the resource classes used for the test case. 1: dependent tasks, empty cell: independent tasks, green: “casing”, blue: “concreting”, purple: “armouring”.

The results of the forward and backward simulation are presented in Figure 5-21. Since the makespan and the order of the tasks of the backward simulation is identical to that of the forward simulation, the difference between the scheduled dates of the backward simulation (latest date) and the forward simulation (earliest date) represent a feasible amount of total float for every individual task which does also not exceed the resource limits.

Furthermore, the critical chain of the tasks can also be identified. It is built up as follows (indicated with red text in the figure): Abutment 2: foundation → Pier 1: head beam → Pier 1: tie rebar→ Pier 1: pour concrete → Pier 1: remove formwork → Superstructure: scaffold → Girder 1: lift & Girder 2: lift → Deck → Girder heads → Sealing → Remove scaffold. The total duration of the project with the given resource configuration is 35 days. The calculated total float time for the individual tasks are indicated in Figure 5-21.

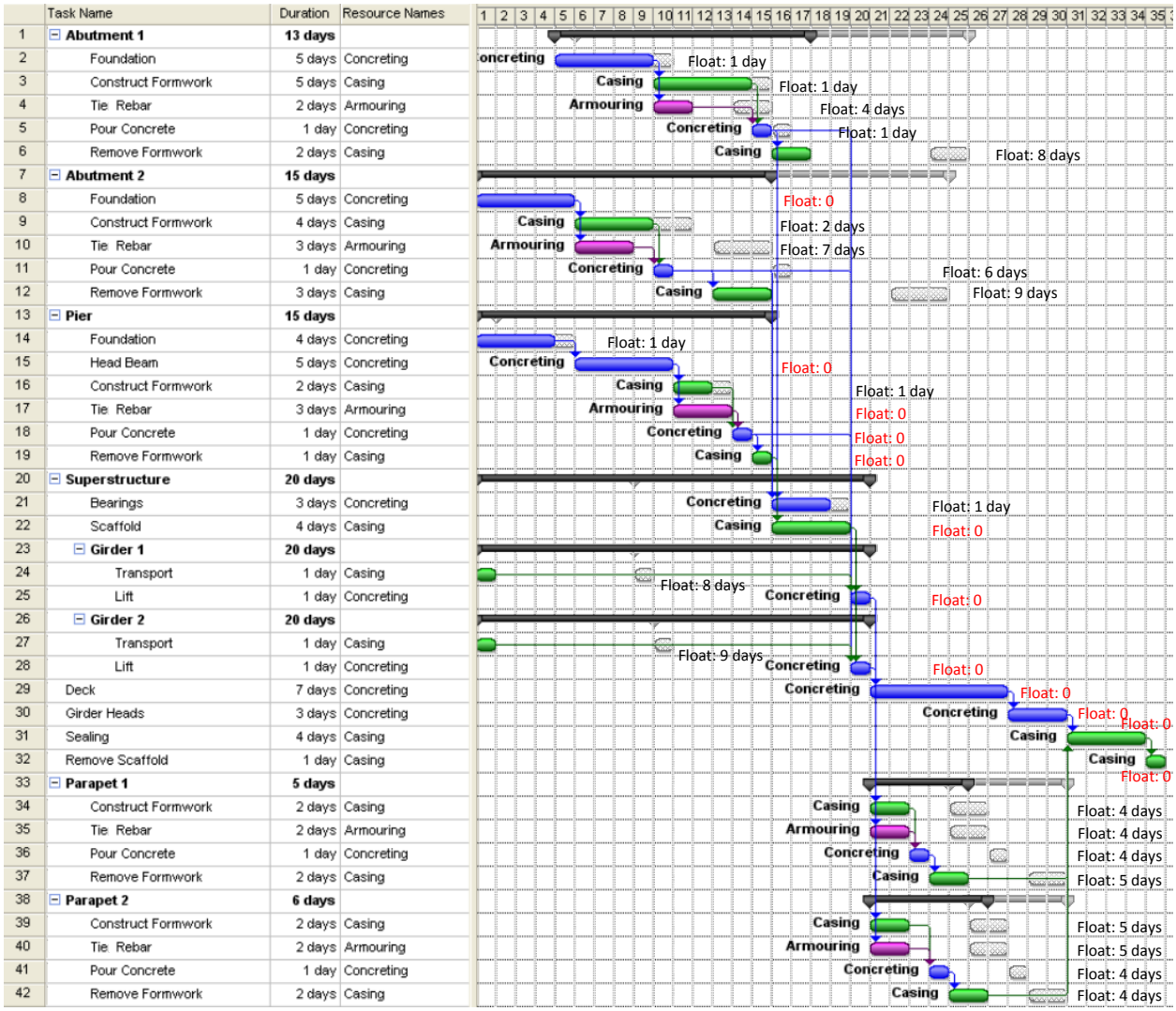


	Task Name	Duration	Resource Names
1	**Abutment 1**	**13 days**	
2	Foundation	5 days	Concreting
3	Construct Formwork	5 days	Casing
4	Tie Rebar	2 days	Armouring
5	Pour Concrete	1 day	Concreting
6	Remove Formwork	2 days	Casing
7	**Abutment 2**	**15 days**	
8	Foundation	5 days	Concreting
9	Construct Formwork	4 days	Casing
10	Tie Rebar	3 days	Armouring
11	Pour Concrete	1 day	Concreting
12	Remove Formwork	3 days	Casing
13	**Pier**	**15 days**	
14	Foundation	4 days	Concreting
15	Head Beam	5 days	Concreting
16	Construct Formwork	2 days	Casing
17	Tie Rebar	3 days	Armouring
18	Pour Concrete	1 day	Concreting
19	Remove Formwork	1 day	Casing
20	**Superstructure**	**20 days**	
21	Bearings	3 days	Concreting
22	Scaffold	4 days	Casing
23	**Girder 1**	**20 days**	
24	Transport	1 day	Casing
25	Lift	1 day	Concreting
26	**Girder 2**	**20 days**	
27	Transport	1 day	Casing
28	Lift	1 day	Concreting
29	Deck	7 days	Concreting
30	Girder Heads	3 days	Concreting
31	Sealing	4 days	Casing
32	Remove Scaffold	1 day	Casing
33	**Parapet 1**	**5 days**	
34	Construct Formwork	2 days	Casing
35	Tie Rebar	2 days	Armouring
36	Pour Concrete	1 day	Concreting
37	Remove Formwork	2 days	Casing
38	**Parapet 2**	**6 days**	
39	Construct Formwork	2 days	Casing
40	Tie Rebar	2 days	Armouring
41	Pour Concrete	1 day	Concreting
42	Remove Formwork	2 days	Casing

Figure 5-21: Feasible results of the forward and backward simulation using the priority-by-end-date method and sequence enforcement constraints between tasks using the same resources: total float

While the presented results of the introduced test case are feasible, three tasks can be identified manually where the sequence enforcement constraints caused an over-constrained case, and the amount of float became shorter than it could be. These tasks are the “remove formwork”, “tie rebar” and “pour concrete” tasks for abutment 2.

By correcting the results manually, the “remove formwork” task of abutment 2 could be pushed forward until the end of day 24. This leads to 9 days of total float time. Under this scenario the „tie rebar” task of the abutment 2 results in 7 days of float time, and the corresponding “pour concrete” task results in 6 days of float (Figure 5-22). When allocating these tasks the described amount of float time, with serialization of the concerned tasks the resource limits and the order of tasks will not be violated.



	Task Name	Duration	Resource Names
1	**Abutment 1**	**13 days**	
2	Foundation	5 days	Concreting
3	Construct Formwork	5 days	Casing
4	Tie Rebar	2 days	Armouring
5	Pour Concrete	1 day	Concreting
6	Remove Formwork	2 days	Casing
7	**Abutment 2**	**15 days**	
8	Foundation	5 days	Concreting
9	Construct Formwork	4 days	Casing
10	Tie Rebar	3 days	Armouring
11	Pour Concrete	1 day	Concreting
12	Remove Formwork	3 days	Casing
13	**Pier**	**15 days**	
14	Foundation	4 days	Concreting
15	Head Beam	5 days	Concreting
16	Construct Formwork	2 days	Casing
17	Tie Rebar	3 days	Armouring
18	Pour Concrete	1 day	Concreting
19	Remove Formwork	1 day	Casing
20	**Superstructure**	**20 days**	
21	Bearings	3 days	Concreting
22	Scaffold	4 days	Casing
23	**Girder 1**	**20 days**	
24	Transport	1 day	Casing
25	Lift	1 day	Concreting
26	**Girder 2**	**20 days**	
27	Transport	1 day	Casing
28	Lift	1 day	Concreting
29	Deck	7 days	Concreting
30	Girder Heads	3 days	Concreting
31	Sealing	4 days	Casing
32	Remove Scaffold	1 day	Casing
33	**Parapet 1**	**5 days**	
34	Construct Formwork	2 days	Casing
35	Tie Rebar	2 days	Armouring
36	Pour Concrete	1 day	Concreting
37	Remove Formwork	2 days	Casing
38	**Parapet 2**	**6 days**	
39	Construct Formwork	2 days	Casing
40	Tie Rebar	2 days	Armouring
41	Pour Concrete	1 day	Concreting
42	Remove Formwork	2 days	Casing

Figure 5-22: The expected result of the float time determination (created manually)

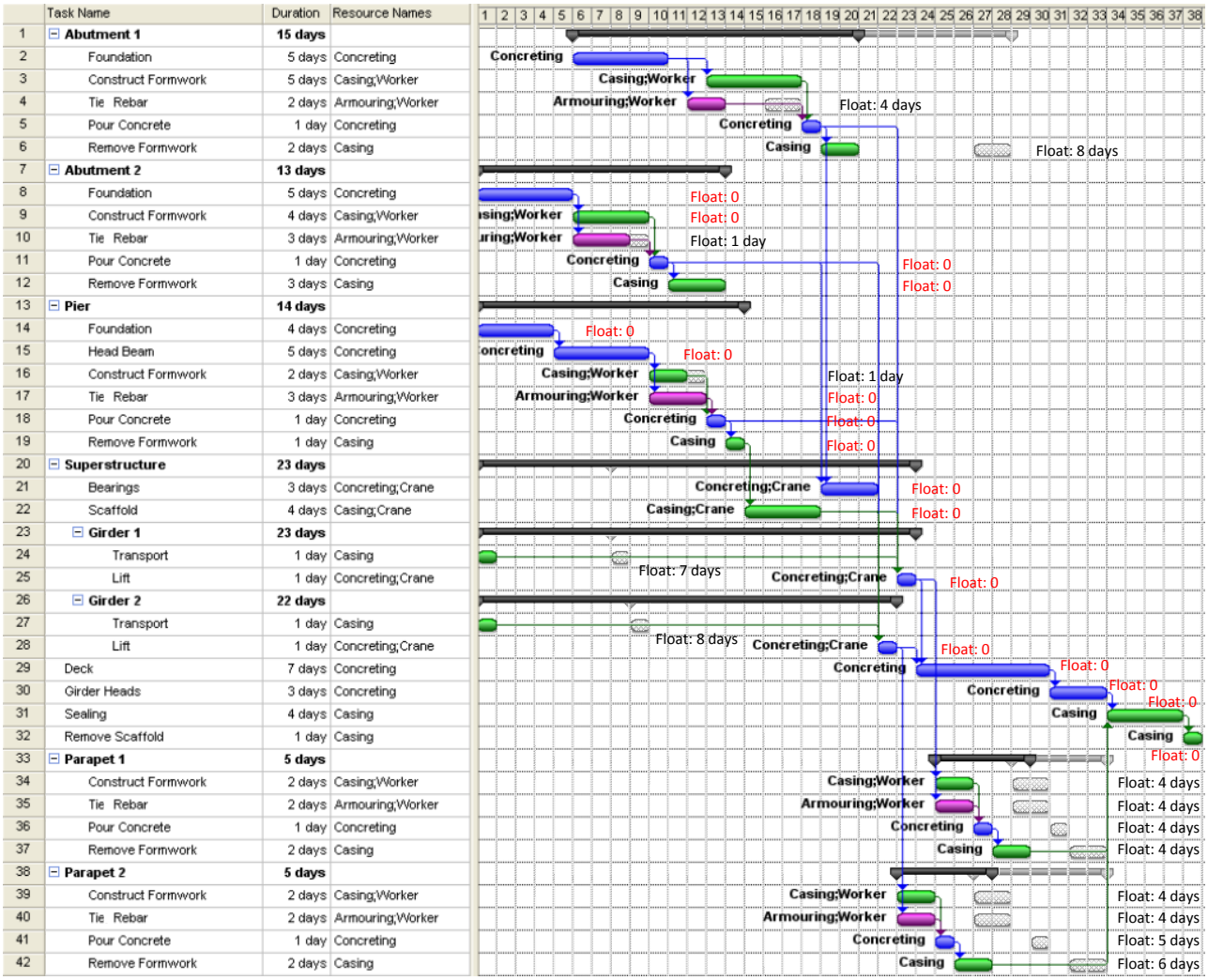
5.7 Taking multiple resource classes into account for construction tasks at float time determination

The determination process of float time for schedules when the tasks require multiple resources from different classes is the same as previously introduced for the one resource class per task case. The difference is that a task will be a part of multiple resource dependency matrices and so it might have more sequence enforcement constraints for the backward simulation than tasks that need only one resource class. Therefore, the more resource classes a task needs for its execution, the more restricted will be the time frame within which it can be moved freely thereby reducing also the possibility of serialization.

To visualize the restricting behavior of multiple resources for the above-introduced bridge construction example, further resource needs for individual tasks have been applied. A crane and a general class of construction worker have been added to the resource management pool in order to

see the effect of multiple resource assignments. The worker has been added as a necessary resource for every "construct formwork" and „tie rebar" tasks. The crane has also been added as a necessary resource for the transport of the two girders and to create the bearings and the scaffold. The amount of available worker units and the one crane have been defined as input data for the simulations. The result of the float time calculation is depicted in Figure 5-23.

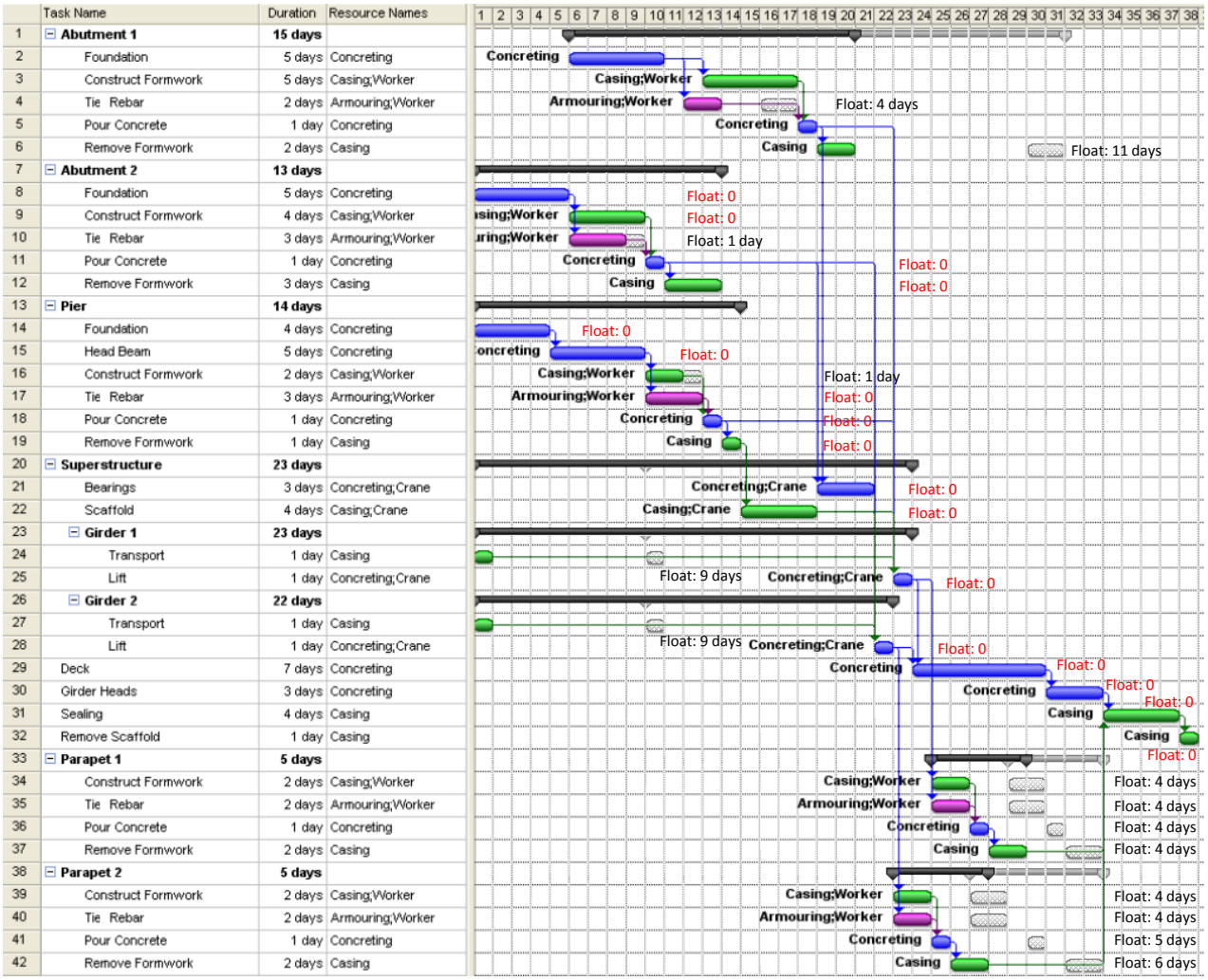
The multiple resource constraints for the concerned tasks resulted in many changes to the schedule. First, the makespan of the project became longer. Second, the float time of the tasks was reduced. Thus, the project has two simultaneous critical chains: one through the construction tasks of the pier, and the other one through the task chain of the abutment 2.



	Task Name	Duration	Resource Names
1	**Abutment 1**	**15 days**	
2	Foundation	5 days	Concreting
3	Construct Formwork	5 days	Casing;Worker
4	Tie Rebar	2 days	Armouring;Worker
5	Pour Concrete	1 day	Concreting
6	Remove Formwork	2 days	Casing
7	**Abutment 2**	**13 days**	
8	Foundation	5 days	Concreting
9	Construct Formwork	4 days	Casing;Worker
10	Tie Rebar	3 days	Armouring;Worker
11	Pour Concrete	1 day	Concreting
12	Remove Formwork	3 days	Casing
13	**Pier**	**14 days**	
14	Foundation	4 days	Concreting
15	Head Beam	5 days	Concreting
16	Construct Formwork	2 days	Casing;Worker
17	Tie Rebar	3 days	Armouring;Worker
18	Pour Concrete	1 day	Concreting
19	Remove Formwork	1 day	Casing
20	**Superstructure**	**23 days**	
21	Bearings	3 days	Concreting;Crane
22	Scaffold	4 days	Casing;Crane
23	**Girder 1**	**23 days**	
24	Transport	1 day	Casing
25	Lift	1 day	Concreting;Crane
26	**Girder 2**	**22 days**	
27	Transport	1 day	Casing
28	Lift	1 day	Concreting;Crane
29	Deck	7 days	Concreting
30	Girder Heads	3 days	Concreting
31	Sealing	4 days	Casing
32	Remove Scaffold	1 day	Casing
33	**Parapet 1**	**5 days**	
34	Construct Formwork	2 days	Casing;Worker
35	Tie Rebar	2 days	Armouring;Worker
36	Pour Concrete	1 day	Concreting
37	Remove Formwork	2 days	Casing
38	**Parapet 2**	**5 days**	
39	Construct Formwork	2 days	Casing;Worker
40	Tie Rebar	2 days	Armouring;Worker
41	Pour Concrete	1 day	Concreting
42	Remove Formwork	2 days	Casing

Figure 5-23: Results of the float time calculation after introducing further resource constraints to the project

Furthermore, due to the multiple resource constraints the possibility of serialization of the tasks is reduced as well. For this example three tasks could be identified that could have more float time: Abutment 1: remove formwork and the two girder transport tasks. The first one could have 3 more days of float, the latter ones 1 and 2 days, respectively (Figure 5-24).



	Task Name	Duration	Resource Names
1	**Abutment 1**	**15 days**	
2	Foundation	5 days	Concreting
3	Construct Formwork	5 days	Casing;Worker
4	Tie Rebar	2 days	Armouring;Worker
5	Pour Concrete	1 day	Concreting
6	Remove Formwork	2 days	Casing
7	**Abutment 2**	**13 days**	
8	Foundation	5 days	Concreting
9	Construct Formwork	4 days	Casing;Worker
10	Tie Rebar	3 days	Armouring;Worker
11	Pour Concrete	1 day	Concreting
12	Remove Formwork	3 days	Casing
13	**Pier**	**14 days**	
14	Foundation	4 days	Concreting
15	Head Beam	5 days	Concreting
16	Construct Formwork	2 days	Casing;Worker
17	Tie Rebar	3 days	Armouring;Worker
18	Pour Concrete	1 day	Concreting
19	Remove Formwork	1 day	Casing
20	**Superstructure**	**23 days**	
21	Bearings	3 days	Concreting;Crane
22	Scaffold	4 days	Casing;Crane
23	**Girder 1**	**23 days**	
24	Transport	1 day	Casing
25	Lift	1 day	Concreting;Crane
26	**Girder 2**	**22 days**	
27	Transport	1 day	Casing
28	Lift	1 day	Concreting;Crane
29	Deck	7 days	Concreting
30	Girder Heads	3 days	Concreting
31	Sealing	4 days	Casing
32	Remove Scaffold	1 day	Casing
33	**Parapet 1**	**5 days**	
34	Construct Formwork	2 days	Casing;Worker
35	Tie Rebar	2 days	Armouring;Worker
36	Pour Concrete	1 day	Concreting
37	Remove Formwork	2 days	Casing
38	**Parapet 2**	**5 days**	
39	Construct Formwork	2 days	Casing;Worker
40	Tie Rebar	2 days	Armouring;Worker
41	Pour Concrete	1 day	Concreting
42	Remove Formwork	2 days	Casing

Figure 5-24: The expected results of the introduced example with multiple resource constraints

5.8 Summary and conclusions on float time determination with constraint-based discrete event simulation

Calculating total float for construction projects while taking into account the available resources is a challenging task. Conventional network planning techniques (e.g. critical path method) are able to analyze and calculate float time but are not adequate for considering resources. By contrast, discrete event simulation is capable of considering resources, but unable to calculate float time. In our research work we use a discrete event simulation engine extended with a constraint-based discrete event simulation methodology to generate feasible construction schedules.

A new methodology has been introduced that extends the constraint-based discrete event simulation with the ability to calculate the total float time for each individual task of the construction project in one iteration step. For this methodology the *backward simulation* in combination with *forward simulation* was used. The most challenging task in the backward simulation is to follow a *task execution sequence identical* to that of the forward simulation. In order to achieve schedule

compatibility between forward and backward simulation, the *execution order of the tasks must be identical* and *every task has to start at the same time or later in time than in the forward simulation*. To this end, the task selection algorithm has been modified so that the backward simulation uses the priority-based *priority-by-end-date* approach to determine the next executable task.

The allocation of priorities according to the end date of a task was important since the decisions about the execution order of the tasks with constraint-based discrete event simulation are made at the beginning of every task. Since the beginning of a task for the backward simulation correlates to the end of that task in the forward simulation, a priority setting according to the end date of the tasks can lead to a similar execution order with the backward simulation that has been generated by the forward simulation. Due to the strict correlation between the end date of a task in the forward simulation and the selection method of the CBDES for the next execution task, any further priority setting rules (according to start-time, based on time interval from project start or to project end, etc.) for the backward simulation might lead to a diverging execution order of the tasks.

To solve the *unintended swap phenomenon* where several independent tasks that use the same resources swap position in the backward simulation, *sequence enforcement constraints* are defined based on the result of the forward simulation.

By comparing the results of the backward simulation after applying these constraints with the schedule of the forward simulation, the execution order of the tasks will be identical and every task will start later in time. Hence, the time difference between the earliest and latest start time of a task represents its total float without exceeding the resource limits at any time during the project. The tasks without total float comprise the critical chain of tasks for the respective configuration of resources. A comprehensive case study was introduced to illustrate the application of this new approach and demonstrated that the determination of detailed total float for each individual task using discrete event simulation taking into consideration available resources is realizable and that the determined results are also feasible. However, in some cases the determined amount of float is lower than is realistically possible.

The lower float time can be explained by the fact that the introduced sequence enforcement constraints were developed to always deliver a feasible result without exceeding the resource limits, even including the time period of the float. This assumes that tasks which were executed simultaneously with the forward simulation will also be executed simultaneously and will be surrounded with the same tasks in the backward simulation. This cannot be guaranteed when there are great differences between the simultaneously executable paths' length, particularly when a task has a larger actual float than the duration of the simultaneously executed independent task.

In this case the two tasks can be pushed after another in time and as one "connected task" they might be pushed further forward in time than apart. On this way they gain more float as would they be pushed separated forward in time. We call this phenomenon *serialization*. Further research should investigate how to loosen the restrictions of the sequence enforcement constraints so that the results are both feasible and correct in every case.

It is important to stress that the introduced total float determination methods always work only with one schedule. Hence, the determined total float of the individual tasks correspond to the schedule that has been determined by the forward simulation. By generating diverging schedules with different task execution orders, the total float for the individual tasks will also be different.

Furthermore, decisions made by the author of this thesis, such as when more independent tasks are executed simultaneously during the forward simulation, the task that finishes later will have the larger float, affect the schedule. By changing these rules diverging float schemes might be generated for the same schedule. Further research should investigate how the total float of a task changes when

this rule is turned around such that the earlier a task finishes, the more float time it gets, or the higher the task's initial priority is, the higher its float time will be.

With the introduced enhancements of the simulation technique such as the methods for data preparation and the float time determination while taking also resource constraints and limits into account, the simulation-based scheduling methods have transcended the functionalities of the conventionally used network-based scheduling techniques. To gain further advantage for the simulation-based scheduling approach over these conventional techniques and to make it more attractive for the industry, a CBDES-based optimization strategy will be introduced in the next chapters.

6 Optimization of construction schedules – State-of-art

6.1 Executive summary

The last two chapters introduced methods to determine feasible solutions for the resource-constrained project scheduling problem (RCPSP). The RCPSP is a combinatorial optimization problem that also accurately describes the optimization problem of construction schedules under resource constraints with the goal of finding the schedule with the shortest makespan. In this chapter the basic principles of optimization will be introduced.

In general, optimization problems are categorized in different ways. The first sort of categorization is according to the number of variables that should be optimized. When there is only one variable that should be optimized then it is a single variable optimization problem. For example, in the RCPSP the variable is the makespan of the project. When there are more variables, such as makespan, cost, quality, etc., it is categorizes as a multi-objective optimization problem. Since the main topic of this thesis is the solution of the RCPSP, only the basic principles of a single variable optimization will be introduced in this chapter. This is approached in Section 6.2.

Section 6.3 discusses a further categorization of optimization problems that is made according to the distribution of the search space. The search space of an optimization problem can either be continuous or discrete. A combinatorial optimization problem has a discrete search space or a search space that can be reduced to discrete. The RCPSP also belongs in this class, therefore it will be introduced in detail.

Section 6.4 discusses the complexity of optimization problems of the RCPSP. The RCPSP belongs to the class of NP-hard optimization problems that are probably unsolvable in polynomial time. After the introduction of the characteristics of an NP-hard problem, solution strategies will be introduced in Section 6.5.

The discussed solution strategies were selected because they are all capable of solving the RCPSP. In these sections, after the detailed introduction of the solution method, the application of the method on the RCPSP will be discussed. The chapter is closed by a summary and an outlook.

6.2 Single variable optimization

Optimization problems can be categorized in different ways. The basic categorization is done according to the amount of variables that should be optimized. When the amount of optimization variables is one, it is a single variable optimization problem. When there are more variables, it is called a multi-objective optimization problem. Since the resource-constrained project scheduling problem (RCPSP) is a single variable optimization problem that searches for the one schedule with the shortest makespan, the single variable optimization will be introduced in detail in this section.

First, let us introduce this problem in general by considering a simple *continuous* optimization problem that is shown in Figure 6-1. The goal of the single variable optimization is to find the one optimal solution from the search space X that best fits the objective function $f : X \rightarrow R$. Single variable optimization problems can be divided into two subcategories according to the nature of the objective function. When the search aims to find the maximum solution, it is a maximization problem; in contrast, when the search looks for the minimum solution, it forms a minimization problem. Maximization problems can be turned into minimization problems and vice versa by simply changing the sign of the objective function (duality principle e.g. Rao 2009).

An optimal solution in the search space can either be local or global. A solution x^* is considered as local optimum when there is no other solution existing in the neighborhood[15] of x^* that has a better objective function value than $f(x^*)$. This means that for a maximization problem $f(x^*) \geq f(x') \; \forall x' \in X'$ where $X' \subseteq X$ is the neighborhood of x´.

Furthermore, x^{**} is considered as global optimum of the optimization problem when no other solution exists with a better objective function value in the entire search space than $f(x^{**})$. This means that for a maximization problem $f(x^{**}) \geq f(x) \; \forall x \in X$. Since $X' \subseteq X$ the global optimum of the search space is always a local optimum as well (see Figure 6-1). The minima of an optimization are described by the same equations as above with inverted relationships (Deb 2004).

[15] For detailed description see Section 6.3, Figure 6-3 and Figure 6-4.

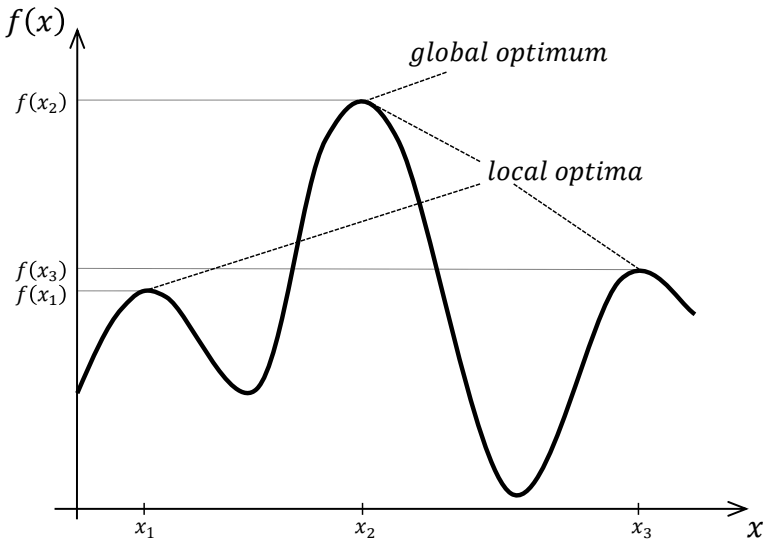


Figure 6-1: Local optima and global optimum of a general continuous single value objective function f(x) for a maximization problem

These equations can describe an optimization problem when the complete search space X is considered as feasible. In reality, the solution space is often restricted by boundary conditions, represented by equations or inequations, so that the feasible search space is reduced to $F \subseteq X$. In addition, F is not necessarily a contiguous space (see Figure 6-2). The solutions of the feasible search space are also called candidate solutions. For the fragmentation of the feasible search space, infeasible solutions have to be used as intermediate steps in some cases in order to reach some candidate solutions. These infeasible solution intermediate steps are not useful results for the optimization problem but in many cases they must be determined in order to reach further feasible solutions.

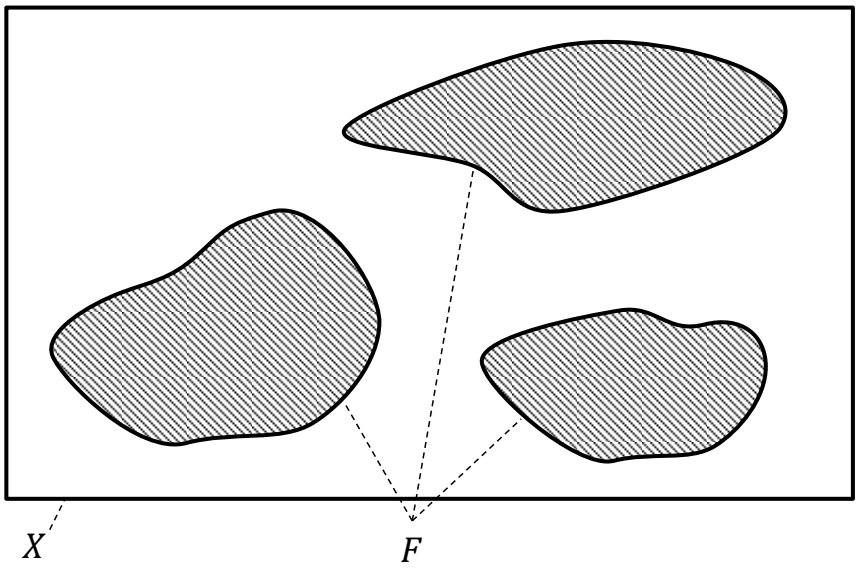


Figure 6-2: The solution space X and the fragmented feasible solution space F for a 2 dimensional space. The space inside X and outside of F represents the non-feasible solutions.

6.3 Combinatorial optimization problem

Another classification scheme of optimization problems can be achieved according to the distribution of the problem's search space. The search space of an optimization problem can either be *continuous* (see Figure 6-3) or *discrete* (see Figure 6-4). An optimization problem with a discrete search space, or with a space that can be reduced to discrete, is called a discrete or combinatorial optimization problem. As already introduced in Section 2.2.2 the resource-constrained project scheduling problem (RCPSP) belongs to the class of combinatorial optimization problems. The solutions of the RCPSP are discrete schedules that represent different sequence combinations of the corresponding tasks.

A further difference between continuous and combinatorial optimization problems is that in continuous optimization problems, generally the objective is a real number or function, such as the value of $f(x_2)$ in Figure 6-1. In the combinatorial case, the objective is a discrete object or a set of objects (Papadimitriou and Steiglitz 1998).

One important characteristic of optimization problems is how neighborhood solutions are defined. Neighborhood solutions can influence both the effectiveness and computational effort of the optimization and play an important role in exploring the search space.

The neighborhood solutions for a continuous problem with a continuous search space can easily be defined by a Euclidian distance range from the considered point (see Figure 6-3). In contrast, neighborhood solutions for discrete problems with discretized solution spaces are not as clearly definable. Often different definitions for neighborhood solutions exist for the same problem.

In Figure 6-4 two examples of search algorithms that are based on traversing between discrete neighborhood solutions are presented. The favored algorithm only traverses between feasible solutions, while the other one also determines non-feasible solutions that could significantly increase the computational effort of the algorithm. The aim of the author was to find an algorithm that could traverse through feasible solutions as a solution for the RCPSP. The developed algorithm that can traverse only through feasible solutions for the RCPSP will be presented in Chapter 7.

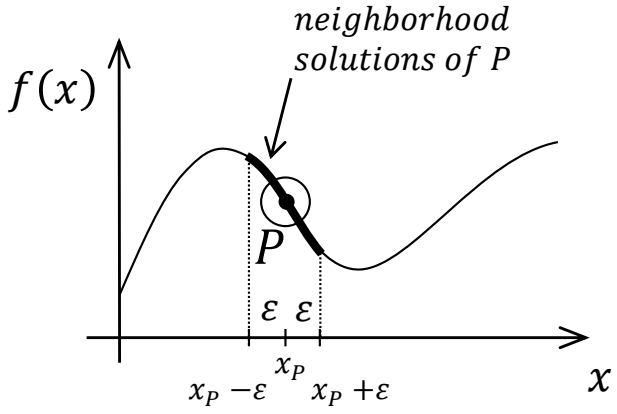


Figure 6-3: Neighborhood solutions for a point P in a continuous solution space: every point Y whose distance from x_P is smaller than a predefined Euclidian distance of ε. $\forall Y \in X, x_P - \varepsilon \leq x_Y \leq x_P + \varepsilon$

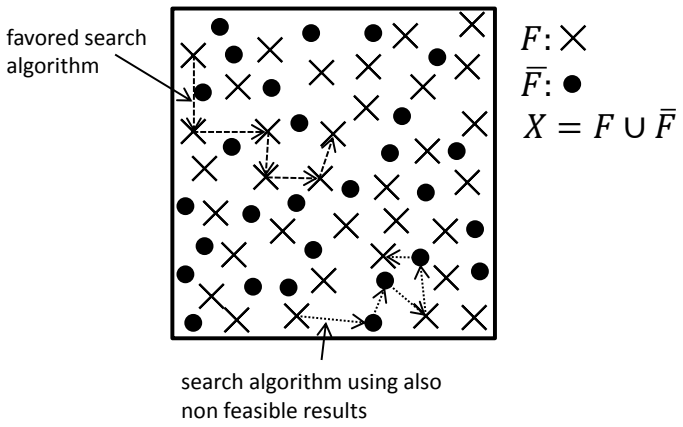


Figure 6-4: Different possibilities of neighborhood solution-based algorithms for a discrete search space to traverse between the solutions. Feasible solutions are marked with a cross, infeasible with a round circle. The first search algorithm also considers infeasible solutions in some cases. The favored search algorithm traverses only between feasible solutions.

Combinatorial problems in general deal with tuples[16], permutations[17], matroids[18], graphs[19] and related structures in most of the cases. The most common problems that involve combinatorial optimization are the traveling salesman problem, the minimum spanning tree problem, the knapsack problem and the vehicle routing problem (Michalewicz and Fogel 2004).

For the traveling salesman problem, a list of cities and the distances between each pair of cities is given. The objective is to find the shortest possible route that visits every single city exactly once and returns back to the city of origin. When there are *n* cities in the model, the complete search space is made up of $n!/(2n)$ solutions (Michalewicz and Fogel 2004). When the amount of cities is, for example 20 the number of solutions is higher than $6 * 10^{16}$. To find the one optimal solution from all of these results can become impossible.

The knapsack problem is defined by a set of items. Every item has a mass and a value. The goal is to find a collection of items whose total weight equals a predefined weight or is under a limit, and whose total value is as large as possible. Similar to the traveling salesman problem, solving this problem can become complex and unsolvable in polynomial time, even in cases where there are just a small number of items.

All of these problems have the combinatorial explosion in common. This means that increasing the size (n) of the problem will excessively increase the size of the search space X. In most cases $|X|$ is a function of the factorial of the problem size, so $|X| = g(n!)$. Presenting an example with a function of n!, the size of the solution space for $n = 5$ objects will be 120. Increasing the size of n by one ($n = 6$), the size of the search space increases to 720 solutions. A problem with $n = 20$ objects will have a complete search space consisting of $2{,}432 * 10^{18}$ solutions.

For smaller problems such as where $n = 5$ or 6, the determination of every possible solution is achievable. This solution method is called the *total enumeration*. For bigger problems, such as where $n = 20$ and more, this type of solution technique becomes impossible due to the high number of solutions. Therefore, further solution techniques must be considered. Possible techniques for this situation will be introduced in Section 6.5.

[16] Ordered list of elements
[17] Arrangement of a set of objects into a particular order
[18] Or independence structure is a structure that captures and generalizes the notion of linear independence in vector spaces
[19] A representation of a set of objects connected with links to each other

Considering the RCPSP where the problem size (n) equals the number of tasks that have to be executed, even for a smaller construction project $n \geq 50$. As mentioned above, the total enumeration of a problem with that size is impossible, so further solution methods are necessary to optimize these construction schedules. To find a suitable solution method, first the complexity of the problem must be discussed.

6.4 NP-hard optimization problem

To describe the complexity of optimization problems, let us now concentrate on *decision problems*. These are problems with the solution of either yes or no. Every optimization problem can be transformed into a decision problem. For example for a minimization problem: change the objective of the optimization instead of seeking the minimum value, ask the question: "is there a solution smaller than k?" (Garey and Johnson 1979).

Every decision problem that can be solved in polynomial time[20] belongs to the class P. A decision problem is called NP (Nondeterministic Polynomial-time) when proof exists that can be verified in polynomial time for the yes answers. However, to find such proof can become very time consuming. Alternatively, NP problems can be defined as decision problems which can be solved by a nondeterministic Turing-machine in polynomial time (Garey and Johnson 1979).

It can be proven that $NP \subseteq P$, since, when a decision problem can be solved in polynomial time (P), it can also be verified in polynomial time (NP). However, an open question is whether $P = NP$ or $P \neq NP$[21].

In addition, a decision problem D is NP-complete if $D \subseteq NP$, making every problem in NP reducible to D. In other words, the problem D can be solved using the same problem with modified formulation. Thus, NP-complete problems can always be transformed into each other. However, obtaining a solution to NP-complete problems is very difficult and so far inefficient. Every known deterministic solution algorithm has exponential complexity which makes the solution of the problem impossible in large cases.

Furthermore, when a problem is at least as hard as the hardest problems in NP, it is called NP-hard. Since NP-complete problems are the hardest problems of NP, they are also considered to be NP-hard. *Consequently, if an NP-hard problem could be solved in polynomial time, it would make it possible to solve all NP problems in polynomial time.*

Common to every NP-hard decision problem is that they can only be solved by exponential or factorial time complexity. Solving these problems with total enumeration is beyond the limits of computational power. Exact solutions though can be determined by using modifications of the total enumeration (Garey and Johnson 1979). For example, an accepted approach is to divide the search space into smaller areas and ignore regions where the optimum solution definitely cannot be found (Branch-and-Bound). Since the resource-constrained project scheduling problem and the optimization of construction schedules are considered as NP-hard problems (Blazewicz et al. 1983),

[20] An algorithm is said to be solvable in polynomial time when the running time of the problem is bounded from the top by a polynomial expression. It means that the number of steps required to complete the algorithm for a given input has the time complexity of $O(n^k)$ for some non-negative integer constant k, where n is the complexity of the input. Problems like addition, multiplication, sorting, shortest path search, linear programming etc. belong to this class.

[21] If an NP-complete problem exists which is polynomial-time solvable, then P = NP. However, if some problem in NP is not polynomial-time solvable, then P ≠ NP and all NP-complete problems are not polynomial-time solvable. (For additional discussion regarding this problem, please see in da Costa et al. (2007))

the different solution methods for these specific problems will be described in detail in the next sections.

6.5 Solution strategies for solving NP-hard optimization problems

Solving an NP-hard optimization problem, that has an exponential complexity and therefore a large workspace (such as the RCPSP), with total enumeration is not a conceivable way. Other exact solution methods for these problems, such as *Branch-and-Bound or Integer Programming*, require a significant amount of knowledge about either the solvable problem or the structure of the search space. Without this knowledge these exact solution methods cannot be applied and the only solution left is the total enumeration. These exact solution methods will be introduced briefly in Section 6.5.1.

As an alternative solution, the heuristic approaches can be applied, searching not for the optimal solution exactly, but for one good solution that is near to the exact optimum. These methods are based on decisions that can significantly reduce the computational effort and can lead to a good, near-optimal solution. These decisions are based on common sense or intuition such as the rule of thumb, a good guess or estimations. The heuristic approaches to solve the RCPSP are introduced in Section 6.5.2.

6.5.1 Exact solution methods for the RCPSP

The RCPSP is considered to be an NP-hard combinatorial optimization problem. Due to the combinatorial explosion for larger problems the search space of the problem might become so large that it can no longer be solved by total enumeration. The basic idea behind the exact solution methods is to find a way to restrict the search space to a smaller search space or to smaller search spaces in which the exact optimal solution must be found. Therefore, exact optimization methods always require a large amount of background knowledge about the problem that has to be solved. When the problem is slightly changed, the applied algorithms must be changed as well.

6.5.1.1 Integer programming

One approach to determine the exact solution of the RCPSP is to turn it into an Integer Programming (IP) problem and solve it. IP is similar to common linear programming (LP)[22] with the restriction that some or all of the variables must take on integer values. The LP problems are considered as solvable in polynomial time (Khachiyan 1979). This small restriction (integer values) of the IP makes the problem NP-hard (Garey and Johnson 1979). This difference between an LP and an IP problem will have an important role later when solving the IP problem. When all of the variables are restricted to integer values, it is a pure *integer programming* problem (IP); when only some of them are restricted, it is called a *mixed integer programming* (MIP) problem. When all the variables are restricted to the value of one or zero, the IP is called a *binary integer programming* (BIP) problem.

[22] Linear programming is a method to determine an optimal solution in a mathematical model whose requirements are represented by linear relationships.

Let us now discuss the general parameters, restrictions and how to solve a general IP problem. Given the following formulation of a general linear programming problem: $c = (c_1, \ldots, c_n)$, $b = (b_1, \ldots, b_m)$, a matrix A of constraints with m rows and n columns (and entry a_{ij} in row i and column j), and I a subset of $\{1, \ldots, n\}$. Find the solution vector $x = (x_1, \ldots, x_n)$, while solving the following problem: $max\ (c^T x)$ subject to:

$$Ax \leq b \qquad (1)$$

$$x \geq 0 \qquad (2)$$

Extending this formulation with the restriction that x_j is an *integer* whenever $j \in I$ (3), then the result is the formulation of an IP problem. Thus, when I is empty, it is a simple LP problem, and when it is not empty, but does not contain all of $\{1, \ldots, n\}$, it is an MIP problem. To visualize the differences between the three different methods, a specific example is introduced:

First, let us consider an LP problem, which should maximize the objective function $x_1 + x_2$ subject to $a_{11}x_1 + a_{12}x_2 \leq b_1$; $a_{21}x_1 + a_{22}x_2 \leq b_2$; $a_{31}x_1 + a_{32}x_2 \leq b_3$ (which corresponds to inequation 1 above) and $x_1; x_2 \geq 0$ (inequation 2 above). The solution of the problem is presented in Figure 6-5. It is visible and coherent how the restrictions reduce the size of the feasible search space. It has been proven that the potential optima of the problems are always in the intersection points of the restricting equations (Dantzig 1951). In our case, the five potential optima are marked with solid circles in Figure 6-5. These possible optima have to be passed to the objective function for x_1 and x_2 and the point with best result is the optimum.

With the restriction that the values of x_1 and x_2 must be integers (restriction 3), the feasible solutions make up the discrete points within the space of the restricting equations (see Figure 6-6). Now this has turned into an IP problem. Allowing x_2 to being continuous, the solvable problem describes an MIP problem, since x_1 must be an integer and x_2 is continuous. In this case, the feasible solutions build parallel lines within the feasible region (see Figure 6-6). These restrictions to the search space turned both IP and MIP problems into NP-hard problems (Garey and Johnson 1979). Let us assume, that the optimum of the LP problem was around the intersection point of the first and second restricting inequation (2,4; 5,1). Since these values are not integers the application of further methods, to find the optimum of the IP and the MIP problem must be contemplated.

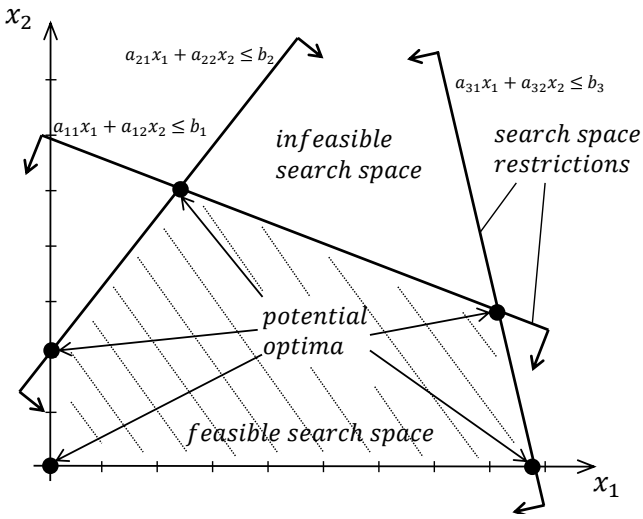


Figure 6-5: Representation of the introduced two-dimensional LP problem

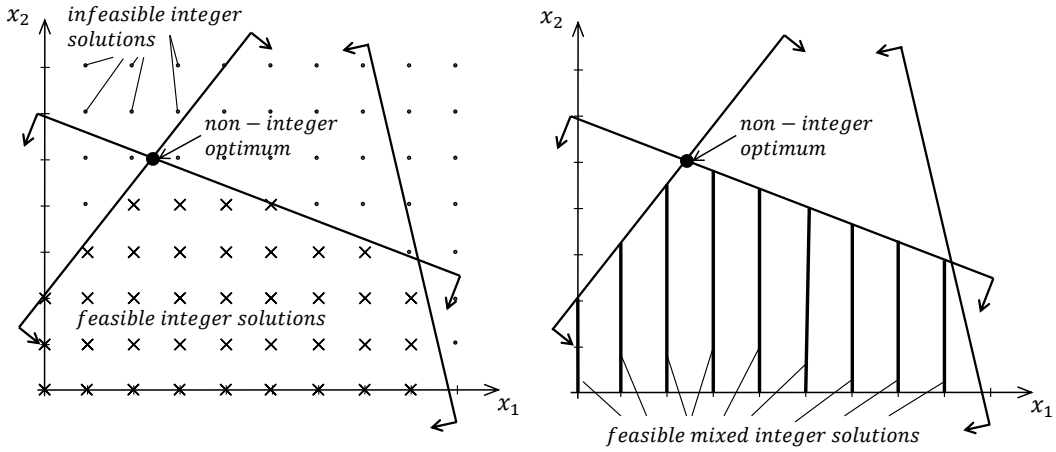


Figure 6-6: IP (left) and MIP (right) representation of the introduced two-dimensional problem (the feasible integer solutions are marked with X.

Solution methods for IP problems

According to Land and Doig (1960) the best strategy to solve IP and MIP problems developed in the last 50 years was to apply the *Branch-and-Bound* method. Nowadays this method is still a powerful and popular method, since in the last decades several methods have been developed based on the theory of the Branch-and-Bound. The common strategy in every solution approach is to divide the original problem into smaller sub problems with the goal of recursively solving each of them (Galati 2010). The difficulty for the solution is how to define accurate and *tight boundaries* for the feasible integer search space. There are three commonly applied methods to define these boundaries, however they are frequently combined into hybrid solution methods as described by Genova and Guliashki (2011).

The first class of methods which was introduced by Gomory in the 1950´s (Gomory 1958) consists of *cutting plane algorithms* that are based on polyhedral combinatorics. The cutting plane method cuts the feasible integer solutions out of the LP solutions of the polyhedron, assuring that every feasible IP solution is still inside the remaining region. A depiction of how this algorithm aims to tighten the bounding area around the feasible IP solutions is presented in Figure 6-7.

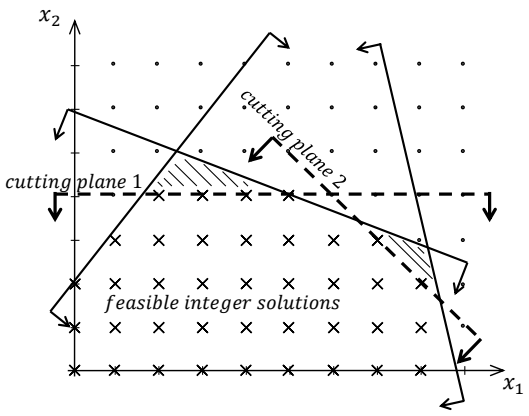


Figure 6-7: Cutting plane method: Cutting out bounding non-integer LP solutions so that every integer solution stays inside the considered solution space.

The second class of methods to solve the IP are the *enumerative approaches* such as *Branch-and-Bound (B&B)*, *Branch-and-Cut (B&C)* and *Branch-and-Price (B&P)*. Of these, the most commonly used enumerative approach is the Branch-and-Bound method consisting of two phases. During the *branching phase*, the algorithm divides the problem into smaller sub-problems and for each sub-problem the algorithm seeks to obtain a bound on how good its best feasible solution can be. In the *bounding phase*, the algorithm seeks to eliminate sub-problems that do not contain the optimum and so they will not be investigated further. By integrating these two algorithms, a tree structure of the search can be introduced. The branching process creates new branches of the enumeration tree and the bounding process cuts off branches that will not be investigated further (see Figure 6-8). The first approach for solving IP problems with Branch-and-Bound was introduced by Land and Doig (1960). The branching process of the IP problem introduced above is shown in Figure 6-9, where the search space is divided into two separate parts: left and right from the LP optimum of the problem.

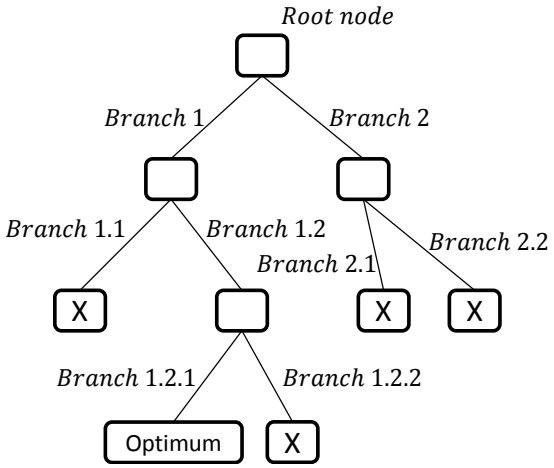


Figure 6-8: Tree structure of the B&B approach (enumeration tree). X: fathomed branches

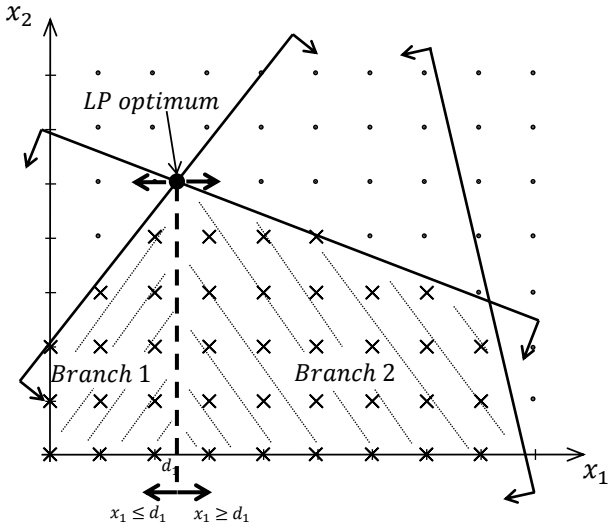


Figure 6-9: Branching process of the introduced IP problem

Combining the Branch-and-Bound technique with the cutting plane algorithm results in a new and powerful class of IP problem solving algorithm: the *Branch-and-Cut* method. The advantage of this

method is that by applying the cutting plane algorithm it tightens the bounds for the branching, restricting those that take on integer values, steering the algorithm to the optimal solution in a better and faster way (see Figure 6-10). A detailed description of this method can be found in Caccetta and Hill (2001) and Mitchell (2002).

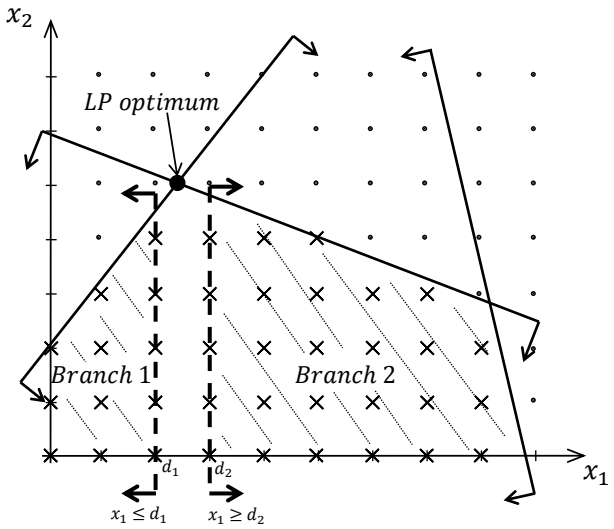


Figure 6-10: The branching process of the introduced IP problem combined with the cutting plane algorithm (B&C)

The third method of the enumerative approaches is the *Branch-and-Price* algorithm, which is similar to the Branch-and-Cut method, except that it focuses on column generation rather than row generation for the enumeration tree. For every node in the enumeration tree further columns might be generated in order to tighten the search space. One column, also called the "pricing problem", creates a new sub-problem, which is solved to identify columns that are allowed to enter the basis. If such columns are found, the bounds of the search space will be updated for the LP problem, and the possible optima will be searched for. Branching occurs when no columns "price" is able to enter the basis and the solution does not satisfy the integrality condition. A detailed review of the column generation and the Branch-and-Price algorithm can be found in Barnhart et al. (1998).

Branch-and-Cut and Branch-and-Price are similar and complementary procedures that can also be combined resulting in the *Branch-and-Cut-and-Price* algorithm. When the methods are combined, cuts and columns are generated dynamically. Further description of this method is provided by Ladányi et al. (2001). A detailed introduction of solving RCPSP with Branch-and-Bound approaches will be introduced in Section 6.5.1.2.

The third category of IP problem solving methods is the *Relaxation and Decomposition* methods. There are three widely used relaxation approaches for the IP problems that create lower bounds for minimization problems or upper bounds for maximization problems: *Linear Programming Relaxation*, *Combinatorial Relaxation* and *Lagrangian Relaxation*. The first two methods extend the search space of the IP problem without changing the objective function, while the third one defines a new objective function which results in the same or a greater value[23] in the fixed feasible search space (Genova and Guliashki 2011).

The first relaxation method is the *Linear Programming (LP) relaxation*, which is achieved by deleting the integrality constraint (3), allowing continuous values as solutions, and retaining the same objective function and constraints. Thereby, an in polynomial time solvable LP problem was generated which bounds the solvable IP problem (Shmuel 1954). This method has already been

[23] For a maximization problem; for a minimization problem smaller value.

applied to the introduced examples (see Figure 6-5 - Figure 6-10) to generate bounds for the B&B or B&C methods. When the optimum of the LP relaxation is not an integer solution, further decomposition methods must be applied, such as the Branch-and-Bound, Branch-and-Cut, etc. The primary reason to apply the LP relaxation method is its ability to efficiently and reliably solve LP problems such as the simplex method, the duality principle (Dantzig1951) and the interior point method (Karmarkar 1984).

The second relaxation approach is the *combinatorial relaxation*. This method removes a set of inequality constraints to make the IP problem easier to solve. While the determination of the LP relaxation is a straightforward procedure, the determination of the combinatorial relaxation is more complex. The structural approach (Genova and Guliashki 2011) applies algorithms that determine the necessary bounds of the objective function and are often solved by graph-theoric methods (see Iwata and Murota 2001).

The third relaxation approach is the *Lagrangian relaxation* method described by Geoffrion (1974) and Fisher (1981). The Lagrangian relaxation aims to relax the complicated constraints matrix (1) and to bring the constraints with associated Lagrangian multipliers into the objective function. In this way, the resulting sub-problems are easier to solve. The sub-problems are solved repetitively until an optimal solution is found for every multiplier. A decomposition method for the optimization, e.g. the Benders decomposition method, can be used, to solve the problem iteratively (see Benders 1962).

Since all the relaxation methods mentioned above restrict the search space for the IP problem, all of them can be combined with the Branch-and-Bound algorithms. The last two methods are special-purpose algorithms and the LP relaxation can be applied generally to every IP problem.

IP problem formulations for the RCPSP

To solve the RCPSP as an IP problem, first, it must be converted into an IP problem. This is a complicated process since the formulation can differ from problem to problem. The most common formulation is achieved as a Mixed Integer Programming (MIP) problem, which can be divided into three further categories[24].

The first category collects the *0-1 time-indexed formulations* of the MIP problem. These formulations are based on the variable x_{it}, where $x_{it} = 1$ if, and only if, the process i starts exactly at time t, otherwise defined as 0. Defining T as the upper bound of the project duration and assuming integer process durations, the objective function of the problem can be defined as the minimization of T. The basic discrete-time formulation (DT) was introduced by Pritsker et al. (1969) and a similar approach called disaggregated discrete-time (DDT) was developed by Christofides et al. (1987) with a different formulation of the precedence constraints. Demeulemeester et al. (1994) discusses some issues with the DDT formulation of Christofides and introduces a new branching strategy that "guarantees the determination of the optimal solution in all instances of the problem at the expense of an increase in node evaluations and average CPU time". Further time indexed formulations for the RCPSP were used by Stinson et al. (1978) and Fisher (1970).

The second category of MIP problem formulation for RCPSP are the *sequence-based disjunctive MILP-formulations*, also called *flow-based formulations*. These formulations involve binary variables y_{ij}, where i and j are processes, and S_i are continuous variables that describe the start time of the process i. When the duration of the process i is p_i, then $y_{ij} = 1$ if, and only if, $S_i + p_i \leq S_j$. To apply this method to the RCPSP further variables must be defined that model the resource flow. These are discussed by Artigues et al. (2008).

The third category of the formulations are the *event-based formulations*. In this case the binary variable z_{ie} is used to describe that process i either starts, ends, or is in process at event e. Events

[24] The description of the three categories follows the description of Koné et al. 2011

occur when a process starts or ends, and the continuous variable t_e is used to describe the start time of an event. This method was originally proposed for single machine scheduling but has been extended to other fields also to solve RCPSP (Zapata et al. 2008, Koné et al. 2011 and Artigues et al. 2013).

Event-based formulations and further hybrid approaches connecting integer programming (IP) with *constraint programming* (CP) and *satisfiability testing* (SAT) are the actual state-of-art in RCPSP modeling as an IP problem (Artigues et al. 2013 and Berthold et al. 2010a). Many years have passed since the statement of Hadley in 1964 about the formulation of the RCPSP as an IP:

"For any realistic problem the number of constraints will be huge and a solution is, at present, quite impossible. However, it is interesting to show that the problem can be formulated as an integer programming problem."

Since that time, both the mathematical and computational improvements have resulted in significant progress in solving IP formulations of the RCPSP. This process is now assisted by automated solvers for IP problems, such as SCIP, LINDO, FICO Xpress, and by LP problem solvers, such as LAZYFX, CPLEX, QSopt-ex providing higher and lower bounds for the IP problem. These software programs and research results made it possible to determine a feasible (but not necessarily optimal) solution for an IP formulation of the RCPSP with 30 activities in reasonable time (Koné et al. 2011). However for bigger problems with 480 instances and 60 activities it is hard to determine the optimal solution, even with the newest hybrid approach (Berthold et al. 2010b). Due to this high computation time and complex formulation of the problem, many researchers are turning to use simpler and faster methods such as heuristic approaches, at the expense of the accuracy, to solve the RCPSP (Section 6.5.2). In the next section the Branch-and-Bound technique, another exact solution method which has already been mentioned, will be introduced in detail.

6.5.1.2 Branch-and-Bound

As described in the last section, *Branch-and-Bound* is a powerful technique to solve IP problems. However, this method alone is also capable of providing solutions for the RCPSP. The general process of the Branch-and-Bound calculation is the follows:

First, the complete problem is subdivided into simpler sub-problems (branching) for a general problem. Next these sub-problems will be solved and enumerated. Thereafter, sub-problems where the optimal solution clearly cannot be found are fathomed and eliminated from further investigation.

The determination process of the solutions can be represented on a so called *enumeration tree* that is made up of nodes and branches. The tree begins with the root node, which in general represents e.g. the whole problem. After dividing these problem into sub-problems the sub-problems will be represented as nodes on the next level of the tree connected with a branch to the root node and so on. When a node becomes fathomed no further calculations will be performed with the corresponding node (See Figure 6-8 on page 136). Another possible interpretation of the enumeration tree is when the nodes of an enumeration tree represent discrete solutions for an optimization problem, such as individual schedules for the RCPSP. In this case the enumeration tree can be used to traverse through the search space of the problem and to ignore non promising solutions and branches. There are three possible ways to traverse through the nodes of an enumeration tree:

1. *Depth First Search*: In this case, the enumeration tree is explored from the root node along one single branch as deep as possible without backtracking[25] (see Figure 6-11). An example of the application of this technique being used to solve the RCPSP can be found in De Reyck

[25] Without jumping back to nodes on the higher level until the deep search is finished.

and Herroelen (1998), in which every node of the enumeration tree represents the same network of tasks and on every level they are extended with extra precedence relationships in order to solve resource exceeding conflicts.

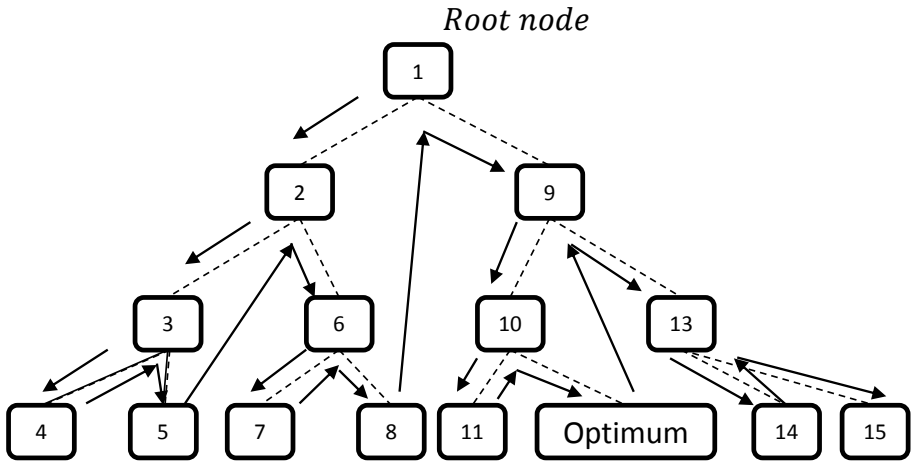


Figure 6-11: Schematic representation of the working mechanism of the depth first search without fathoming any nodes (numbers: order of determination)

2. *Breadth First Search*: In contrast to the Depth First Search, this exploration of the enumeration tree occurs not top-down, but rather sidelong. First, every neighbor node of one level will be examined and only after that level is complete will the exploration jump one level deeper (see Figure 6-12). An example for this approach for the solution of the RCPSP is the enumeration tree presented by Stinson et al. (1978). In this tree, every node is a partial schedule, representing scheduling decisions for the complete network.

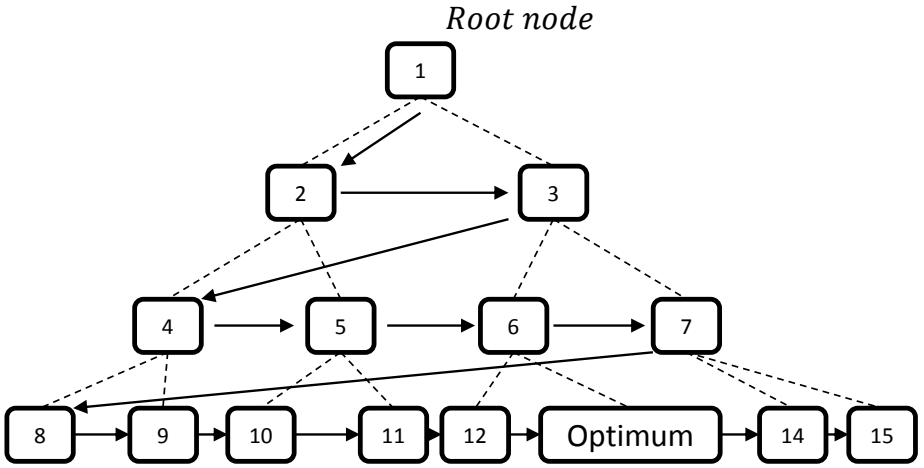


Figure 6-12: Schematic representation of the working mechanism of the breadth first search without fathoming any nodes (numbers: order of determination)

3. *Best First Search*: This method always selects the most promising solution on one level for further investigation and further branching (see Figure 6-13). An example for the best first search is the Dijkstra´s algorithm (Dijkstra 1959). This algorithm is a solution method for finding the shortest path between two nodes of a network. After searching for possible routes,

it always takes the shortest one for further investigation until the desired destination node is reached (Dijkstra 1959).

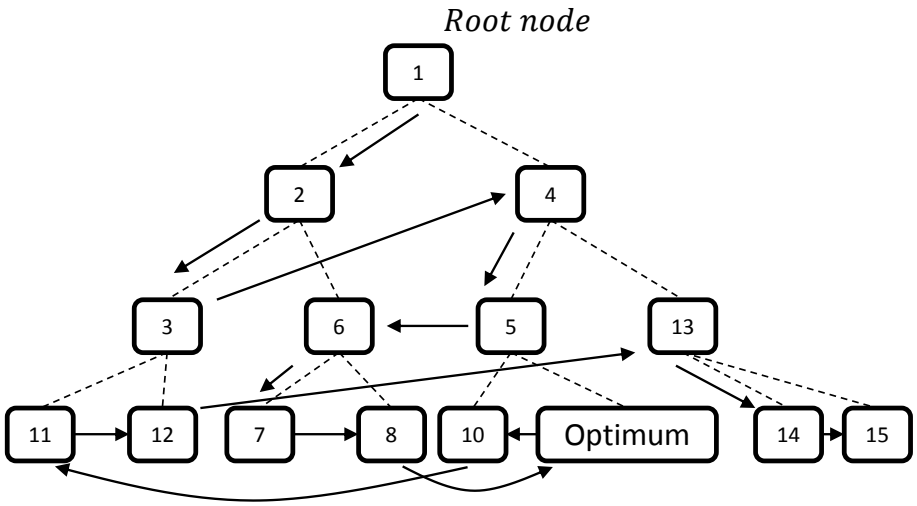


Figure 6-13: Schematic representation of the working mechanism of the best first search without fathoming any nodes (numbers: order of determination)

Since the 1960's a variety of Branch-and-Bound algorithms have been developed to solve the RCPSP. The different alternatives use diverse branching schemes and pruning methods in order to find the one schedule with the shortest total duration.[26] The main functionalities of these three methods are similar to each other: The Branch-and-Bound algorithm reduces the search space or the nodes of the enumeration tree as long as it is sure that the optimum is still inside the search space. After identifying all of the non-fathomed results or nodes, the optimum can be identified. This method can work for small problems as presented in the next example. However for a larger problem it becomes time consuming and inefficient, since it must evaluate every non-fathomed node.

The *Precedence Tree* was introduced by Patterson et al. (1989). This method starts the enumeration tree with a dummy task at the top and then for every further level it extends the "schedule" with one eligible task whose predecessors are already scheduled. Following a depth first search, the scheduling process runs down from the top to the bottom of the tree until every task is scheduled. Then a backtracking to the previous level occurs and the next eligible task is chosen to be scheduled. Thus, a new branch for the tree is created.

Each branch from the root to a leaf node is one complete schedule and corresponds to the permutation of the precedence feasible set[27] of tasks. An example is presented in Figure 6-14. Here four tasks must be scheduled: 1, 2, 3 and 4. Two precedence relations are defined: one between 1→3 and another between 2→4. For this example we will assume that the duration of every task is one time instant. Task 2 requires 2 units of resources, task 1, 3 and 4 each require one unit. The amount of available resources is limited to 2 units.

[26] The description of the alternative methods follows the structure of (Brucker et al. 1999)

[27] Precedence feasible set: each predecessor of a task has a smaller index in the set than the task itself (Brucker et al. 1999)

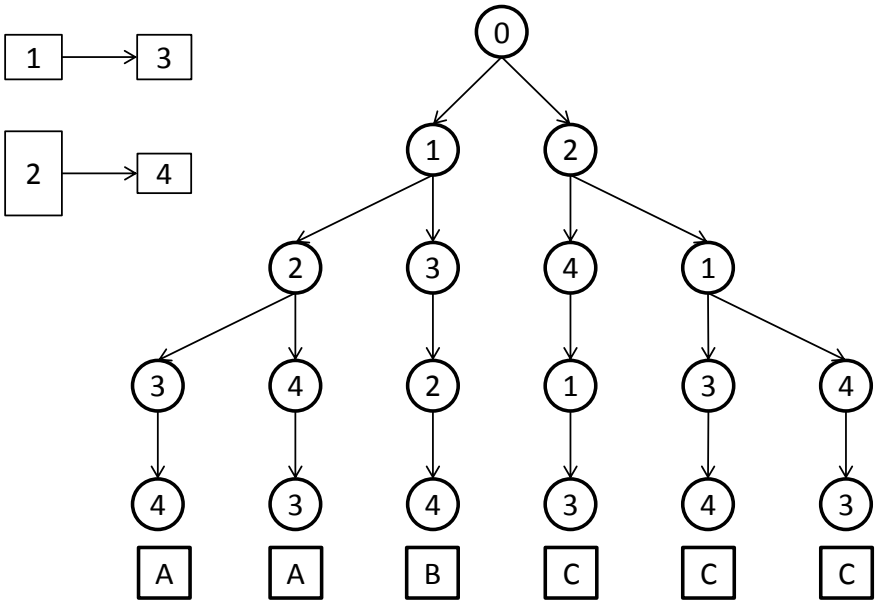


Figure 6-14: Representation of a precedence tree for the task chains 1→3 and 2→4. The higher the activity the more resources it needs. The letters in the boxes at the bottom represent the resulting schedules from Figure 6-15.

Starting the tree with the dummy element 0, the first tasks that are eligible to be started are tasks 1 and 2. These tasks form the first level and the first two branches of the precedence tree. The next possibility is following the branch of task 1, task 2 or task 3 so that these could be scheduled next. Similarly task 2, task 4 and task 1 could be executed as followers. These tasks build the next level of the tree. This branching process proceeds until every possible sequence of the four tasks has been determined. To generate a schedule, the tasks will be placed into the schedule according to their order in a branch of the precedence tree, and are pushed back in time as long as the resource limits and their precedence constraints allow. In this way three different schedules are determined for the introduced example (see Figure 6-15). The first schedule, schedule A, belongs to the first and second branch, the second possible schedule, schedule B can be determined by the third branch, and the schedule C, is developed from the evaluation of the fourth, fifth and sixth branches.

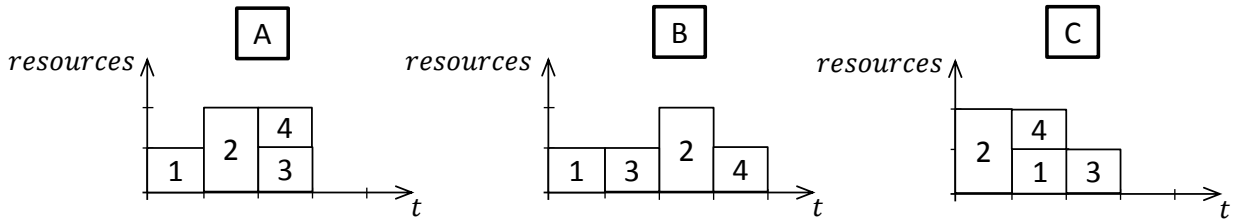


Figure 6-15: The three different schedules determined by the precedence tree

An explanation for the same results for different branches is that even though the order of the tasks was different after determining the schedule, the tasks could be executed simultaneously due to the amount of available resources. By decreasing the available amount of resources to one unit (and assuming that task 2 only requires one unit of resource instead of two), six different schedules would result. As a conclusion it can be said that if two tasks can be started at the same time their sequence order in the precedence tree is insignificant. Therefore the further tree parts of one of the tasks can be

neglected. The results would be identical with the results of the other tasks branches. Hence, by fathoming parts of the enumeration tree, the analysis of the precedence tree can be accelerated. As a result of the precedence tree a list of possible schedules has been determined which also contain the actual optimal schedule. The optimal schedule can be found after the evaluation of the resulting schedules.

In this case the Branch-and-Bound technique was used to prune those branches out of the precedence tree that will achieve the same results as another branch thus reducing the search space of the problem.

The second approach in using Branch-and-Bound for RCPSP is the enumeration tree with the *Delay Alternatives*. The major difference between this tree and the aforementioned one is that instead of tasks, it works with partial schedules. Each level of the tree represents one time instant (decision point) where non-scheduled tasks may be started. This method was introduced by Christofides (1987) and then improved upon by Demeulemesteer and Herroelen (1994). Further enhancements were made by De Reyck and Herroelen in order to solve RCPSP with generalized precedence relationships (De Reyck and Herroelen 1998).

The algorithm begins with an empty schedule. During the first step it collects every task that could be started at the examined time frame. The branching process operates by delaying different combinations of tasks, so that the necessary resources remain under the predefined resource limit. During the next time frame new tasks might be added to the schedule, which might result in the resource limits being exceeded again. Thus, further branches have to be created by pushing tasks forward in time. The process ends when every task has been scheduled without exceeding the resource limits.

The Branch-and-Bound tree of the previous example determined by delay alternatives is presented in Figure 6-16. At time step 0 (t=0) tasks 1 and 2 could be started but this will exceed the resource limits so either task 1 or task 2 has to be delayed. These two forward pushes create the first two branches of the tree at time step 1. If task 2 is pushed forward and task 1 is scheduled that means that task 3 could be started simultaneously with task 2. This would once again exceed the resource limits, so either task 2 or task 3 have to be delayed, creating two new branches for the tree (t=2). Delaying task 3 results in the schedule A, since in the next step task 4 can be executed in parallel with task 3 without exceeding the resource limits. While pushing task 2 forward in time, task 4 has to wait until task 2 is scheduled due to the precedence constraint. Thus, at time step 3 no tasks have to be delayed since the resource limits have not been exceeded. Next, task 4 can be scheduled and the result is schedule B from Figure 6-15. Return to the first branching process (t=1) and select the next branch where task 1 is delayed. Now task 4 can be scheduled parallel with task 1 at the next time step. Thereafter task 3 can be scheduled after task 1 and task 4 without delaying any tasks resulting in the schedule C. In this way every possible branch has been identified for the presented problem. After identifying and evaluating all these schedules, the optimum schedule can be selected.

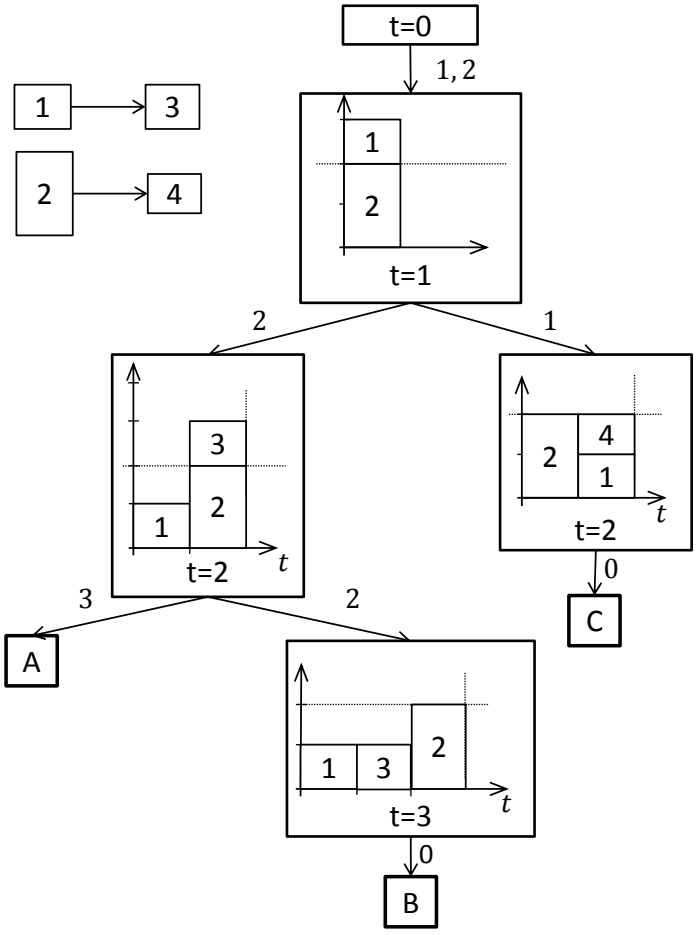


Figure 6-16: The Branch-and-Bound tree of the example determined by delay alternatives

In comparison with the precedence tree method, this approach allows the user to withdraw scheduling decisions at the current level rather than only at a lower level (Brucker et al. 1999). This allows the user to make scheduling decisions during the creation of the tree and not just afterwards.

The third method for creating the Branch-and-Bound tree is the *Extension Alternatives*. Similar to the last delay-based approach, the levels of the extension alternatives tree symbolize decision points and contain partial schedules. This method also starts with an empty schedule and searches for possible task combinations that could be scheduled that do not exceed the resource limits. These task combinations build the different branches of the tree. After selecting one combination the algorithm jumps to the next decision point and again collects possible executable task combinations under the resource limit. This process continues until every task has been scheduled. The interpretation of the previous example is presented in Figure 6-17.

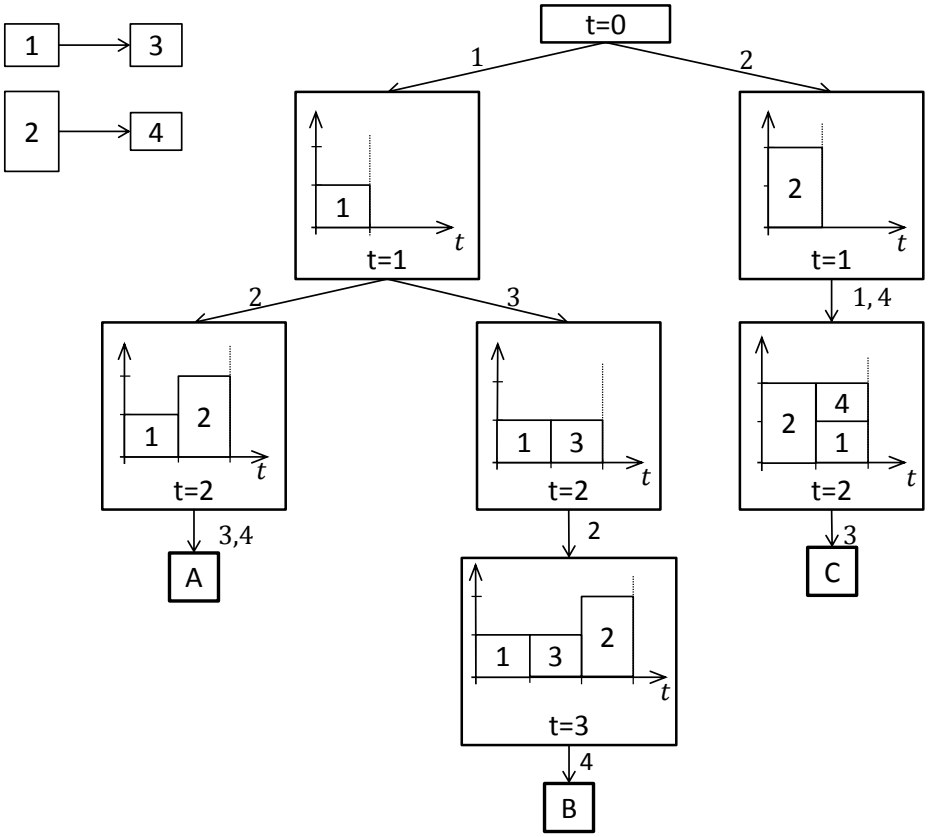


Figure 6-17: The Branch-and-Bound tree of the previous example determined by the extension alternatives

Starting with an empty schedule, the first possible extensions are either starting with task 1 or task 2. They cannot be started simultaneously because in that case the resource limits would be exceeded. Further, tasks 3 and 4 cannot be started because of the precedence relationships to tasks 1 and 2. Following the branch of task 1, then task 2 or task 3 could be scheduled at the next decision point (t=1). These two tasks structure the next two branches. If task 2 is scheduled first, then task 3 and task 4 can be executed simultaneously in the next step, resulting in the schedule B. Scheduling task 3 after task 1 is the only possibility to schedule task 2 after task 3 (t=3) and task 4 after that. This results in schedule A (see Figure 6-15). When starting the schedule with task 2, at the second decision point (t=1) the schedule can be extended with task 1 and task 4, which can be executed simultaneously. The last extension is task 3 and creating schedule C. Note that while the previous method allows the user to delay tasks that are scheduled on a lower level, this method does not allow the withdrawal of scheduling decisions from the lower levels. This restriction leads to a smaller search space for the extension alternatives method without losing optimality. The optimum can be identified after evaluating the schedules.

One further approach to represent the Branch-and-Bound tree is the *Block Extensions* method, introduced by Mingozzi (1998). This approach uses time intervals as markers for the levels in the Branch-and-Bound tree. The blocks represent a list of tasks that can be scheduled jointly within the defined time interval of the level. Moreover, partial schedules are represented as the sequence of blocks. The branching process occurs by adding different new blocks to the partial schedule.

Brucker et al. (Brucker et al. 1998) introduced a further Branch-and-Bound algorithm using *Schedule Schemes*. Schedule schemes represent sets of feasible schedules using different relations

between tasks like parallelism (N), conjunction (C), disjunction (D) and flexibility (F). Every $S = (C, D, N, F)$ vector represents a single schedule of the search space. The root of the Branch-and-Bound tree contains the scheduling scheme $(C_0, D_0, 0, F)$. Selecting one flexibility relation and turning it into a disjunction or a parallelism relation causes branching. The branching process is repeated until $F = 0$ is reached[28] for every possible scheme. An example for the application of this approach is presented in Brucker et al. (1998).

The last Branch-and-Bound algorithm is based on the *Minimal Forbidden Sets*. It was introduced by Igelmund und Radermacher (1983a and 1983b) and was also used by Stork and Uetz (2005). Since a similar approach is used for our optimization algorithm, the minimal forbidden sets method will be introduced in detail[29]. Let's assume that $V = \{1,2, \dots, n\}$ is a set of tasks that has to be scheduled. $(V, \prec)$ is a partially ordered set of tasks according to the precedence constraints so that if $task\, i \prec task\, j$, then j can not be started before i has been completed. In order to start a task, it needs different types of resources: $k = 1, \dots, d$. The available amount of resources for the resource type k is defined by $b_k \geq 0$. Each task j requires $0 \leq a_{kj} \leq b_k$ units of resources during its execution.

A schedule is considered to be feasible if at any time t, all the precedence constraints are respected and the sum of necessary resources for the tasks in process $(S(t) \subseteq V)$ does not exceed the amount of available resources b_k. Given the latter description of the RCPSP, "a subset $F \subseteq V$ of tasks is called a *forbidden set*, if F is an anti-chain[30] with respect to the partial order $(V, \prec)$ and some resource type k exists with $\sum_{j \in F} a_{kj} > b_k$. If F is forbidden, but no proper subset of F is forbidden, then F is a *minimal forbidden set* (Stork and Uetz 2005). In other words, a minimal forbidden subset of the tasks is when all the n tasks of the subset are pairwise not linked through any precedence relationship to each other, however, they must not be scheduled simultaneously, because in that case at least one resource limit would be exceeded.

The Branch-and-Bound algorithm is used to create a tree of minimal forbidden sets where every leaf represents a minimal forbidden set. At the beginning the tree contains every possible subset of the nodes. A node u is discarded as soon as it is clear that neither the corresponding set U nor any superset of U that is located in the subtree rooted at u is a minimal forbidden set. This process is repeated recursively as long as the tree only contains minimal forbidden sets. The exact description of the algorithm can be found with an example in Stork and Uetz (2005).

The advantage of forbidden sets is that when all of the resource conflicts caused by every minimal forbidden set can be solved, they can be used to generate a feasible schedule. One possible solution would be to pause one task until more resources are again available. This leads to combinatorial problems on graphs with AND/OR precedence constraints (Möhring et al. 2004). It has been shown that certain IP formulations for the RCPSP can be based on minimal forbidden sets. As such, minimal forbidden sets might be used to derive cutting planes and even facets of the scheduling polyhedron (Olaguibel and Goerlich 1993).

Determination of higher and lower bounds for the Branch-and-Bound-based RCPSP calculations

In order to decrease the size of the search space of the Branch-and-Bound algorithm, certain branches, where the optimum surely cannot be found, should be fathomed and so removed from further calculations. To determine if the optimum could be inside a sub-region, *higher* and *lower*

[28] It means that no more change can be introduced to the schedule scheme.
[29] The RCPSP formulation of (Stork and Uetz 2005) is used to introduce the meaning of minimal forbidden sets.
[30] If P is a finite partially ordered set, then an antichain in P is a set of pairwise incomparable elements. In Section 5.4 these anti-chain elements are described as independent tasks.

bounds are used which try to approximate the optimum from above and below, respectively. The tighter these bounds are, the earlier a branch could be fathomed thereby reducing the calculation time even more.

If it can be proven that a reliable lower bound of the examined branch exceeds the higher bound, the branch can be fathomed. For RCPSP the commonly used value for the *higher bound* is the makespan of the *current best schedule*. After determining some elements of the enumeration tree this can provide a good upper bound for the optimum. In contrast, to determine an accurate and tight lower bound for the RCPSP is a more challenging task.

One possible lower bound is the *relaxation of the resource constraints*. This leads to the *critical path* length, which is a very simple bound. With this method partial schedules can be extended with the length of the critical path of the nonscheduled tasks. When this result exceeds the duration of the higher bound, the examined branch can be fathomed. Since the lower bound is the possible lowest duration of the branch and the higher bound represents the so far found best solution, every further calculation on this branch will result in a suboptimal schedule. This approach can fathom branches at the later phases of the calculations when the big part of the tasks has already been scheduled and the difference between the optimum and the partial schedules is small.

Stinson et al. (1978) introduce a second possible lower bound for the RCPSP by *relaxing the precedence constraints*. This is called the *resource-based lower bound*. In this case, the already scheduled partial schedules will be extended by taking only the resource constraints and limits into account.

Chaleshtari and Shadrokh (2012) use the so called *precedence-based earliest start time* and the *resource-based earliest start time* of the tasks in order to determine the feasible lower bound for the RCPSP with cumulative resources. This is based on the aforementioned two methods.

Stinson et al. (1978) introduce a third approach called the *critical sequence lower bound*. It takes the critical path of not yet scheduled tasks and inserts an extra task into the sequence that is not part of the critical path. The determination of the lower bound simultaneously takes both precedence and resource constraints into account, gaining a higher lower bound in certain cases than for the critical path-based lower bound.

Christofides et al. (1987) introduce a *Linear Programming-based* approach similar to the critical sequencing lower bound. First, they relax the resource constraints. After that they assume, that task a and b cannot be executed simultaneously because of the resource limits. Therefore in every schedule, task a and b will be executed after each other, although their relative order is unknown. Therefore they create two subproblems: one, when a is before b and second, when b is before a. If both of the solutions exceed the current best solution, the partial solution cannot lead to a better solution, so the node can be fathomed. The method can be extended for higher amount of tasks as well.

Mingozzi et al. (1998) introduced an *LP-based partial relaxation of the precedence constraints* allowing preemption, and provide lower bounds for the RCPSP. These lower bounds are proven to be tighter than the critical sequence lower bound of Stinson et al. (1978) (Kolisch et al. 1995).

Demeulemeester and Herroelen (1997) have improved this method and proved it to be more powerful in combination with the critical path-based lower bound than the one from Mingozzi. It is based on a list of companions for every task which can represent e.g. tasks, which can be executed simultaneously, respecting both the resource and precedence constraints. "All unscheduled task i with non-zero duration are then entered in a list L in non-decreasing order of the number of companions (non-increasing duration as a tie-breaker). The following procedure then yields a lower bound, called LB3, for the partial schedule under consideration:

LB3:= the earliest completion time of the tasks in progress
While list L not empty do:
Take task j on top of list L and determine its duration d_j;
LB3:=LB3 + d_j;
Remove task j and its companions from list L;
End do."

A detailed example for the application of the method is presented by Demeulemeester and Herroelen (1997).

One further approach applies *Column Generation Technique* in order to solve the linear program directly. The original approach was introduced by Gilmore and Gomory (1963) and has been enhanced by Brucker and Knust (1998) taking time windows into account for the tasks. "These time windows are derived from precedence constraints and using a fictitious upper bound T for the makespan. Now the columns correspond to sets of tasks, which can be scheduled jointly in the given time window. The objective is to find a preemptive schedule that respects all the time windows. If such a schedule does not exist, is a lower bound." (Brucker et al. 1999)

Two further and obvious approaches to generate lower bounds for the RCPSP are the *LP-relaxation* and the *Lagrangian-relaxation* (introduced in Section 6.5.1.1) of the IP formulation of the problem.

These aforementioned approaches can also be combined with each other and the one with the highest result can be used as the lower bound for the examined schedule or partial schedule. Determining tight lower and higher bounds for the Branch-and-Bound solution of the RCPSP can reduce the search space and so largely accelerate the calculation. Therefore, Brucker et al. were the first, who were able to verify 326 of the 480 benchmark problems with 60 tasks under an hour time limit (Brucker et al. 1999). Since there is a high potential to use Branch-and-Bound algorithms for optimization problems like the RCPSP, a new Branch-and-Bound-based heuristic algorithm has been developed to determine near optimal solutions for simulation-based schedule optimization problems. The new approach will be introduced in Section 7.5.

6.5.2 Heuristic approaches to solve the RCPSP

Although the deterministic solution approaches are able to find the optimal solution for the RCPSP, they need a very high computational time to deliver this result. Therefore, further approaches have been developed that are capable to deliver not the optimal but one good, near optimal solution for the problem in a shorter computational time. These solution approaches are called the heuristic methods and are based on previous experiences and intuitions.

At the early stages[31] heuristic algorithms were developed with the focus on solving single complex combinatorial optimization problems. Later, the focus of the research turned to find a way to apply these methods to different problem classes instead of finding a single solution method for a single problem (Henderson and Nelson 2006). These general applicable heuristics that can be applied for a wide range of optimization problems are called *metaheuristics*. The common characteristic of these methods is that they only define the general strategies of how to solve the problem. The fine tuning of the optimization parameters must be completed by the user, however these vary for every single

[31] Before the 1980s

problem. Guidelines how to use heuristics in combinatorial optimization can be found in Hertz and Widmer (2003).

The general solution method of an optimization problem with metaheuristics starts by obtaining an initial solution or an initial set of solutions. During an improving search guided by the rules of the applied method better solutions will be searched for. The search continues till either the search is stopped manually, a maximum number of iterations is reached or the best solution so far has not been improved for a predefined amount of iterations. In every iteration step there is a solution or set of solutions that represent the current best solution of the search. Therefore, stopping the search at any time will deliver a current best solution. Applied metaheuristic approaches such as simulated annealing, genetic algorithm, tabu search or ant colony optimization to solve the RCPSP will be presented in Section 6.5.2.2.

6.5.2.1 Heuristics

The most known and frequently applied heuristics to solve the RCPSP are the X-pass, also referred to as priority rule-based heuristics. As its name says it applies a priority rule that will decide which task to start in tied situations, e.g. when two tasks are competing for the same resource. As Brucker et al. (1999) describes, the advantages of priority-based scheduling are its intuitiveness, easy implementation and fast computational effort.

Priority rules can be very diverging. Kolisch and Hartmann (1999) collected the list of applied priority rules and distinguished between network, time, resource-based and lower and upper bound rules. Network-based priorities can be applied e.g. according to the rank of the task within the precedence graph, or depending on how near the task is to the resource unconstrained critical path making its priority higher. Secondly, time-based priorities can be applied e.g. according to the processing time of the task or the float time of the task. Applying resource-based priorities according to the necessary resources of the task causes the priority to increase the more resource is needed or vice versa might be.

Further classification such as local or global rules can be applied according to the amount of employed information. Local rules only employ information from the considered task itself, while global rules also consider further information such as the project makespan, robustness etc.

When the priority of a task stays constant for the iteration process of the schedule generation it is called static priority. When it changes during scheduling it is a dynamic priority. A list of corresponding literature is collected by Kolisch and Hartmann (1999).

The oldest heuristics are the *single pass methods* that apply one priority rule to generate one feasible schedule. A list of literature about how different priority rules can be applied to generate one schedule is collected by Kolisch and Hartmann (1999 and 2006) and Brucker et al. (1999).

In the so called *multi-pass methods* not only one but several feasible schedules are generated for the same problem. According to Kolisch and Hartmann (1999) the most common multi-pass methods are the multi-priority rule methods, the forward-backward scheduling methods and the sampling methods. The first method applies different priority rules in order to generate schedules. The forward-backward scheduling methods alternate between the forward and backward scheduling iteratively to generate schedules. Sampling methods use one priority rule to determine different schedules by "biasing the priority rule through a random machine" (Hartmann and Kolisch 2000).

As Brucker et al. (1999) describes, the advantages of the priority-based scheduling over mathematical approaches are its intuitiveness, easy implementation and fast computational effort. Due to these advantages and their simplicity they have been widely adopted to solve problems in construction schedule optimization (Zhou et al. 2013).

Further, more elaborate heuristics such as the truncated Branch-and-Bound approach, integer-based heuristics and disjunctive arc concepts are mentioned by Brucker et al. (1999). A detailed description of the Fondahl's method, the Structural model, the Siemens approximation and structural stiffness heuristics is found in Kolisch and Hartmann (1999) and Zhou et al. (2013).

6.5.2.2 Metaheuristics

Metaheuristics have mainly been developed to solve the hardest combinatorial optimization problems with a discrete search space like the RCPSP. The general working mechanism of a metaheuristic algorithm is to first define one or a set of initial solutions that will be iteratively enhanced in regard to a desired criteria using assumptions and instincts (Zhou et al. 2013). In this section the metaheuristics used to solve the RCPSP such as the greedy method, tabu search, simulated annealing and the population-based metaheuristics like the genetic algorithm, the particle swarm optimization and the ant colony optimization will briefly be introduced.

Greedy algorithm

The greedy heuristic algorithm, which is also called best fit strategy solves the optimization problem in a series of steps. Its specialty is that at every iteration step when a value has been assigned to the decision variables only the one best solution will be selected for further calculations. Hence, the algorithm assumes that at every iteration step the current best solution provides the best possible move to find the optimal solution, therefore the name greedy algorithm (Michalewicz and Fogel 2004). As Gass and Harris (2001) describe, the greedy algorithm is " a heuristic algorithm that at every step selects the best choice available at that step without regard to future consequences". This could however lead to a situation where the algorithm stacks into a local optimum, since it cannot find any better solution in the surrounding area and therefore it cannot find the actual optimal solution.

To exit such local optima an extension is necessary that allows it to also consider worse results than the current best solution. However such extension will make the simple algorithm more complex.

Solution strategies with greedy algorithm for combinatorial optimization problems such as the traveling salesman problem are described by Michalewicz and Fogel (2004). The so called greedy randomized adaptive search procedure (GRASP) is used for the heuristic solution of the multi-mode RCPSP. Since the multi-mode RCPSP is not the topic of this thesis, the method will not be introduced in detail. A detailed description of the method and its applications can be found in Feo and Resende (1989), Festa and Resende (2001), R. Alvarez-Valdes et al. (2008) and Ranjbar (2012). The working mechanism of the GRASP is similar to the greedy algorithm: each GRASP-iteration consists of two phases. In the first construction phase randomized feasible solutions are produced and in the second phase, the improvement or local-search phase, a local optimum is searched for in the neighborhood of the constructed solutions (Ranjbar 2012). During the greedy randomized construction the algorithm randomly selects elements from the candidate list and adaptively reevaluates the best result. The local search algorithm works iteratively and replaces the current best solution by a better solution in the neighborhood of the current solution.

A GRASP heuristic with constraint-based simulation (CBDES) for the generation of daily efficient schedules in regard to specific execution strategies has been developed by König et al. (2009b). The GRASP heuristic is used to extend the start task routine of the CBDES in order to determine the next executable tasks and its successors on the corresponding day of the schedule. Afterwards an optimized order of the concerned tasks and resource assignments is identified (König et al. 2009b).

In the construction phase of this GRASP an initial schedule is generated for a working day. For every new working day an event is generated. On the corresponding working day executable tasks are stored in the daily candidate list and in a topological ordered graph of the tasks. The schedule and the resource assignments for the day are determined based on this daily candidate list.

The local search phase of the GRASP is based on a standard tabu search algorithm, which will be introduced in detail in the next section. To generate neighboring solutions a random swap between two candidates of the daily candidate list considering the fulfillment of both precedence and resource constraints is applied. An executed substitution of the candidates is then prohibited for a predefined amount of further steps. Once an improved solution is found the current best solution will be updated. The search stops when a predefined number of iterations is reached without improvement (König et al. 2009b).

One further greedy-like heuristic algorithm extended with a tolerance factor CBDES for the solution of the single mode RCPSP will be introduced in Section 7.5.2.

Tabu search

Tabu search is the extension of the greedy or best fit search heuristic which allows it to exit from local optima, through forbidding previously executed moves that might lead back to an already visited result for a predefined amount of time or calculation steps. The method was developed by Glover (1989 and 1990). During optimization, it evaluates the neighborhood solutions and proceeds with the best solution. To exit local optima and avoid cyclic behavior such as returning back to the same node time after some time a tabu list has been set up, which acts as a memory for the search. The purpose of the tabu list is to forbid the neighborhood moves that might lead back to recently visited solutions such forcing the search to explore new fields of the search space and to escape from local optima.

Therefore, in contrast to the greedy algorithm, the tabu search heuristic can accept worse solutions than the actual one. How many of those solutions are accepted depends on how long a move stays in the forbidden set. Defining this value too small, the search might not escape from local optima, defining it too large maybe causes potential good solutions to be excluded from the search.

For the RCPSP Baar et al. (1999) developed two tabu search algorithms. Their schedule generation is based on the schedule scheme representation of the project, in which relations like conjunction, disjunction, parallelism and flexibility are defined. To determine a feasible schedule from the representation form they developed a decoding procedure that determines a feasible schedule that satisfies every disjoint relation. To generate neighborhood solutions they turn flexibility relations into parallel relations and vice versa. The dynamic tabu list is used to forbid previous relation turns (Hartmann and Kolisch 2000).

Artigues et al. (2003) iteratively selects tasks from the schedule, which first will be deleted from the schedule and then reinserted based on their network-flow-based insertion algorithm. The tabu list is constructed of the task itself and its resource predecessors and successors (Kolisch and Hartmann 2006).

Skowronski et al. (2013) introduced a tabu search-based heuristic algorithm to solve the multi-skilled RCPSP, where tasks need resources that have all the necessary, mostly diverging skills. They generate neighborhood solutions on two different ways: the swap-based neighborhood and the random-based neighborhood. “The first one bases on the swapping resources within the pair of tasks, while the second approach assumes assigning any resource that is capable of performing given task (Skowronski et al. 2013)”. The tabu list consists of previous moves that are forbidden in order to discover new solutions for the problem.

Further tabu search-based solution methods for the RCPSP such as the method from Nobe and Ibaraki or from Thomas and Salhi are collected in Kolisch and Hartmann (2006).

Simulated annealing

Simulated annealing is a metaheuristic that is capable to escape local optima, similar to the tabu search algorithm. The basic analogy of the simulated annealing is taken from thermodynamics. To grow crystals the process begins with the heating of raw materials into the molted state. Then, the temperature will be reduced and the crystallization process begins. When the temperature sinks too fast, irregulations will appear in the crystal structure which lead to a higher trapped energy state of the crystal as for a perfectly structured crystal. The following analogy between the physical system and the optimization problem can be drawn. The feasible solution can be represented as the state of the system, where an evaluation function represents the energy stored in the system. The optimal solution is the ground state of the system and the control parameter temperature can be applied to model (Michalewicz and Fogel 2004).

The simulated annealing optimization method was introduced by Kirkpatrick et al. (1983). The solution starts with a set of initial solutions. Neighborhood solutions are generated by "perturbing" the current solutions. When a solution is better than the current optimum it will be accepted directly. If it is worse it must satisfy the evaluation criteria (e.g. temperature) to be accepted and to be used to determine further solutions. As the algorithm proceeds, the limit of the evaluation criteria is lowered and such the probability to accept worse results than the current best one sinks (Kolisch and Hartmann 1999).

For the solution of the RCPSP Bouleimen and Lecocq (2003) introduced a simulated annealing-based method. To generate schedules they used a precedence feasible activity list representation of the project, where each task in the list must appear later than its predecessors. A neighborhood solution is generated by shifting one task in the list such that it still stays precedence feasible. They applied a multiple cooling chain that is progressively reducing the temperature and also the acceptance rate.

König and Beißert (2009) introduced a simulated annealing method that uses the constraint-based discrete event simulation to generate feasible schedules. As input for the simulation they use an execution list that defines the position of a task within the schedule. To create neighborhood solutions they randomly substitute two construction tasks of the same rank in the execution list. The rank of a task depends on its ancestor degree and is determined by the maximum length of connected ancestors within the precedence graph (König and Beißert 2009). The new list is then used as input for the CBDES that generates the feasible neighborhood solutions. To lower the acceptance rate after the iteration steps they used a linear decreasing temperature value. The limitation of this method will be discussed in Section 7.2, where a similar CBDES-based simulated annealing method will be introduced.

Population-based optimization, the evolutionary algorithms

The previously introduced heuristic optimization methods all only consider one single current best solution in order to generate neighborhood solutions. The idea behind the evolutionary algorithms is not to consider one, but to consider a whole population of solutions. These optimization techniques are based on the natural evolution and/or on the social behavior of species that is guided by learning, adaption and evolution (Elbeltagi et al. 2005). The most widely used evolutionary algorithms are the genetic algorithms, the particle swarm optimization and the ant colony optimization. These three techniques and their application on the RCPSP will be introduced in the next sections. Further

evolutionary algorithms, such as memetic algorithms, shuffled frog leaping methods are briefly described by Elbeltagi et al. (2005).

Genetic algorithms

Genetic algorithms (GA) have been introduced by Holland (1975) and were the first evolutionary algorithms. They are inspired by the improved fitness of biological systems or populations through the evolution. The three most important steps of the GA are the *cross over*, *mutation* and *selection*. After determining an initial population of solutions for the optimization problem, two existing solutions will be mated (cross over) and/or mutated in order to generate new ones (offspring) (Kolisch and Hartmann 1999). As next, the best fitting solutions of the population will be selected as survivors and the rest of the solutions will be deleted. The fitness value of a solution is mainly determined based on the objectives of the optimization.

From the solutions that survived through crossover and/or mutation offspring solutions will be determined, which are then again followed by a selection process and so on. This process is usually continued for a large number of generations in order to obtain near-optimal solutions (Elbeltagi et al. 2005). The results of the GA depend on the initial size of the population, the amount of generations, the cross over rate and the mutation rate. While large populations raise the likelihood to find better fitting solutions, they also increase the processing time.

A solution for the GA is represented by a chromosome. By cross over a part of one chromosome will be overwritten or exchanged with another chromosome to generate offspring solutions. Mutation can be interpreted as an arbitrary change of the information stored in a chromosome. Mutation is useful to avoid stagnation or to exit local optima (Elbeltagi et al. 2005). In usual cases a mutation rate under 0.1 is used.

To use GA to solve the RCPSP first a suitable mapping between the task list or schedule and the chromosome representation must be found. The two main mappings or so called encoding schemes that are used are the priority-based and the permutation encoding. In the former one, the priority values of the tasks are set in the priority list. If a resource conflict occurs during scheduling the task with the higher priority will be scheduled first. In case of the permutation encoding, all the activities are permutated in a precedence feasible task list (activity list representation). The earlier the task appears at the beginning of the list the earlier it will be scheduled (Zhang et al. 2008).

Kolisch and Hartmann (1999 and 2006), Brucker et al. (1999) and Zhou et al. (2013) collected many existing GA approaches for the solution of the RCPSP or the construction optimization problem using these schemes. Some of them will also be briefly introduced here.

Hartmann (1997) introduced a GA for the RCPSP-based on the activity list representation of the project. In his work Hartmann describes three possible types of cross overs for the GA.

The *one-point cross over* takes two parents, splits them at one certain point and puts the selected parts together in order to generate offsprings (Figure 6-18). The *two-point cross over* defines two points where the chromosomes of the parents will be split and it takes e.g. the first and the third part from the first parent and the second part from the second parent or the other way around (Figure 6-19). The third, *uniform cross over* uses a sequence of random numbers to determine the origin of a chromosome component. Depending on the value of the random number the value of the component either comes from the first or from the second parent (Figure 6-20).

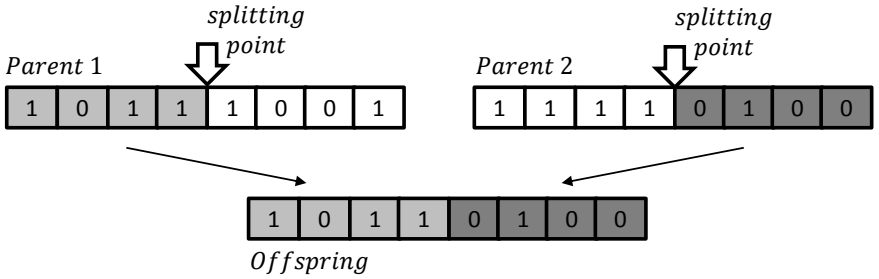


Figure 6-18: One-point cross over: chromosomes made up of eight binary components. The components of the offspring are originated before the splitting point from the first parent (light grey) and after that from the second parent (dark grey)

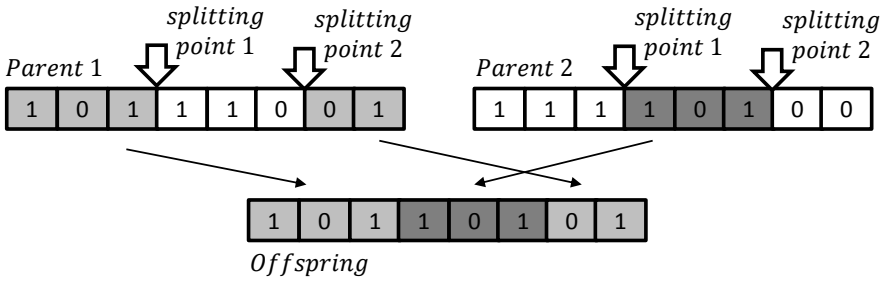
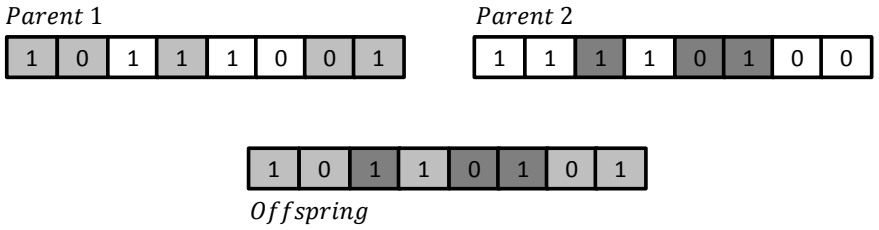


Figure 6-19: Two-point cross over: chromosomes made up of eight binary component. The components of the offspring are originated before the first splitting point and after the second one from the first parent (light grey), between them from the second parent (dark grey).



RND: {0,0,1,0,1,1,0,0}

Figure 6-20: Uniform cross over: chromosomes made up of eight binary component. The components of the offspring are originated from the first parent (light grey) if the random value (RND) is zero, from the second parent (dark grey) results the random value if it is one.

Hartmann used the two-point cross over for his calculations and compared the results with 2 further GAs and the sampling method, which had been outperformed.

Sriprasert and Dawood (2003) introduced a GA that is capable to consider multiple constraints such as "activity dependency, limited working area and resource and information readiness". The GA alter the tasks' priorities and construction methods in order to reach near optimal solutions corresponding to project duration, cost and resource utilization.

Zheng et al. (2004) developed a multi-objective GA model for the optimization of construction time-cost trade-off problems. For that they introduced the modified adaptive weight approach (MAWA) that replaces the "traditional fixed or random weights, and integrates time and total cost into a single objective for simulation". Furthermore it can efficiently exploit information of previous calculations and so it can guide the search toward the desired solution without losing randomness.

Zhang et al. (2008) introduced another effective GA to solve the single mode RCPSP. They use a *position-based cross over* method that takes components of the first parent's chromosome inserts it to the child's chromosome and then fills the missing components from the second parent keeping their relative order (Figure 6-21).

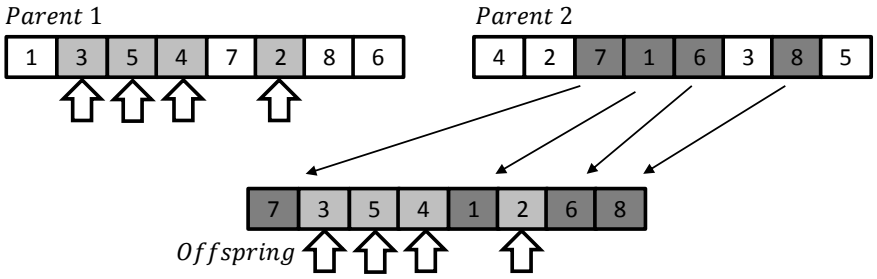


Figure 6-21: Position-based cross over: chromosomes made up of eight components that represent task IDs from a schedulable project. The selected component of the first parent (light grey) will be inserted into the offspring at the same position. The missing element will be filled up from the second parent (dark grey) keeping the sequence of the components.

The GA inherits the merits of the priority-based and the permutation encoding schemes and such reduces the search space (Zhang et al. 2008).

A two phase GA has been introduced by Chen and Weng (2009) to solve the RCPSP. In the first phase of the GA, a time-cost trade-off analysis is performed to find the execution mode for the individual tasks. In the second phase the priority-based encoding is used with the one-point cross over method to generate feasible schedules that satisfy all constraints of the project.

Montoya-Torres et al. (2010) proposed an alternative representation of the chromosomes using a multi-array object-oriented model in order to take advantage of programming features for the design of decision support systems. By comparing their results with the results of previous GAs it has been shown that this method delivers similar results in less computational time.

Szczesny et al. (2012) introduced a GA-based heuristic optimization with the CBDES for construction schedule optimization. They use the CBDES extended with the priority-based task selection to determine the next executable task in order to generate feasible construction schedules. As input data for the simulation the precedence graph, the priority list of the tasks, their resource needs and the available resources is used. In their paper a new rank-based cross over operator is introduced by Syswerda (1991) that enhances the Order Cross over-2 (OX2).

In their GA-based heuristic optimization, a chromosome describes the priority of the individual tasks and is used as input for the simulation. Therefore, the length of a chromosome corresponds to the amount of tasks within the project. The earlier the task appears in the chromosome the higher is its priority for scheduling.

In order to generate new offsprings, the new rank-based cross over operator is used that combines the chromosomes of two parents. The rank of a task is described by the maximum length of connected ancestors within the precedence graph. In the cross over, first the first parent's chromosome is copied into the child's chromosome. Then, a third or a half of the second parent's chromosome is applied also keeping the relative order of the tasks to the child's chromosome. The working mechanism of the AOX2 is based on the theory that the permutation of rank equal tasks is considered to be safe since no precedence constraints are violated. Therefore, the AOX2 selects one task randomly in the second parent's chromosome and identifies every other tasks with the same rank. Then, the relative order of these equal ranked tasks is applied to the child's chromosome. The crossover is finished when at least one third of the child chromosome is permutated (Szczesny et al. 2012).

To turn the chromosomes into feasible schedules the CBDES is used. A comparison between the crossover operator OX2 and AOX2 is presented by Szczesny et al. (2012). As a result it has been shown that AOX2 clearly outperforms OX2, however the methods should be tested on more detailed and realistic case studies.

The critical point of using GA to solve the RCPSP is the size of the population (Zhou et al. 2013). Large populations have a higher likelihood to find the desired optimal solution, however, it can greatly increase the computational effort. Small populations on the other hand might miss the optimal solution. Due to the random searching mechanism of the GA there is always a chance to find a better solution for the problem in the next generation. Therefore it is difficult to determine an accurate stopping criterion for the algorithm (Zhou et al. 2013).

Particle swarm optimization

The particle swarm optimization (PSO) technique has been developed by Kennedy and Eberhart (1995) and it was inspired by the social behavior of a "flock of migrating birds trying to reach an unknown destination" (Elbeltagi et al. 2005). Every bird in the population is concerned as a particle for the optimization. A particle is analogue to the chromosomes of the GA. In contrast to GA, the particles are not mated to create offsprings, but to discover their neighborhood by interacting with each other and heading towards their destination: the optimum.

To reach their destination every bird in a flock looks to different directions. Through communication they can identify the bird that is in the best position to reach the goal and every other bird will fly towards it with a velocity depending on its current position. Then every bird investigates its new location again and identifies the bird in the best position and so on till the desired destination is reached (Elbeltagi et al. 2005).

A PSO starts with an initial group of random particles. Every particle stores its current position, its best previous solution and the flying velocity. In every iteration step the position of the best particle is determined and the velocity of every other particles will be updated in order to catch up with the leading one (Elbeltagi et al. 2005).

The first PSO application for the RCPSP was introduced by Zhang et al. (2006). As for the GA, PSO is also important to find a suitable mapping between the particles and the RCPSP representation. Zhang et al. (2006) used the priority-based representation of the tasks as particles. Since the priority of a task can be determined on multiple ways such as critical index, duration, amount of necessary resources, amount of successors, etc. a multi-pass heuristic is proposed. There is no systematic measure that describes which sole result has the best solution according to the optimization criteria.

The multiple-pass heuristic considers diverging priority strategies and tries to propose the best fitting one for every single pass. Furthermore, a local search way is recommended that adjusts the start time of tasks based on the current multiple-results of the multi-pass method (Zhang et al. 2006), such improving the schedule. To determine a schedule from a particle a parallel transformation is used.

After a performance comparison between the GA and the PSO for a benchmark project the PSO turned out to be more efficient than the GA. Therefore, as Zhang et al. (2006) summarized: the PSO provides an "efficient and easy-to-implement" alternative to analyze the RCPSP.

Ming Lu (2008) used the Simplified Discrete Event Simulation (SDESA – see Section 3.5.6) with a PSO to automate the solution of the RCPSP. For mapping, as in the previous case, the priority-based activity representation is used. The SDESA is used to generate schedules based on the priorities of the tasks defined in the particles. After a population of schedules have been generated the PSO algorithm updates the position of every particle, identifies the best neighboring and global solution and adjusts the velocity and position of every particle (Ming Lu 2008). The method is evaluated both on a small example and a real construction project.

Chen (2011) introduced the justified particle swarm optimization technique (JPSO) to solve the RCPSP. Justification is a technique, developed by Valls et al. (2005), which adjusts the start time of each scheduled activity to further shorten its makespan. After justification it is important for JPSO to synchronize the justified position and the corresponding position vector of the particle. By comparing the results of the JPSO with the results of further, previously introduced metaheuristics the JPSO is stated as an effective and efficient algorithm to solve the RCPSP.

The general PSO method and its further developments can be summarized as effective algorithms to solve combinatorial optimization problems and with this they can solve the RCPSP. However, since it is a heuristic algorithm the obtained solution is not necessarily the global optimum. The parameters of the PSO play an important role for the analysis. According to Zhou et al. (2013), the selection of these parameters could cause convergence, divergence and oscillation of the particles. In addition the speed of the particles will decrease and such the capability to find further feasible solutions is also reduced when the variety of the population deceases (Zhou et al. 2013).

Ant colony optimization

The ant-colony optimization (ACO) involves the social behavior of ant colonies. The algorithm has been introduced by Dorigo et al. (1991) and is based on the theory that ants are able to find the shortest path between a specific destination (e.g. food source) and their nest. This is achieved through the pheromone trails. Pheromones are used as an indirect communication source between ants and they are set free every time ants travel. Other ants are capable of following this trail and they deposit further pheromones on it. The higher the density of pheromone is on a trail the higher the chance will be that other ants select the same route.

When searching for food sources ants randomly discover their environment by rotating around obstacles and depositing pheromones on all sides of the obstacle. After finding a food source the ant will randomly select a route back to the nest which is located between the available trails. The ants going to the food source and returning back to the nest will select a route and deposit further pheromones on it as they travel. The shorter the trail, the faster the ant will be for every cycle of food transport and will be able to run more often than ants with longer trails between the food source and the nest. Therefore the shorter trail will contain more pheromones and will be favored by other ants as well. Gradually, this process will lead every ant to select this shortest path.

To implement the construction scheduling problem as an ACO, the scheduling problem might be represented as a weighted network graph. The graph is the precedence graph of the tasks and the weight is the current pheromone level of the considered edge (Zhou et al. 2013). In order to start the search an initial amount of pheromones should be assigned to every edge of the network. Selection probabilities can be determined based on the pheromone level of the edges. An artificial ant is capable to travel from the first to the last task in the project following the pheromones. When the ants travel on the network the pheromone level of the edges and of the selection probabilities will be updated. The iteration of ant runs is processed until the termination criteria is reached (Zhou et al. 2013).

Fundamental research has been done on this field by Merkle et al. (2002), Ng and Zhang (2008), Afshar et al. (2009) and Duan and Liao (2010). On the one hand they have proven that the ACO is a powerful method to solve the RCPSP and on the other hand they have drawn attention to several problems of the algorithm. The premature convergence phenomenon, the termination criterion and the parameter determining methods should be analyzed further in order to enhance the ACO and gain a better application for the algorithm (Zhou et al. 2013).

6.5.2.3 Hyperheuristics

In the previous sections heuristic solution methods have been introduced for the resource-constrained project scheduling problem without completing the list. Compared to the deterministic approaches these algorithms are capable to generate feasible solutions for optimization problems in low computational time, however their results might be suboptimal. Based on these algorithms a new generation of heuristics called *hyperheuristics* was born.

Hyperheuristics are high-level approaches that control multiple low-level heuristics in order to find a near optimal solution for the optimization problem. The hyperheuristics know the strength and weaknesses of low-level heuristics and can select the most suitable one to apply for the next iteration step (Glover and Kochenberger 2003). The biggest difference between the heuristics and hyperheuristics is that the heuristics are working on the solution space of the problem, while the hyperheuristics are working on the solution space of the heuristics. With other words, depending on the current results a high-level heuristic algorithm first selects a low-level heuristic algorithm which should be applied to determine the next new solutions.

One hyperheuristic approach for the solution of the RCPSP has been introduced by Anagnostopoulos and Koulinas (2012). They applied a GRASP-hyperheuristic that controls eight low-level heuristics against ACO and GA algorithms in two case studies. The hyperheuristic resulted in a better solution than the heuristics in every case. Further hyperheuristic approaches to solve the single-mode RCPSP are not known to the author to the date of writing this thesis. However, since the hyperheuristics are able to use the advantages of all the low-level metaheuristics, further research should be carried out in order to investigate the application of this technique to the solution of the RCPSP.

6.6 Summary

In this chapter existing optimization techniques have been discussed that solve scheduling problems in the construction industry. After introducing the basic compounds and characteristics of an optimization, the combinatorial optimization problems and the complexity of a problem were discussed. The RCPSP accurately describes the scheduling problem in the construction industry and is considered as an NP-hard combinatorial optimization problem. This means that a rather complex problem cannot be solved in polynomial time.

Due to the combinatorial explosion, the size of the search space for more complex problems becomes very large. Thus the introduced deterministic solution methods for the RCPSP aim to reduce this large search space into a smaller area where the optimum can be found. Such algorithm is e.g. the Integer Programming approach or the Branch-and-Bound method which can be used to solve the RCPSP in different formulations.

Although the deterministic methods determine the optimal solution for the RCPSP, they require a high computational time. Therefore, heuristic approaches have been introduced that are capable to deliver not the optimal but one good, near optimal solution for the problem in shorter computational time. These solution approaches are based on previous experiences and intuitions.

The discussed heuristic algorithms are more suitable for the application in the industry due to the shorter computational time, than the exact solution methods. Furthermore these heuristic methods are simpler and easier to handle, than the deterministic methods. However they deliver only good solutions for the problem instead of optimal ones. Due to the continual improvement of the diverging

heuristic methods it cannot be clarified which one suits best to solve the RCPSP. All the introduced methods determine good solutions for the RCPSP with continuously improving performance. Based on the discussed metaheuristic approaches, in the next chapter three combined metaheuristic algorithms will be introduced with the constraint-based discrete event simulation for the automated solution of the RCPSP.

7 Simulation-based optimization of construction schedules

7.1 Executive summary - Optimization of construction schedules based on CBDES

As introduced in Chapter 6 the optimization of a construction schedule can be formulated as a search for the optimal schedule for the resource-constrained project scheduling problem (RCPSP). The RCPSP is an NP-hard combinatorial optimization problem, which in most cases has such a large search space of discrete solutions that all of them cannot be determined in polynomial time.

In Chapter 4 the constraint-based discrete event simulation (CBDES) that is capable of determining single feasible schedules for the RCPSP was introduced. With a suitable steering algorithm that manipulates the input data for the simulation, the CBDES could be used to generate various feasible solutions with multiple simulation runs. Furthermore, by using one optimization method as a steering algorithm for the simulation it can be used not only to generate various schedules, but also for optimization purposes.

Therefore an algorithm has been developed within the framework of this thesis that varies the input data for the simulation in a systematic manner so that the simulation generates various feasible schedules in order to find a near optimal schedule. Such methods for the steering of the CBDES were initially introduced by König and Beißert (2009) (see Section 7.2) and Szczesny et al. (2012) (see Section 6.5.2.2 – Genetic algorithms). The common thread in both of the existing methods is that they manipulate the priority of the single tasks to influence their execution order during the simulation and thus within the generated schedule. The drawback of these methods is that there is a chance that after manipulating the tasks' priority the generated schedule is the same as it was before the manipulation. The reasons for this result will be discussed in Section 7.2 in detail. One of the author's goals was to

develop an algorithm that is capable of manipulating the priority of the tasks so that after the order of the tasks has been changed, a new schedule is produced (see Section 7.3).

The working mechanism of the algorithm is similar to that of the disjunctive graph model for the shop problems (see Section 2.3.2). In this case, however, it is not the direction of the disjunctive arcs that are swapped but rather the priorities of specific tasks. The disjunctive graph model itself cannot be applied for the RCPSP since for the completion of a task in the RCPSP more than one resource can be applied and tasks that require the same resource class for their execution can also be executed simultaneously. Therefore an exact predefined sequence of tasks, such as the fixed arcs of the disjunctive graph, is inapplicable. Thus the usage of priorities instead of further precedence constraints allows a more flexible selection of the tasks. Priorities define the execution order of the tasks. When the necessary resources are available the affected tasks can also be executed simultaneously. Sections 7.2 and 7.3 address changing the priorities of tasks to generate new schedules with the constraint-based discrete event simulation. Section 7.4 discusses how the introduced algorithm influences the search space of the possible solutions for the RCPSP.

The application of this algorithm however ensures the generation of various schedules with the CBDES. It is still not an optimization tool. This algorithm has been coupled with different methods to improve the generated results of the simulation. For the optimization the author applied three heuristic optimization methods – greedy-like, simulated annealing and tabu search algorithm. How best to use these methods to generate near optimal schedules with the CBDES will be discussed in Section 7.5. These specific methods were selected due to their short computational time and simple working mechanism. The chapter is closed by two comprehensive case studies and a summary.

7.1.1 Limitation of the constraint-based discrete event simulation

As introduced in Chapter 4, the constraint-based discrete event simulation (CBDES) is able to generate feasible solutions for the resource-constrained project scheduling problem (RCPSP). To obtain the solutions it uses the described start and end event functions. The start event checks for which tasks are all precedence constraints fulfilled. If one task is found and the necessary resources are available then the task will be executed. Every task will be executed as early as possible taking precedence and resource constraints into account. However, executing a task as early as possible within a schedule is not always a good strategy. Sometimes delaying a task might deliver a better result. Consider, for example the four tasks depicted on the precedence graph in Figure 7-1. Tasks A and B use resource “gray”, and tasks B and C use resource “white”. Only one resource is available in both resource classes. Due to the constraint configuration, with the CBDES task B will always be executed before task C, because at the point in time when B could be started, C still has a non-fulfilled precedence constraint: A. Since the CBDES requires that every task be executed as early as possible, there will be no schedule where B is executed after C (Figure 7-1 – 2. schedule). This is due to the time increment-based scheduling generation scheme of the CBDES (introduced in Section 4.2.2).

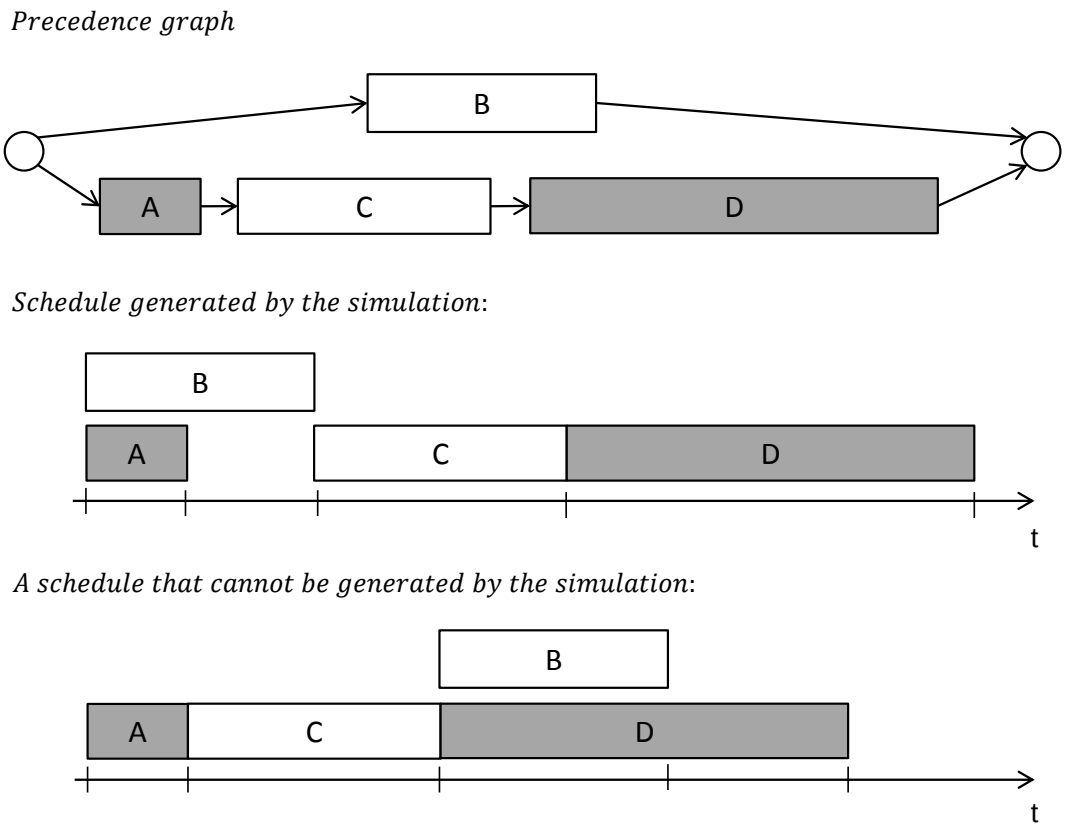


Figure 7-1: Limitation of the constraint-based discrete event simulation: In some cases the complete set of results does not contain the optimal schedule (minimal makespan). White and grey: two resource classes.

The algorithms that the CBDES applies to generate feasible schedules belongs to the class of parallel scheduling (time increment-based) methods, which has been proven to generate only *non-delay schedules*[32] (Kolisch 1996, Hartmann and Kolisch 2000). As it has been proven that the optimal solution for the RCPSP might be outside of the non-delay schedules, it is also possible that the results of the CBDES do not contain the actual optimal schedules (as presented in Figure 7-1). Hence a deterministic optimization strategy using the CBDES shall not be considered. However, this scheduling method is useful for heuristic optimization techniques, where a near optimum or good solution is the goal, rather than the optimal solution.

7.2 The priority swap of tasks

In order to find the optimal or a "good" solution for an optimization problem a very large number of feasible solutions must be determined and taken into account. For the RCPSP the constraint-based discrete event simulation is a method that is able to determine single feasible schedules. To determine various schedules for the scheduling problem in which the technological dependencies and the resource configurations are fixed, the execution order of the tasks within the schedule must be manipulated. A simple example of how various schedules with different makespan can be determined

[32] See Figure 2-2 in Section 2.2.2.1.

by swapping tasks within the schedule is presented in Figure 7-2 and Figure 7-3. The project is made of nine tasks (A-I) and requires two different classes of resources (dark and light grey). There is only one unit for both resource classes available on the site.

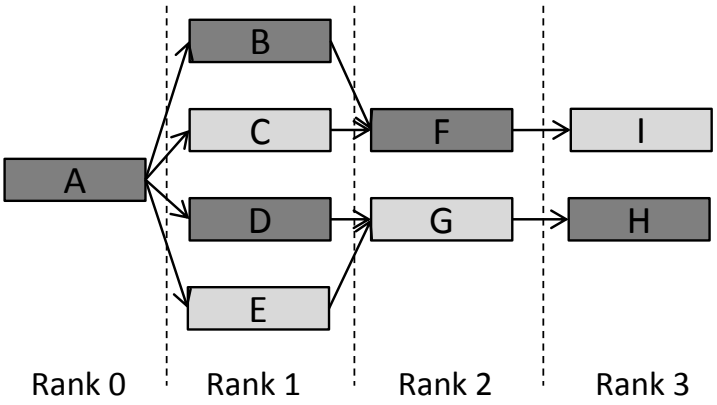


Figure 7-2: Dependency graph and ranking of an example project with two resource classes (dark and light grey)

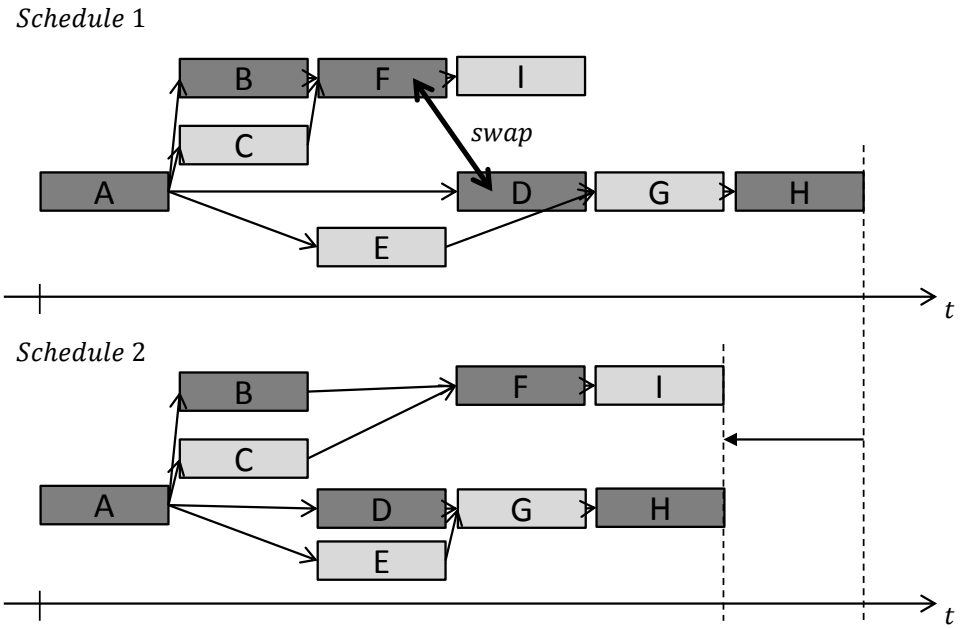


Figure 7-3: Resulting different schedules by swapping the position of task D and F within the schedule

With the constraint-based discrete event simulation the determined order of the tasks within the schedule is ruled by the predefined precedence constraints (technological dependencies) and the priorities of the tasks. Since the precedence relationships of the project must be kept fixed and newly introduced precedence constraints would prevent the possible simultaneous execution of the corresponding tasks, one promising approach to introduce a change into the execution sequence of tasks is to manipulate the priorities of the tasks. To generate one base schedule, like schedule 1 in Figure 7-3, an arbitrarily distributed priorities might be applied for the single tasks. However, it is important that none of the tasks should have identical priorities, otherwise a priority swap between

tasks might have no effect on the result. Thus, swapping the priorities of task D and F in schedule 1 will have the result presented as schedule 2. As this example shows, a good selected priority swap can be used to determine a new schedules. However, a randomly applied priority swap is not a way to consider. For example, a priority swap of task A-H or C-I will not change anything on the schedule since they are either precedence related or they do not use the same resource. Therefore further rules must be defined which can identify the swaps that can cause a change in the schedule. Deciding which swaps to make is addressed in the following sections. The results that are determined by applying one of these swaps are defined as neighborhood solutions for the optimization problem.

The feasible search space of the RCPSP can become very large, and many different solutions must be taken into account. Therefore, in order to find the optimal or near optimal solution in an efficient way, only the swaps that will certainly generate new schedules should be applied. Thus, the development of an algorithm that is able to cover all of the feasible solutions of the search space is required. This algorithm must avoid the generation of repeated schedules and must be able to traverse through the search space by steering the simulation in such a way that it generates a new schedule with every single swap.

Therefore, as it has been already discussed, applying a random swap-based algorithm to steer the simulation to find the optimal or good solutions for the RCPSP is not a way to consider. A more sophisticated approach is the rank-based swap of tasks within the execution list introduced by König and Beißert (2009). The rank of the task is described by the maximum length of the chain of connected predecessors from the start until the end of the examined task (Figure 7-2). Since this approach allows for the execution of a swap only between tasks with the same rank, the amount of swaps that do not change the schedule is less than in the completely random case, but is still not zero.

This is because this algorithm is not taking the resource classes into account. In a situation where more of the tasks that need the same resources for their execution could be started at the same point in time but where the available amount of resources do not allow the simultaneous execution of these tasks, the tasks with the highest priority will be executed. In this example, the tasks with lower priority must wait until the resources are once again available. By swapping the priority of two of the tasks, a new execution order might be generated. Therefore a priority swap must always be defined between tasks that use the same resource classes. Swapping the priorities of two tasks that do not use the same resource classes will not directly affect their position in the queue. For example, swapping the priority of tasks B and C, or D and E (all rank 1) will not affect the generated schedule and the result will be identical to schedule 1.

Another drawback of allowing swaps only between tasks with the same rank is that swaps which could introduce changes into the schedule are neglected. To illustrate, consider the project presented in Figure 7-2. A swap between D-F or C-G is not allowed with the introduced rank-based method since these tasks have different ranks within the graph. Although, as it has been presented in Figure 7-3, these tasks can actually change the result of the schedule.

To make the swapping approach of task priorities more efficient a new methodology has been developed that can determine *every possible swap* that can result in a change in the schedule while also taking into account the different resource classes.

7.3 The possible and the reasonable swaps

As described in the last section, it is important that the tasks whose priorities are to be swapped use the same resources. Additionally, they should build an independent or disjunctive pair of tasks, otherwise their execution order would already have been defined by precedence constraints (see Section 5.3). The pair of tasks that satisfies these criteria create the list of *possible swaps*. This list includes all of the swaps that could be applied to a schedule in order to introduce a change.

To determine every possible swap the fragmented dependency matrices of the precedence graph will be used (see Section 5.4). The fragmented dependency matrices describe exactly those tasks which use the same resources and which are independent from each other (zero element in the matrix). Therefore the amount of possible swaps is the sum of the zero elements within the different transitive closures for the resource classes divided by two, since the matrixes are symmetric and swapping the priority of task A with task B is the same as swapping the priority of task B with task A. Swapping priorities of dependent processes (transitive closure value 1) is pointless because they have a predefined and strict order in the graph and generated schedule which cannot be changed. The dependency graph of the introduced example is presented in Figure 7-4 (already presented in Section 5.4 and in Figure 5-12).

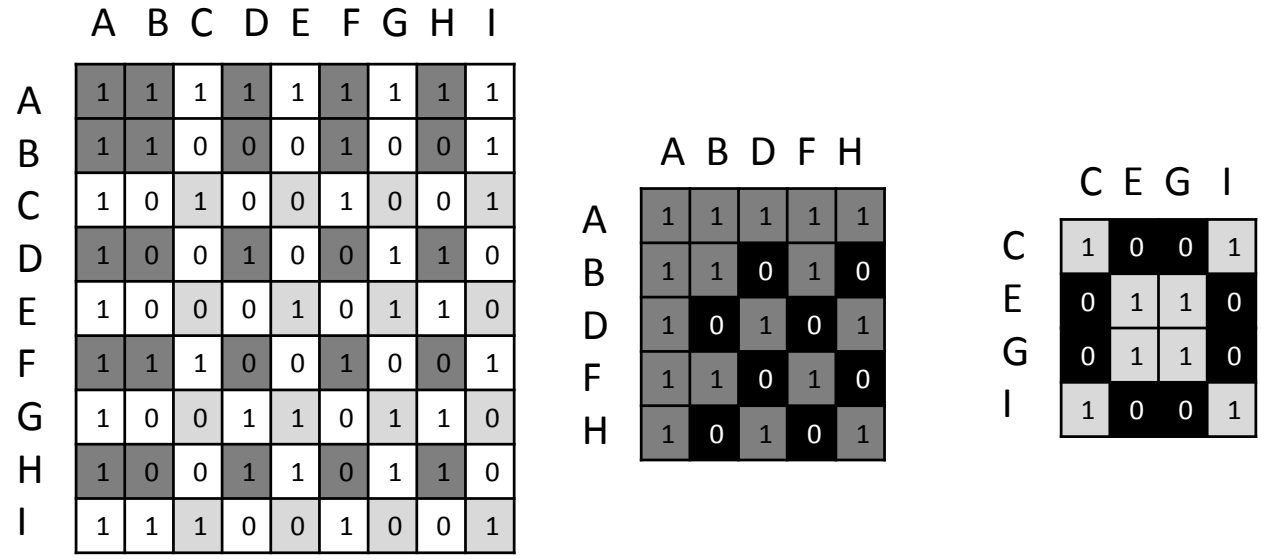


Figure 7-4: The dependency matrix and the fragmented dependency matrices for the different resource classes (dark and light gray). Black elements with white text: independent pair of tasks

However, applying a possible swap to a schedule with a restricted resource configuration will not always have an effect on the newly generated schedule. This occurs when one of the tasks which could be swapped is executed at the beginning of the schedule and the other one is executed at the end. This will not change anything in the schedule because there are other tasks between the examined tasks using the same resources and so their position within the schedule cannot be swapped.

The swaps which will change the schedule are called *reasonable swaps*. Reasonable swaps are a subset of the possible swaps. The amount of reasonable swaps varies between zero and the amount of possible swaps depending on the resource utilization of the project.

To determine the list of reasonable swaps, first a feasible schedule must be generated. This is important to find out how the execution sequence of the tasks is alike after resource constraints and limits are considered. The next step of a reasonable swap determination is to apply an algorithm that identifies all of the possible swap candidates for every single task within the schedule using the fragmented dependency matrices of the project. This results in a list of possible tasks with which the considered task could be swapped. The reasonable swap candidates for the task will originate from this list. To determine which tasks qualify as reasonable swap candidates, a time frame will be defined within which this candidate task must be started. This is in addition to the resource and precedence relationship restrictions that have already been checked by the dependency matrix. Should a task not start in the required time frame, it will be deleted from the list of reasonable swaps.

The time frame is defined by the latest termination date of the tasks predecessors and the start date of the task itself (Figure 7-5). This time frame is important because the examined task cannot be started earlier than its predecessors' finish date. Therefore it is useless to swap priority with another task that starts earlier than this date, since the other task will always be executed earlier than the examined task. Furthermore, when there is a possibility to swap the task's priority with another task that starts later than the task itself, it will be determined during the search for possible swaps conducted on behalf of the other task.

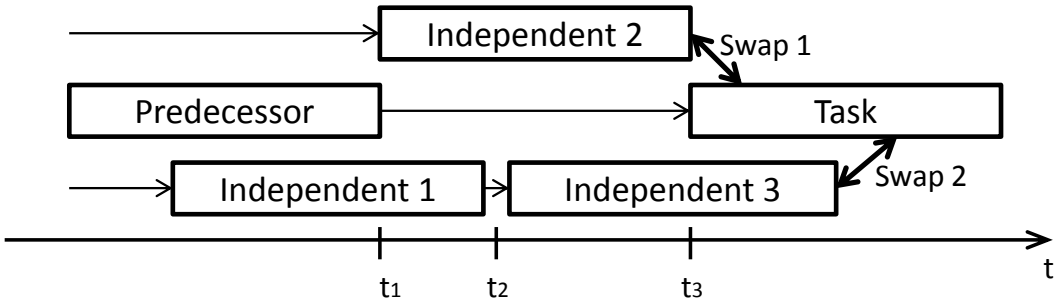


Figure 7-5: Schematic illustration of the reasonable swap determination: the list of possible swaps of independent tasks is restricted by the ones that start between the latest finish date of the task's predecessors (t1) and the task itself (t3). Priorities: Predecessor: 10, Independent 1: 8, Independent 2: 6, Independent 3: 4, Task: 2. Reasonable swaps: Task and Independent 2; Task and Independent 3.

When the reasonable swaps for every task in a particular schedule have been determined, applying any of them will result in a different schedule than the previous one. Any other swaps will not directly change the results. An indirect effect from a random swap is possible when the relation in any of the reasonable swaps is changed. For example, swapping the priority of the considered task and its predecessor in Figure 7-5 will have no direct effect on the results. However, if the priority of the predecessor was higher than the one of the task "independent 2", then after the swap the considered task will have a higher priority than "independent 2" and therefore it will be executed right after its predecessor (Figure 7-6).

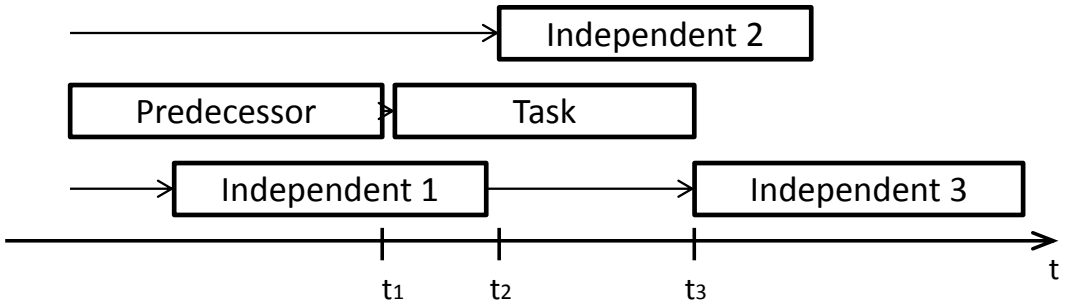


Figure 7-6: New partial schedule after swapping the priorities of the Task and its Predecessor. New priorities: Predecessor: 2, Independent 1: 8, Independent 2: 6, Independent 3: 4, Task: 10. Indirect effect: the priority relation of the Task and Independent 2 changed.

A random priority swap that changes the order of tasks resulting in a new schedule can always be traced back to a relation change in one or more of the reasonable swaps. Therefore, in order to generate new, different schedules the author only uses the reasonable swaps. Applying one reasonable priority swap for the project results in a new schedule, which is defined as a neighbor solution of the base schedule. Therefore, the distance between the two different schedules in the search space can be

defined as the amount of reasonable swaps that are necessary to reach one schedule from the other one.

7.4 The effect of limited resources on the size of the search space and the amount of reasonable swaps

Using the technique described above, the complete solution space of schedules that can be generated with the constraint-based discrete event simulation for the schedulable project with predefined precedence relationships and resource configurations, can be covered.

However, the number of possible schedules and the number of reasonable swaps is highly dependent on the resource utilization of the project. When the available number of resources is high enough and only one schedule can be determined, it is similar to the results of the CPM and no reasonable swaps are available. This is due to the fact that every task has been started at its earliest possible start time and no variety can be introduced to the schedule. Lowering the amount of available resources for the project means that an increasingly fewer number of tasks can be executed simultaneously, causing them to have to be delayed due to the lack of necessary resources. Depending on which tasks are delayed the number of possible schedules increases.

When a task is delayed the gap between the task's start date and its predecessor's finish date becomes larger and larger. This leads to a growing time frame for the task where reasonable swaps can be searched. Therefore, in decreasing the amount of available resources, the number of possible schedules and the amount of reasonable swaps will increase. However, this number is limited by the amount of all possible swaps.

Essentially, reasonable swaps can be viewed as the decisions the scheduler must make in order to stay under the predefined limit of resources (e.g. task A and B could have been executed simultaneously but one of them must be pushed forward in time due to the lack of resources). Therefore, this technique is not only an efficient method to determine neighborhood solutions for a schedule, but it is also useful to analyze existing schedules and identify bottlenecks within them. A reasonable swap illustrates a decision made during the scheduling process. Increasing the available resources at this date results in eliminating the identified swap and thus the concerned tasks can be executed simultaneously, shortening the duration of the project on minimal costs.

The advantage of the introduced technique is its capability to identify every reasonable swap within a schedule, thus the neighboring solutions can be generated efficiently without duplicating any of the existing schedules. The next section will discuss how the reasonable swaps can then be used to find near optimal solutions for the RCPSP.

7.5 Different optimization strategies to find near optimal solutions for the RCPSP using task priority swaps

As introduced in Chapter 4, the constraint-based discrete event simulation is capable of generating single feasible solutions for the RCPSP. Using the reasonable swaps introduced above, neighborhood solutions for one base schedule can be generated. The RCPSP's objective is to find the one schedule with the minimal makespan. Since the search space of the RCPSP can become large, an algorithm that can efficiently traverse through the search space jumping from one solution to another is necessary to search for the schedule with the shortest makespan.

One solution is to use an enumeration tree. The randomly generated base schedule, which should be optimized, is placed at the top. Every branch of the tree symbolizes one applied reasonable swap and results in a neighborhood solution. Since the branching process is independent from the fitness function of the optimization, the same model can be used for different optimization criteria (e.g. duration, cost, robustness etc.).

Before introducing the functionality of such an enumeration tree, we must determine how much the application of the reasonable swaps can accelerate the search process for the optimal solution as opposed to a random possible swap.

The most general approach to determining the optimal solution for the RCPSP would be the total enumeration of the problem. This means generating a schedule by simulation for every permutation of the possible swaps. In this case the number of generated schedules equals the sum of all permutations selecting k swaps of n distinct possible swaps:

$$\sum_{k=0}^{n} nPk = n! * \sum_{k=0}^{n} \frac{1}{k!}$$

So far this sum is independent from the number of tasks within the schedule. Therefore, when t equals the number of tasks in the schedule, the total number of possible swaps is limited from above with the sum of the arithmetic progression from 1 to t-1:

$$\frac{\big((t-1)+1\big) * (t-1)}{2} = \frac{t^2 - t}{2}$$

This value represents the number of elements within the half triangle of the dependency matrix, which stands for the maximal total number of possible swaps when every task needs the same resource and there are no precedence relationships defined between the tasks. In this case every element of the dependency matrix except for the diagonal will be zero. Every single introduced precedence relationship will decrease this number exponentially. Thus, the number of generated schedules using this swap technique is limited to:

$$n! * \sum_{k=0}^{n} \frac{1}{k!} \leq \left(\frac{t^2 - t}{2}\right)! * \sum_{k=0}^{\frac{t^2-t}{2}} \frac{1}{k!}$$

Taking the introduced precedence graph in Figure 5-10 (Section 5.4) as an example where $t = 9$ and $n = 8$, the total number of solutions are:

$$(8! * \sum_{k=0}^{8} \frac{1}{k!} = 109601) \leq (36! * \sum_{k=0}^{36} \frac{1}{k!} \approx 1.01 * 10^{42})$$

By only using the possible swaps instead of swapping the priority of every task with every other task it is apparent how many of the possible schedules can be reduced. Because of the factorial relationship between the number of swaps (n) and the number of all possible schedules, increasing the number of possible swaps causes the number of possible schedules to explode. For the above example with 8 swaps using an average calculation time of 25 milliseconds, it would take 45.6 minutes to identify every possible schedule. If you increase the number of possible swaps by just one further swap, the total calculation time will be 6.85 hours. For 10 possible swaps the amount of possible schedules will be 9864101 and the total calculation time will be 68.5 hours.

Every further swap will multiply the number of possible schedules and so the calculation time will increase as well. As presented earlier, the list of possible swaps also contains swaps that will not change the schedule, so the duplication rate of the schedules is higher than zero. Therefore, to reduce the calculation time and the number of duplicated schedules instead using all the possible swaps, just the reasonable swaps should be used.

However, since the amount of possible schedules and reasonable swaps is dependent on the resource utilization of the project, the exact amount of possible schedules cannot be foreseen. Although the amount of diverging schedules is limited from above with the amount of schedules for all the possible swaps, and from below when the results are equal with the CPM schedule: one. When the amount of different schedules according to reasonable swaps is "m", then:

$$1 \leq m \leq n! * \sum_{k=0}^{n} \frac{1}{k!}$$

Considering, for example, a bridge construction project with approximately 300 tasks. Due to the amount of precedence constraints let us assume that the number of the possible swaps is approximately 100. Using the introduced total enumeration method the determination of the optimal schedule would take years. A more sophisticated method is required that is capable of reducing this calculation time into an acceptable range.

Since the constraint-based discrete event simulation is only capable to generate non-delay schedules for the RCPSP (see Section 7.1.1), it is possible that the actual optimal schedule cannot even be generated by using this method. A deterministic optimization strategy should not be considered until the CBDES has been extended to cover the complete search space of active schedules. However, the CBDES supported by an efficient heuristic algorithm could deliver not necessarily the optimal result, but one good, near optimal solution in a short calculation time. For that, an enumeration tree-based approach has been developed. The enumeration tree represents all the possible solutions the CBDES could generate, with a randomly generated schedule at the top. To reduce the search space, the nodes of the enumeration tree will only be identified where it is probably profitable to do so. To determine near optimal solutions for the RCPSP with the enumeration tree two basic meta-heuristic approaches have been implemented and compared to each other. These two heuristic methods that traverse by way of different paths through the enumeration tree and steer the simulation in different ways will be introduced in the next sections.

7.5.1 The enumeration tree

In order to collect and order the solutions of the constraint-based discrete event simulation for the heuristic optimization of the RCPSP an enumeration tree will be used (introduced in Section 6.5.1.2). The working mechanism of the calculation is based on the principles of the Branch-and-Bound method. The similarities are that both methods try to reduce the large search space by pruning branches from the tree. The main difference is that while the Branch-and-Bound is a deterministic method that only prunes branches of the tree that clearly do not contain the optimum result, our approach is a heuristic method that prunes away branches even when the probability of finding a better solution than the current optimum on the branch at issue is low.

Every node of the enumeration tree represents one feasible schedule. On the top of the enumeration tree is a randomly generated schedule, which shall be enhanced. The schedules on the next level will be generated by applying one of the reasonable swaps from the schedule. Therefore

the schedules on the second level are the neighbor schedules of the base schedule, the ones on the third level are the neighbor schedules of the ones on the second level, and so on.

In order to decide which nodes of the enumeration tree should be determined two meta-heuristic algorithms have been applied. A third strategy will also be introduced based on the experiences with the former algorithms. The difficulty of the reasonable swap-based heuristic optimization is that the result of a swap cannot be predicted. Applying a swap might result in a better schedule but it could also lead to a worse one. It is also possible that applying one or two swaps at first leads to slightly worse results initially but with the third swap the result will become better again (Figure 7-7). This corresponds to the optimization theory that in order to leave local optima, worse results should also be accepted (see Section 6.5.2). Therefore, the applied optimization algorithms have been developed such that they are able to take this behavior into account.

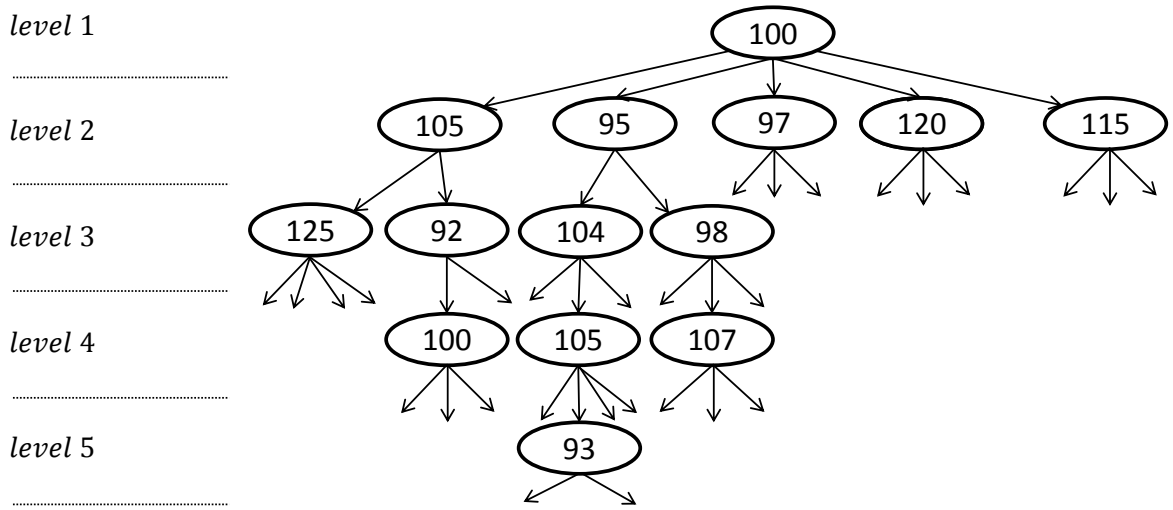


Figure 7-7: Schematic representation of a part of the enumeration tree. Every node symbolizes one schedule. On the top is level 1 representing a randomly generated base schedule. The number in the nodes represents the makespan of the schedule. One arrow accords to one applied reasonable priority swap.

The following sequence illustrate the general working mechanism of the enumeration tree:

Step 1. Generate a random base schedule (root node)
Step 2. Determine every reasonable swap for the base schedule and by applying them one-by-one determine the child nodes
Step 3. Evaluate the generated nodes. Fathom all nodes where the makespan of the considered schedule is worse than the makespan of the current best solution extended with a predefined tolerance factor
Step 4. Determine the child nodes of the non-fathomed nodes
Step 5. Repeat 3-4 till there are no further nodes to consider or a predefined amount of levels reached.

The first discussed strategy is a greedy-like optimization where only those nodes of the tree that are within a predefined tolerance range of the current best schedule will be considered further. The second strategy is based on the simulated annealing strategy. In this case the more schedules that have been determined, the lower this tolerance will be set. The exact working mechanism of these methods

will be discussed in the corresponding sections. Furthermore, these heuristic methods have been enhanced with a tabu search algorithm that forbids a re-swap of previously swapped priorities, e.g. after swapping the priority of task D and F, a child node that is created by swapping back the priorities of task F and D is forbidden. This would create a loop inside the tree since the children of the re-swapped node are the same as for the node two levels higher thus exploding the search space (see Figure 7-8). The introduction of the developed heuristic optimization methods is followed by two case studies and the comparison of the effectiveness of the techniques.

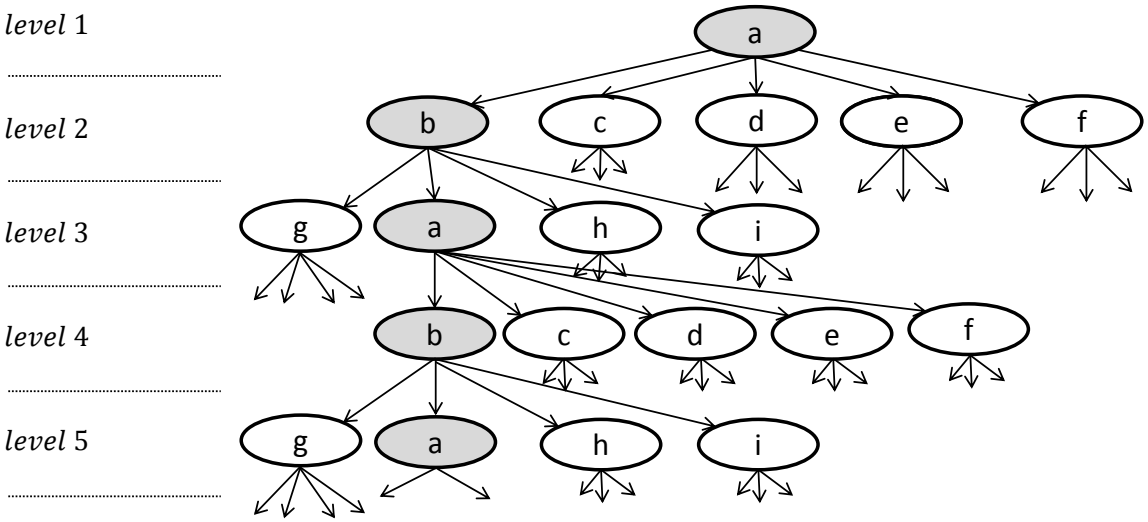


Figure 7-8: The effect of re-swapping: a loop inside the enumeration tree. The child nodes of *node a* and *node b* are determined at every 2nd level. The nodes represent a feasible schedule. The re-swapping is marked with gray node-background. To avoid this behavior a tabu search algorithm is applied that forbids to re-swapping of previously swapped priorities.

7.5.2 Greedy-like heuristic approach

Since the search space of the RCPSP can become large depending on the size of the problem, the number of possible schedules also becomes high. In order to find one good, near optimal solution in a time efficient way, many of these solutions, which have a high likelihood of not representing a good solution, should not be determined. However, to decide which schedule to neglect and which to explore is a complex task. As mentioned in the previous section, the result of a priority swap cannot be foreseen since it can either result in a schedule with a shorter or with a longer makespan. Hence, an exact algorithm, such as the lower and upper bunds for the Branch-and-Bound method (see Section 6.5.1.2), to determine which nodes should not be considered any further cannot be formulated.

However, to reduce the amount of schedules that should be determined, a tolerance factor has been introduced. It represents a limit for the makespan of the schedules. The tolerance factor is either given as a percentage or in the form of a time interval and describes how much longer the makespan of a schedule is allowed to be than the current minimal makespan in order to be considered for further calculations. The value of an accurate tolerance factor depends on the makespan of the project and the actual duration of the single tasks. As discussed at the presented case studies in Section 7.5.4, for

smaller problems with the makespan of about 100 time instances and tasks with an average duration of 4-5 time instances a tolerance factor of approximately 10% shall be considered as a good start value. For larger projects with a makespan of approximately 1000 time instances and tasks with an average duration of 10-20 time instances the tolerance factor shall be lowered to 1-5%.

When the makespan of a schedule is longer than it is allowed to be as defined by the tolerance, then the likelihood that the schedule will deliver an outstanding schedule later is low. Therefore it will not be considered for further calculations. To determine the ratio of the considered schedule and the current schedule with minimal makespan, the function f shall be used.

$$f = \left\{ \frac{makespan}{current\ minimum\ makespan} \right\}$$

When $1 < f < T$, the node will be used for further calculations and its neighborhood solutions will be determined. If $f > T$ the node will not be considered for further calculations.

The working mechanism of the greedy-like heuristic is the same as discussed in the last section for the general case. First, a base schedule must be generated. This forms the node on the highest level. Next, this node will be selected to create its neighborhood solutions (child nodes), which will construct level 2. To create level 2, all the reasonable swaps of the node will be determined using the dependency matrix of the project and the formulas defined in Section 7.3. When the list of reasonable swaps is complete, one priority swap will be applied and a new schedule will be created by the constraint-based discrete event simulation. This result builds a new branch of the enumeration tree and a new node on the next level. This process is repeated until a schedule for all the reasonable swaps has been determined. Once they are all complete they will be evaluated based on their own makespan and the makespan of the current shortest schedule. When a node does not meet the criteria of the tolerance factor, the node will be discarded. Next, the nodes of the second level that were not fathomed will be selected and their neighborhood solution will be determined. The functionality of the method is presented on a simple fictitious example with a manually created enumeration tree in Figure 7-9.

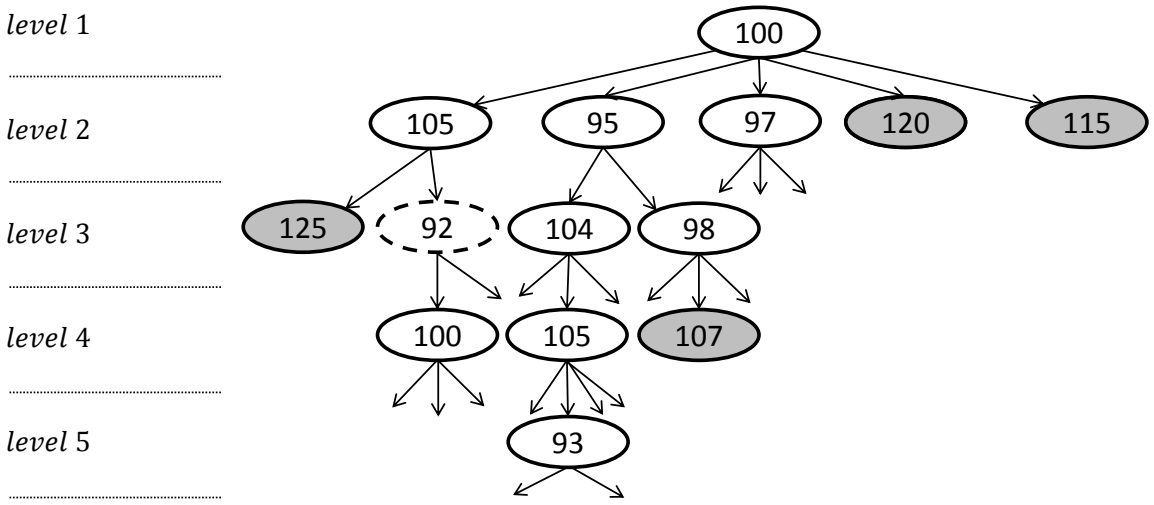


Figure 7-9: Schematic representation of the greedy-like heuristic algorithm. Grey: Fathomed nodes, dashed ellipse: current best solution, arrows: applied swaps. Tolerance factor: 15%.

After generating the base schedule with the duration of 100 time instances, all reasonable swaps will be applied and the nodes of level 2 will be generated. On level 2 the tolerance factor is set to 15% and the current best solution is 95 time instances, leading to the outcome that every schedule on this level that lasts longer than 109 time instances will be identified (120 and 115). For the nodes that were not identified (105, 95 and 97), their reasonable swaps will be determined and applied. On level 3 a new current best solution has been found with 92 time instances, therefore the new tolerance limit for schedules is 106 time instances. The search continues until there are no new nodes available or until a predefined amount of levels is reached.

In order to reduce the amount of repeated schedules in the enumeration tree, the swap that has been applied to get the considered node is prohibited for the determination of the next child nodes. This avoids the situation where after executing a swap at the next level it will be re-swapped again. Further investigation on repeated schedules within the enumeration tree has been made by Bügler and Borrmann 2014. They pointed out that a swap should be prohibited until another swap has been made either before the temporal position of the concerned swap or until one of the involved tasks has been used by another swap.

Using the above described tolerance factor for the optimization the size of the search space can be varied. Setting the tolerance to 15%, for example only the schedules under the makespan relation value of 1.15 will be considered further (as depicted in Figure 7-9). At the first level of this example only the schedules under the duration of 115 time instances will be selected for further consideration when the actual minimum project duration is fixed to 100 time instances. If the tolerance values are set too low the results may quickly get stuck in a local optimum. If the tolerance values are set too high, the search space and the calculation time will explode. To determine some general rules on how to select an accurate tolerance factor a parametric case study will be presented in Section 7.5.4.

The completely greedy variation of the technique is when the tolerance value is set to 0%. This algorithm only considers the best solution of every level as long as it is improved. When no improvement can be found the algorithm terminates from itself.

7.5.3 Simulated annealing-based heuristic approach

The working mechanism of the simulated annealing-based approach is similar to that of the greedy-like algorithm. The idea behind the simulated annealing is to define a higher bound for the results that are accepted. With every calculation step lower this higher bound till no better results can be found. This method due to the high bound for the results has a good chance to exit local optima. This is also important for the reasonable swap-based heuristic optimization, since a good result can be preceded with some worse solutions within the enumeration tree. In this case the higher bound of the algorithm is represented by the tolerance factor introduced in the last section, which will be sink by every new level of the tree.

The functionality of the reasonable swaps-based simulated annealing algorithm is the follows:

1. Generate base schedule and initialize tolerance factor
2. Select the next not yet considered level
3. Select a not yet considered and not fathomed node on the selected level
4. Determine every reasonable swap for the selected node
5. Apply one reasonable swap, determine the schedule and create a new branch for the enumeration tree
6. Repeat 5. till all the reasonable swaps has been applied for the node

7. Evaluate the generated nodes. Fathom all nodes where $f > T$, or where the makespan is larger than the makespan of the current best solution extended with the tolerance factor
8. Repeat 3-7 till there are no further available nodes on the considered level
9. Lower the tolerance (T)
10. Jump back to 2. or terminate the calculation if the predefined amount of levels is reached

Similar to the greedy-like algorithm the simulated annealing algorithm works with an enumeration tree and a tolerance factor. However, they differ in how the tolerance factor is handled during the optimization. While for the greedy-like algorithm the tolerance factor is fixed to one value, the tolerance factor for the simulated annealing case changes dynamically. It is advised to start the simulated annealing with a higher initialized tolerance than for the greedy-like method, since it will be sunk with every generated level. The higher initial tolerance also means a wider start field for the simulated annealing which will be narrowed later on by lowering the tolerance factor. The lowest limit of the tolerance factor is 0% which would mean that equally good results or better ones will be accepted by the algorithm for further calculation. When none of the nodes satisfy such criteria the optimization will be terminated and the best solution found will be delivered to the planner.

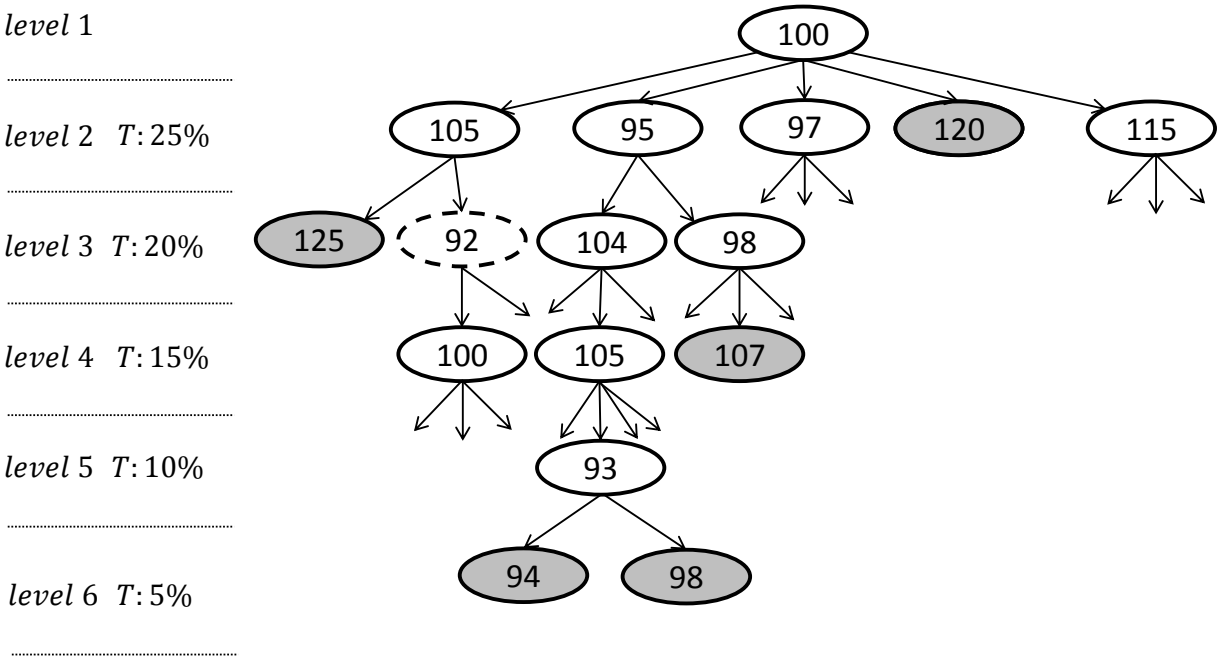


Figure 7-10: Schematic representation of the working mechanism of the simulated annealing algorithm to traverse through the solutions of the enumeration tree. Grey node: fathomed, dashed ellipse: current best solution, arrows: applied swaps, T: tolerance factor at the corresponding level

Figure 7-10 represents the application of the simulated annealing algorithm for the introduced simple fictitious partial enumeration tree example extended by one further level. The initial tolerance has been set to 25%, being higher than the one for the greedy-like algorithm. After the generated base schedule and the applied reasonable swaps, a new current best node has been found with 95 time instances. Therefore in this example, only the node with the duration of 120 time instances will be fathomed. The next step in to sink the tolerance factor to 20%. The reasonable swaps of the not fathomed nodes on level one will be determined and applied one by one. The current best solution will be updated to 92 time instances and the nodes that do not meet the tolerance criterion will be fathomed. The calculation is continued as long as non-fathomed nodes are available on the next level

or until a predefined amount of levels has been reached. In the presented case every node on level six in the partial enumeration tree is fathomed and so the calculation terminates.

7.5.4 Depth oriented heuristic search algorithm

As discussed in Section 7.5.1, the previously introduced heuristic optimization methods are enhanced with a tabu search algorithm. This enhancement was put in place to avoid the re-swapping of task priorities for the child nodes which would lead to a loop inside the enumeration tree. Although the tabu search algorithm avoids the repeated generation of the same schedule on the same branch, the same schedule can still be generated on different branches of the tree. When this schedule is identified as good solution and is not fathomed, then the child nodes of these nodes will be identical and thus every child node will be determined multiple times. No exact algorithm has yet been developed which avoids this duplication. A solution to reduce the amount of identical determined schedules on different branches is to use enumeration trees that determine results only a few levels deep (about 4-6, depending on the size of the problem) and then restart the optimization with the current best solution as the root node. Through a predefined number of levels the identically determined schedules will be ignored and the optimization continues with only the current best solution. Since this algorithm is able to reach deep levels of the enumeration tree, the author has identified this method as a depth oriented heuristic search algorithm. Similar conclusions were drawn also by Bügler and Borrmann (2014). Since the simulated annealing method can restrict the number of determined levels by lowering the tolerance factor, the application of the depth oriented method coupled with the introduced simulated annealing-based heuristic approach is recommended. The application of this method is necessary for problems that are made of more than 40-50 tasks. If there are fewer tasks the effect of identical generated schedules on the determination process is acceptable. Hence the application of this method will be introduced for a larger case study in Section 7.6.2. A comparison between the introduced heuristic optimization techniques executed on a real bridge construction project will be presented in the next section.

7.6 Case Studies

7.6.1 First case study

To test the applicability of the first two heuristic optimization strategies introduced above, a small case study has been conducted using the simple bridge project example introduced in Section 5.5. The project is made up of 34 tasks which require one of three kinds of resource classes: “Armouring”, “Casing” and “Concreting”. In order to obtain more alternative solutions, the amount of available resources compared has been lowered from the amount used in the test case in Section 5.5. The unit resources for this problem are: casing: 1, armouring: 2 and concreting: 1. The priorities of the tasks were assigned according to the start time of the corresponding task in the schedule depicted in Figure 5-23 (Page 123) and is presented in the Appendix. The generated base schedule is depicted in

Figure 7-11. The calculations were executed on a computer with four Intel i7-4700 HQ @2.400 GHz cores with 8GB RAM.

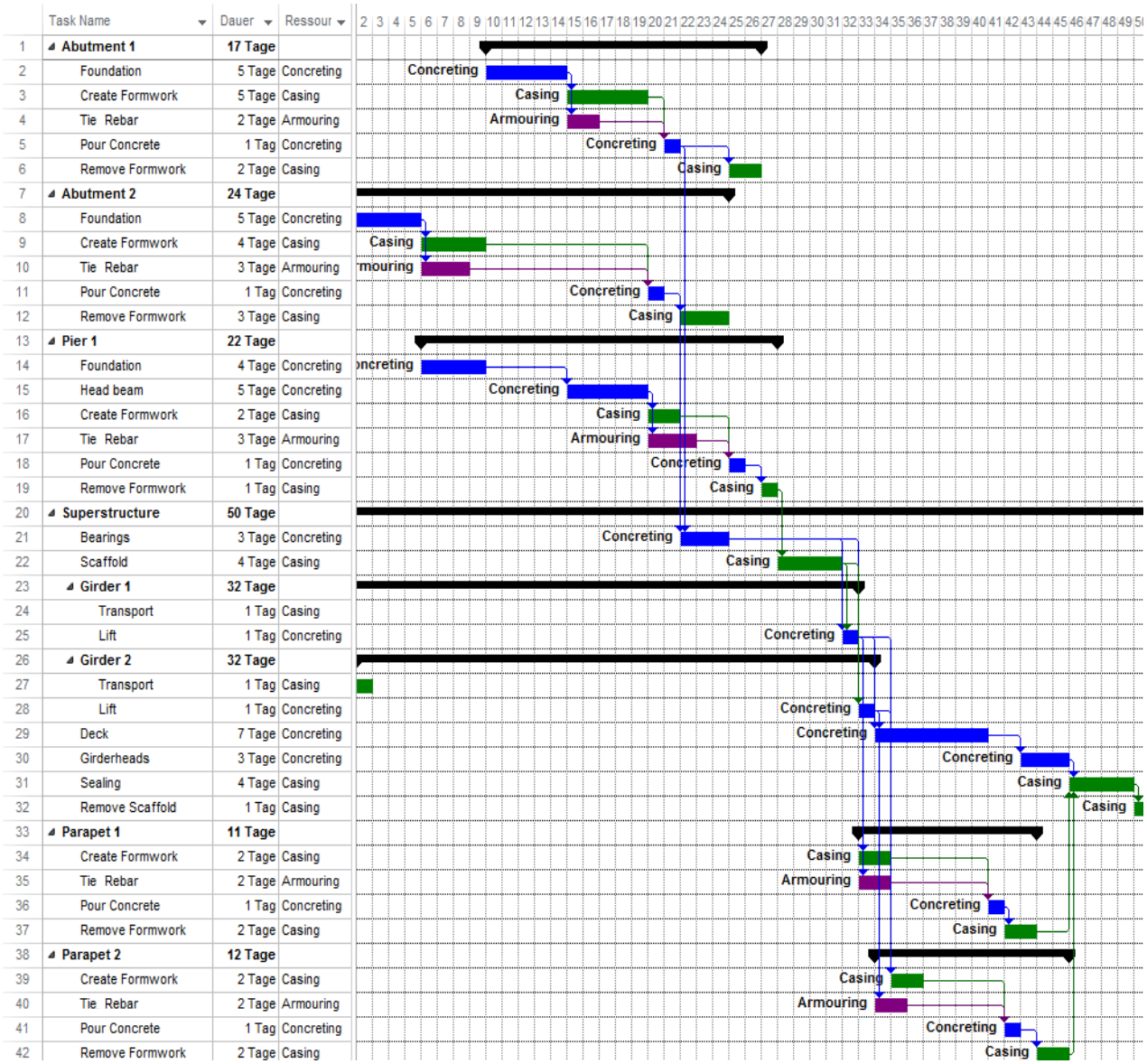


Figure 7-11: The base schedule for the introduced test case with reduced resource availability

The makespan of the base schedule is 50 days and 13 reasonable swaps have been identified that can change the execution order of the tasks. The objective of the case study is to enhance this makespan using the greedy-like and the simulated annealing-based heuristic optimization strategies and compare their results. The fragmented dependency matrices of the project are represented in Section 5.5, in Figure 5-20. The number of independent task pairs within the schedule is 78. Thus, the number of possible schedules for this simple project according to Section 7.4 is about 2,35*10^78.

Since the makespan of the project is relatively low compared to real construction projects that often last several years, higher tolerance factors might have been applied. For example 10% tolerance would accept schedules that are shorter than 55 days in the first generation of schedules. Table 7-1 demonstrates the necessity of the tolerance factor and how it influences the results of the optimization. Since both optimization methods were developed for industrial use, where the duration of the

calculation is important, the optimization process for this small problem will be stopped after 2 minutes. Furthermore, this manual stopping will only be applied when the amount of accepted nodes on one level is high (more than 100000) and because of such cases the algorithm becomes too inefficient.

Test scenario	Strategy	Tolerance factor [%]	Rate of tolerance decrease [%/level]	base schedule makespan [days]	best found makespan [days]	amount of simulation runs	calculation time [min]	reached level
1	Greedy	0	0	50	50	15	< 00:01	2
2	Greedy-like	1	0	50	49	1262327	02:00	10
3	Greedy-like	5	0	50	47	1137682	02:00	9
4	Greedy-like	10	0	50	47	947878	02:00	7
5	Greedy-like	5	0	50	47	2561125	05:00	8
6	Simulated annealing	10	1	50	47	1278110	02:00	9
7	Simulated annealing	10	3	50	49	2600	< 00:01	5
8	Simulated annealing	15	3	50	47	3679	< 00:01	6
9	Simulated annealing	20	3	50	47	726698	00:59	9

Table 7-1: Results of the first optimization case study

In the first scenario a completely greedy algorithm was applied to solve the problem. Since the tolerance factor in this case was 0% and since in the first neighborhood schedules none had a shorter makespan than the base schedule, the optimization terminated without any improvement. Therefore, to determine more schedules and possibly improve the results of the makespan optimization, a higher tolerance factor must be set.

Thus, the following scenarios were executed with the greedy-like algorithm with non-zero tolerance factors. In the first case (scenario 2), a low tolerance factor of 1% was applied. The results show that the algorithm proceeded further than it had in the completely greedy case and reached the 10^{th} level of the enumeration tree identifying more than 1.2 million schedules. The best solution found enhances the makespan of the base schedule by one day. Because of the low tolerance factor, the algorithm identified many nodes and therefore a higher tolerance factor of 5% was tested in the second scenario. By using the second scenario, a solution was found at the fourth level of the enumeration. This solution has a makespan of 47 days. This schedule is the best solution for the considered problem.

Since the solution is on the fourth level of the enumeration tree and it was not identified in the first and second scenario, it is clear that one of its ancestors on a higher level had a larger makespan than it was allowed. Therefore this node had been ignored and the node with a shorter makespan could not be reached. By raising the tolerance factor in the second scenario, this schedule was not precluded, and in the next iteration step the best solution uncovered so far was found.

Further scenarios using the greedy-like algorithm considered even higher tolerance-factors, however they did not lead to any better solutions than the second scenario. To test whether a better solution might exist a scenario (5) with the computation time of 5 minutes has been executed with the tolerance factor of 5%. The scenario did not deliver any further improvement.

For the scenarios 6-9 the simulated annealing algorithm was used. Since with this algorithm the tolerance factor is lowered at every level, the scenarios might start with a higher tolerance factor than the ones for the greedy-like algorithm. This allows the consideration of a wider field of schedules at the beginning of the optimization than the greedy-like strategy. This wider field will be narrowed at every level of the enumeration tree by lowering the tolerance factor. When the tolerance factor of the simulated annealing algorithm reaches zero, the optimization proceeds from that point on as a greedy algorithm.

At the first scenario, the tolerance factor was set to 10% and the rate of decreasing was set to 1% per level. The calculations were interrupted after 2 minutes at the amount of over 1.2 Million determined schedules. Within these schedules the best one was the same as the best solution of the greedy-like search.

In order to test how the decreasing rate of the tolerance factor might influence the results, in the second scenario it has been raised to 3% per level. In this case the optimization terminated after 2600 schedules since it could not find any better solutions in the neighborhood of the determined solutions. Since the starting tolerance factor was not high enough due to the higher decreasing rate, the solution with the makespan of 47 days on the fourth level was not found. Therefore for the next scenarios the initial tolerance factor has been raised. In both further scenarios this solution has been found but no further improvement was observed.

However the introduced case study was a simple example and the makespan of the best schedule was near to the makespan of the base schedule on order to describe the general characteristics of the introduced two heuristic methods. Furthermore the importance of considering worse results than the current best has also been shown.

7.6.2 Second case study

With the previous test scenario it has been shown that both introduced heuristic optimization techniques are capable of improving the makespan of a construction schedule. However, it is important to stress that the results depend on the proper selection of the tolerance factor and the decreasing rate, which is difficult to properly select without any previous knowledge about the considered problem. Furthermore, since the actual optimum is not known, the quality of the best found schedules cannot be evaluated. Due to the small size (number of reasonable swaps) and makespan of the first case study, the number of results were low and therefore making it was hard to find any noticeable improvement in the makespan. Therefore a larger construction project scenario with 457 tasks using 6 different classes of resources has also been tested.

The list of the tasks and precedence relationships has been adapted from a real construction project, but the durations and resource requirements have been modified manually. The resource utilization has been calibrated such that the makespan of the base schedule is 590 days and within the schedule 353 reasonable swaps can be executed. The input data for the simulation is presented in the Appendix. Since the makespan of this project is much longer than in the previous test case, the duration-based tolerance factor will be used. Furthermore, due to the larger project size a longer calculation time limit of 5 minutes is set. The results are presented in Table 7-2.

The first completely greedy scenario terminated after the third level of neighborhood solutions with the improvement of 14 days. Introducing a tolerance factor of 1 day (scenario 2) the calculation proceeded one level further, but the best solution has not been improved. Setting the tolerance factor to 5 days (scenario 3) the optimization proceeded even further and the calculation was stopped manually after 5 minutes. This resulted in a schedule that improved the makespan by 16 days when compared to the base schedule. To analyze how higher tolerance factors might influence the results a 10 day tolerance factor was applied for scenario 4. It resulted in a new best solution with the makespan of 568 days. Raising the tolerance factor to 20 days (scenario 5) the optimization reached only the 4th level of the enumeration tree, due to the high amount of accepted schedules and the limited time. The current best solution with the makespan of 568 days on the 5th level was not located.

In the first two scenarios (scenario 6 and 7) with the simulated annealing algorithm the base tolerance was set to 5 days which is relatively low. In the former case the decreasing rate was set

from 2 days in the latter one to 1 day. In both cases the calculations terminated by themselves. In scenario 6 the best solution that was found had the makespan of 574 days. In scenario 7 the best solution has a makespan of 562 days. The former result corresponds to the schedule determined with the 3rd greedy-like scenario while the latter one is a new current best solution for the problem. A new best solution could be found because the search was able to proceed deeper in the search tree than in the previous scenarios.

For the next scenarios, similar to the scenarios with the greedy-like algorithm, the tolerance factor was raised. Scenarios 8, 9, 10, 11 and 12 had a base tolerance factor of 10 days and a decreasing rate of 5, 3.5, 3, 2 and 1 day(s) respectively. The lower the decreasing rate, the lower was the level reached within the enumeration tree thereby improving the results found. The best solution was found in scenario 12 with the makespan of 556 days. This scenario has been stopped manually after 5 minutes again due to the high amount of accepted schedules. Every other scenario terminated by itself after the calculation time represented in Table 7-2. In the last scenario (13) the tolerance factor was raised to 20 days and the decreasing rate was lowered to 3 days/level. This scenario had to be stopped manually after 5 minutes, similar to scenario 5, due to the high number of accepted solutions.

Test scenario	Strategy	Tolerance factor [days]	Rate of tolerance decrease [days/level]	base schedule makespan [days]	best found makespan [days]	amount of simulation runs	calculation time [min]	reached level
1	Greedy	0	0,00	590	576	597	00:02	3
2	Greedy-like	1	0,00	590	576	840	00:03	4
3	Greedy-like	5	0,00	590	574	88240	05:00	5
4	Greedy-like	10	0,00	590	568	88498	05:00	5
5	Greedy-like	20	0,00	590	570	88430	05:00	4
6	Simulated annealing	5	2,00	590	574	48515	02:53	5
7	Simulated annealing	5	1,00	590	562	51598	03:04	8
8	Simulated annealing	10	5,00	590	574	53417	03:10	5
9	Simulated annealing	10	3,50	590	570	57242	03:26	6
10	Simulated annealing	10	3,00	590	562	1849	00:07	7
11	Simulated annealing	10	2,00	590	559	16722	00:59	8
12	Simulated annealing	10	1,00	590	556	86500	05:00	10
13	Simulated annealing	20	3,00	590	570	87317	05:00	5

Table 7-2: Results of the second optimization case study

As the presented case studies have shown, both the greedy-like and the simulated annealing strategies turned out to be an effective approach to optimize construction schedules. The challenge in both techniques is setting the correct tolerance factor and setting the proper decreasing rate for the simulated annealing. As demonstrated earlier, these two values can have a significant effect on the calculation time. Setting them too low might cause good solutions to be neglected (scenario 1, 2, and 6 for test case 2), while setting them too high might cause the search to become inefficient (scenario 5 and 13 for test case 2). As a general rule, the starting configuration for smaller problems with the makespan of approximately 100 time instances and tasks with an average duration of 4-5 time instances should be a tolerance factor of approximately 10% for both strategies. For larger projects with a makespan of approximately 1000 time instances and tasks with an average duration of 10-20 time instances the tolerance factor should be lowered to 1-5%.

In comparing the two techniques it is clear that within the same time limit the simulated annealing approach was able to reach deeper levels of the enumeration tree and often resulted in a schedule with a shorter makespan than the greedy-like algorithm. On the one hand this can be attributed to the decreasing tolerance factor because more and more solutions can be identified on deeper levels than with the greedy-like algorithm. On the other hand, this could also be attributed to the lower rate of identical determined schedules (introduced in Section 7.5.1).

However the tabu search algorithm prohibits to swapping back the last applied priority swap, making it still possible that the same schedule is determined multiple times. With the greedy-like strategy these multiply determined non-fathomed schedules deliver the same results every time and, due to the constant tolerance factor, they will always be accepted thereby expanding the width of the enumeration tree and the calculation time. For the simulated annealing case, due to the decreasing tolerance rate, these nodes might all be fathomed on the next level. To demonstrate the effects of multiply determined schedules the depth oriented algorithm coupled with the simulated annealing heuristic approach will be tested in the second case study. The results are presented in Table 7-3.

In the beginning the tolerance factor was set to 10 days and the decreasing rate to 3 days/level. When the improvements of the current best schedule became lower, the tolerance factor was decreased to 4 days and the decreasing rate to 2 days/level. With a higher tolerance factor most of the previous results would have been accepted again and the search space would have stayed large. Therefore when the rate of improvement of the current best schedule decreases, the starting tolerance factor and the decreasing rate should also be decreased.

Run	Tolerance factor [days]	Rate of tolerance decrease [days/level]	base schedule makespan [days]	best found makespan [days]	amount of simulation runs	calculation time [min]	reached level
1	10	3,00	590	562	1849	00:07	6
2	10	3,00	562	548	40080	02:22	6
3	10	3,00	548	543	77198	04:25	3
4	10	3,00	543	541	91250	05:20	3
5	10	3,00	541	540	89302	05:02	3
6	4	2,00	540	538	71019	03:49	4
7	4	2,00	538	536	120542	06:35	4
8	4	2,00	536	527	7190	00:24	4
9	4	2,00	527	525	6644	00:22	4
10	4	2,00	525	523	41126	02:21	4
11	4	2,00	523	519	54405	03:00	4

Table 7-3: The results of the depth search oriented simulated annealing optimization strategy

After the 11th run no further improvements could have been determined. The best determined solution had a makespan of 519 day which means an improvement of 71 days or 12% compared to the makespan of the randomly generated base schedule. The complete calculation took 33:47 minutes, which is a time worth to invest to gain the previously introduced improvements of a schedule.

Comparing the results with the results of the first simulated annealing-based approach an improvement can be clearly identified. However, the calculations took longer the best found schedule is 51 days (550 days - 519 days = 51 days = 10% of total makespan of the current best solution) shorter than for the simulated annealing approach. Therefore the application of the depth oriented search is preferred as long as an efficient algorithm is developed that is capable to avoid the repetition of the same schedule inside the whole enumeration tree.

7.7 Summary

To summarize the results of the last chapter, the introduced reasonable swap-based optimization strategies are effective ways to optimize construction schedules. Using the depth oriented simulated annealing approach deeper levels of the enumeration tree might be reached and such further

improvements might be found than in the normal simulated annealing approach, since the number of multiply determined identical schedules is lower. It is important to stress that since the introduced strategies are heuristic approaches, it cannot be proven that the best solutions found while using the program are in fact the actual optimum of the optimization problem. In order to further enhance the effectiveness of the introduced reasonable swap-based optimization strategies an algorithm is necessary that is capable of detecting and identifying existing results already in the enumeration tree. This is important to avoid repetition of accepted schedules and to keep the breadth of the search as narrow as possible. With a depth oriented search the chances of obtaining more results that are better than those obtained from a base schedule are even higher.

Selecting a proper tolerance factor for the determination is a challenging task. Its value depends on the makespan of the project and the actual duration of each single task. As discussed in the presented case studies, for smaller problems with the makespan of approximately 100 time instances and tasks with an average duration of 4-5 time instances, a tolerance factor of around 10% should be considered as a good starting value. For larger projects with a makespan of approximately 1000 time instances and tasks with an average duration of 10-20 time instances, the tolerance factor shall be lowered to 1-5%. When the average duration of the tasks increases, the tolerance factor should also be increased since the variance of the makespan of the results is larger than for projects with lower average task duration. Furthermore, for the depth oriented search, when the rate of improvement of the current best schedule decreases between optimization runs the starting tolerance factor and its decreasing rate should also be decreased in order to reduce the search space and to decrease the rate of the repeated calculation of results that were already determined by previous optimization runs.

8 Summary and outlook

Nowadays in the construction industry schedules are created manually in a time consuming and laborious way. Therefore, the aim of this thesis was to enhance, accelerate and, where possible, automate this scheduling process. In this thesis different concepts and methods have been presented for the creation of near optimal and flexible schedules for construction projects. The beginning of the thesis discussed in general terms the definition of a construction project, including its components and kinds of relationships and restrictions that exist among the various components. Here, the ideas of construction tasks, precedence relationships, also called technological dependencies between tasks, resources and resource constraints were introduced (Chapter 1).

Since every construction project is unique, they all require an individual schedule for their execution. Schedules describe the execution sequence of the construction tasks and also give information about the resource utilization. Scheduling is an iterative process that searches for a schedule that satisfies all of the predefined restrictions such as resource limits and objectives. An example of a common objective for a construction project is finishing a project before a deadline.

A schedule that satisfies all the restrictions or constraints is called a feasible schedule. While feasible schedules satisfy all the precedence and resource constraints, they do not necessarily satisfy all of the scheduling objectives. An objective of scheduling (Section 1.3) can be to find the schedule with the shortest makespan. This scheduling problem in the construction industry can be formulated as a resource-constrained project scheduling problem (RCPSP - Section 2.2.2). The objective of the RCPSP is to find the schedule with the shortest makespan while satisfying all of the precedence and resource constraints of the tasks without preemption. The RCPSP belongs to the class of NP-hard optimization problems, which means that finding the optimal solution is not possible in polynomial time. How to determine feasible, but not necessarily optimal, solutions for the RCPSP is one of the main topics of this thesis.

In the construction industry, as described in Section 2.2, the schedules are primarily determined by the network scheduling techniques such as the Critical Path Method or the Precedence Diagram method. These methods have the advantage of allowing the user to create a schedule quickly and easily. However, these methods cannot take resource constraints into consideration. This can result

in an overutilization of resources, since the necessary resources are not available on the construction site, thereby leading to a continuous extension of the completion time. Furthermore, due to this limitation, these methods are not capable of determining feasible solutions for the RCPSP.

To enhance the scheduling process in the construction industry, first, similar scheduling problems and their solution methods from the manufacturing industry were studied and discussed (Section 2.3). In the manufacturing industry the most important scheduling problems are the shop problems. As described in Section 2.4.1, the shop problems are a subset of the RCPSP. Due to the more general resource and precedence constraints their solution methods are only useful in finding a solution for the RCPSP if extensions are included.

One promising approach that is capable of taking resource constraints into account is the simulation-based scheduling technique, specifically called the discrete event simulation (Section 3.3). The discrete event simulation is a type of simulation that models systems for which changes in the system state, so called events, occur only at discrete points in time. The simulation time of the model jumps forward in time between these events while the model´s state between these discrete time steps stays constant. This simulation is able to handle complex systems and provide information about its behavior. Therefore both researchers and those in the industry apply this simulation technique to support the decision making processes. Within the wide range of applications the discrete event simulation is also suitable for solving scheduling problems.

Section 3.5 introduces a detailed review of existing scheduling approaches based on simulation techniques for the construction industry. It is however important to emphasize that these approaches are only applicable for repetitive construction processes such as earth hauling, earth transport and tunneling. The review collects the most significant simulation frameworks from the origins of construction simulations until the present time. The concept of CYCLONE, STROBOSCOPE, Simphony and Cosye were discussed in detail.

The direct application of the discrete event simulation to solving construction project scheduling problems was not possible due to the complexity of the scheduling problems and the differences between the main characteristics of the manufacturing industry versus the construction industry (Section 3.4). This is primarily due to the necessarily rigid sequences of activities that are common in the manufacturing industry. Such a behavior is only suitable for machine driven or for cyclic construction processes, such as earth transportation, as described by the introduced construction simulation frameworks. In contrast, the sequence of tasks and the working path of the resources on a construction site are more dynamic and spontaneous. Hence, decisions such as identifying which of several tasks to execute when there are only enough resources to start one task can only be solved with the discrete event simulation in a very complex way.

This dynamic decision-based behavior describes a combinatorial problem. Since combinatorial problems can be solved efficiently by the constraint satisfaction paradigm, a group of German researchers (König et al. 2007a, König et al. 2007b, Beißert et al. 2007b) integrated the constraint satisfaction paradigm within the discrete event environment and created the constraint-based discrete event simulation approach (CBDES – Section 4.2). The CBDES is capable of determining single feasible schedules for scheduling problems while also taking precedence and resource constraints into account. The advantage of this technique over other simulation approaches is its capability to make dynamic decisions about which task to execute in tied resource situations. If different decisions in the execution sequence of the tasks for the same problem are made, then the program generates a number of practicable solutions which can be analyzed to address specific issues such as time, cost and quality. Therefore, with the proper steering algorithm, the CDBES can also be used for optimization purposes such as solving the RCPSP.

However, before starting an optimization problem with the CBDES a number of extensions and enhancements must be introduced to make this simulation approach useful and acceptable to the construction industry. The first characteristic that negatively affects the popularity of simulations in the industry is the time-consuming preparation process of the input data (Section 4.3). Every simulation run requires the definition of every single construction task along with their technological dependencies, their resource requirements and the available amount of resources. To collect and organize all of this data is not only time consuming but also a complex task where the overview of the completed data can easily be lost. To accelerate this data preparation process and get the data better organized, Wu et al. (2010b) introduced a levels-of-detail hierarchy (Section 4.3.2) and a pattern-based approach (Section 4.3.3) that was then further enhanced by the author of this thesis.

To enhance this preparation process a software-tool called Preparator (Section 4.3.5) was developed that applies the introduced levels-of-detail approach and the process patterns. With the Preparator the user can connect these process patterns and activity packages with 3D objects of the building to provide a clear overview of the already assigned construction tasks. By using the Preparator, building components whose construction methods are not yet assigned might be identified. The use of a 3D model also facilitates the creation of precedence relationships between construction tasks that belong to different building components such as identifying that the building of the abutment must precede the construction of the superstructure.

Therefore, using the Preparator can accelerate the preparation process of the input data required for simulations. However, the overall preparation is still time consuming due to the large amount of customizable data (e.g. performance factors, necessary resources for individual tasks, etc.) in spite of automated hierarchic modeling, process patterns and 3D-model connection.

Hence, further research must be undertaken in this field to achieve a simulation technique that is also favored by the industry. A BIM and knowledge-based method, such as the case-based reasoning, represent a good basis for overcoming these issues and for an automated precedence graph generation for an entire construction project. Further research should focus on ways in which the creation of custom construction methods and input of custom data can be accelerated or automated. In addition, another important research task is how to compose automated definitions of different construction scenarios for the same project. This would allow the planner to compare not only schedules with different task sequences, but also schedules with different construction scenarios, such as diverse resource utilizations.

Beside the enhancement of the input data preparation of process simulation, in order to raise the acceptance rate of the simulation-based scheduling within the construction industry it must become competitive with the conventionally used network scheduling techniques. The network scheduling techniques are able to determine float time for individual tasks in a simple manner, taking only precedence constraint into account. Therefore the author introduced a newly developed simulation-based technique that is able to determine total float for every individual task in the project also taking precedence and resource constraints into account in one determination step (Section 5.3 and Section 5.4).

This was achieved by a methodology that is similar to the forward and backward pass of the network scheduling techniques. For the simulation, the concept of the forward and backward simulation has been introduced. The concept of the forward simulation is identical to the concept of the introduced CBDES method (Section 4.2.2). It is used to generate a feasible schedule for which the total float time of the tasks is determined.

The concept of backward simulation (Section 5.3.1) is basically the same as that of the backward pass – the simulation actually runs forward in time but with reversed execution conditions. Using this approach, the resulting schedule is calculated based on a simulation that starts from a virtual

completion date for the construction project and runs backwards in time until the starting point of the construction process is reached. The most challenging task in the backward simulation is to follow an identical schedule sequence to that of the forward simulation.

In order to achieve schedule compatibility, every task has to start at the same date or later in time than in the forward simulation. To this end, the task selection algorithm was modified so that the backward simulation uses a priority-based approach to determine the next executable task. The priorities of the tasks are assigned according to the completion date of the task in the forward simulation. To solve the unintended swap phenomenon where several independent tasks that use the same resources swap position in the backward simulation, sequence enforcement constraints are defined based on the result of the forward simulation.

By comparing the results of the backward simulation after applying these constraints with the schedule of the forward simulation, the execution order of the tasks will be identical and every task will start later in time. Hence, the time difference between the earliest and latest start date of a task represents its total float without exceeding the resource limits at any time during the project. The tasks without total float comprise the critical chain of tasks for the respective configuration of resources. A comprehensive case study was introduced to illustrate the application of this new approach and demonstrated that the determination of detailed total float for each individual task using discrete event simulation taking into consideration available resources is realizable and that the determined results are also feasible (Section 5.5). However, the limitations of the method were also revealed.

When there is a large enough time frame within the schedule, and within this large time frame independent tasks that are executed simultaneously, can be pushed one after another without exceeding the resource limits, a phenomenon called serialization of tasks can be observed. This situation leads for the float time determination to an over-constrained case. Therefore the determined float times for these tasks will be less than their actually possible total float.

The serialization of the tasks means, that the simultaneously executed tasks during the forward simulation might be ordered into a series of tasks. Thus, creating one larger, connected task made of two smaller tasks. On this way they require fewer resources than unconnected and so they can be pushed more forward in time such gaining more float than would they be pushed separated forward in time. When no serialization can occur the determined float values for the individual tasks are correct. Further research should investigate how to loosen the restrictions of the sequence enforcement constraints so that the results are both feasible and correct in every case.

The application of the developed methodology was also been presented with multiple resource constraints for the individual tasks (Section 5.7). Here, due to the larger number of resource constraints, the flexibility of the tasks has been restricted thereby reducing the amount of total float for the tasks.

With the enhancement of the input data preparation process and the float time determination method, the constraint-based discrete event simulation has been made competitive with the conventionally used network scheduling techniques. An additional development for this technique was an external algorithm that is capable of guiding and steering the simulation through the search space of the RCPSP to determine feasible schedules with near optimal makespan for the considered construction project.

Chapter 6 included the most important definitions, characteristics and features of an optimization problem, such as the RCPSP, and a discussion regarding some already developed optimization techniques.

The RCPSP belongs to the class of NP-hard combinatorial optimization problems. As already explained above, this means that it is not possible to determine the optimal solution in polynomial

time. Due to the exponential complexity of the problem and the combinatorial explosion, the larger the problem becomes (e.g. the more tasks the project contains), the more the size of the search space explodes making the identification of all of the possible solutions impossible. Therefore, the application of advanced solution methods is necessary since they are capable of manipulating the size of the search space by subdividing, reducing or traversing through it.

The two deterministic approaches, the Integer programming technique (Section 6.5.1.1) and the Branch-and-Bound method (Section 6.5.1.2) were analyzed in detail by focusing on the solutions for the RCPSP. Such deterministic solution techniques are capable of identifying the actual optimal solution for the optimization problem but greater computational effort is required for larger problems.

In contrast, heuristic solutions are capable of generating feasible solutions for optimization problems in low computational time. However, their results may be suboptimal. These methods are based on decisions that are inspired by common sense or intuition such as the rule of thumb, a good guess or the use of an estimate. The working mechanism of the most widely used heuristic approaches like the Greedy algorithm, the tabu search, the simulated annealing, the genetic algorithm, the particle swarm optimization and the ant colony optimization were discussed in detail in Section 6.5.2. Using these techniques to solve the RCPSP was addressed in the corresponding sections.

Since the CBDES uses a time increment-based scheduling algorithm it is only able to generate non-delay schedules, which might not contain the actual optimal solution for the optimization problem (Section 7.1.1). Therefore, the application of a deterministic solution strategy that uses the CBDES to solve the RCPSP might result in a solution other than the actual optimal solution. When the industrial application of software is desired, the computational time of the solutions process should be low in order to keep the techniques attractive. Therefore the author decided to develop a heuristic method that guides the CBDES in an effective way through the search space of the makespan optimization problem to generate near optimal solutions for the RCPSP.

The idea behind the new heuristic approach is to use an algorithm that swaps the position of certain task pairs within the schedule and so it is capable of traversing through the search space of the RCPSP. Furthermore it was important that applying a swap of tasks within the program results in a new schedule. This is considered as an important development compared to existing heuristic approaches. The first phase of the research was to identify the task pairs that when swapped result in a change in the schedule. These are called reasonable swaps (Section 7.3). To determine the list of reasonable swaps first a feasible schedule must be generated with the CBDES. In this schedule points in time can be identified where tasks requiring the same resources cannot be executed simultaneously due to resource limits. The combinations of the corresponding tasks create the list of reasonable swaps. The list of reasonable swaps varies with the schedules. In order to manipulate the order of these tasks within the schedule a priority-based scheduling algorithm was applied for the CBDES. Swapping the priorities of a reasonable swap will generate a new and different schedule. Furthermore, a reasonable swap is identified as a decision made by the simulation during the scheduling process. Increasing the available resources at this date results in eliminating the identified swap and thus the concerned tasks can be executed simultaneously, shortening the duration of the project on minimal costs.

The next phase of the research was to define an algorithm that is capable of using these reasonable swaps and steering the simulation to generate schedules that work toward the optimum. An enumeration tree was used to represent the search space and the results of the optimization. A neighbor solution is generated by applying one reasonable swap to the schedule. A randomly generated base schedule is on the top of the enumeration tree. Each level works toward enhancing the schedule to move it more toward an optimal schedule. The schedules on the next level will be generated by applying one of the reasonable swaps from the schedule.

To traverse effectively in this enumeration tree two meta-heuristic approaches were introduced. An additional third strategy has also been developed based on the experiences with the previous two methods. The idea behind traversing within the enumeration tree is to determine a node and evaluate its result. When it is acceptable the neighborhood solutions are generated for it. When a node is not acceptable under the program, it is identified and deleted from any further calculations. Since the result of a reasonable swap cannot be foreseen, a swap might result in a better schedule but it could also lead to a worse schedule. Therefore it was important for the developed techniques to be able to exit from local optima. This was achieved by adding in a tolerance factor. The tolerance factor defines a limit of how much longer than the current best solution a schedule is allowed to be in order for it to be considered for further calculations.

The first steering technique is a Greedy-like algorithm (Section 7.5.2) that uses a constant, relatively low tolerance factor. The second steering technique is a simulated annealing-based technique which starts with a higher tolerance factor then the greedy-like algorithm. The tolerance factor is then reduced at each new level of the enumeration tree (new neighbor solutions). Both strategies are enhanced with a tabu search algorithm that prohibits the "swap-back" of the previously applied last reasonable swap thereby lowering the chance of identifying the same schedule multiple times. Although the tabu search algorithm avoids the repeated generation of the same schedule on the same branch, the same schedule can still be generated on different branches of the tree causing the identification of the same child nodes multiple times. To reduce the effect of the potential of identifying the same schedule multiple times within the enumeration tree, a third approach (Section 7.5.4) has been developed. This approach is called the depth oriented heuristic search algorithm. This approach uses the simulated annealing approach to determine results only few levels deep within the enumeration tree (levels 4-6 depending on the size of the problem) and then restarts the optimization with the current best solution as root node. Thus after the predefined number of levels the duplicated schedules will be ignored and the optimization continues with only the current best solution.

Two case studies have been investigated to test the applicability of these heuristic optimization strategies (Section 7.5.4). As a result of the study, both previously introduced former meta-heuristic optimization methods turned out to be effective for identifying the optimal construction scheduled relative to makespan and for solving the RCPSP. In comparing the two techniques, it was demonstrated that within the same computation time limit the simulated annealing was able to reach deeper levels of the enumeration tree and often was able to find a schedule with a shorter makespan than the greedy-like algorithm. This could be due to the decreasing tolerance factor which allows for more solutions identified on deeper levels than with the greedy-like algorithm. Alternatively, this could be due to the lower rate of duplicated schedules.

By comparing the results of the third, depth oriented heuristic search algorithm with the simulated annealing approach, it is clear that the depth oriented approach found a better solution for the problem. Although the calculations took longer, the best schedule found from using the depth oriented approach had a makespan that was approximately 10% shorter than the total makespan of the current best solution using the simulated annealing approach. Therefore the application of the depth oriented search is preferred. The better results can be explained by the reduced search space of the method due to the reduction of the duplicated schedules.

It is important to stress that since the introduced strategies are heuristic approaches, it cannot be proven that the best solutions found by these approaches are in fact the optimal solution for the optimization problem. Further improvements can be introduced by further reducing the number of duplicate schedules.

The selection of the proper tolerance factor for the optimization is a challenging task. Its value depends on the makespan of the project and the actual duration of each task. For smaller problems with the makespan of approximately 100 time instances having tasks with an average duration of 4-5 time instances, a starting tolerance factor of 10% should be considered. For larger projects with a makespan of approximately 1000 time instances having tasks with an average duration of 10-20 time instances, the tolerance factor should be lowered to 1-5%. When the average duration of the tasks increases, the tolerance factor should also be increased. Furthermore, for a deeper search, when the rate of improvement of the current best schedule decreases between optimization runs the starting tolerance factor and its decreasing rate should also be decreased in order to reduce the search space and to avoid or lower the rate of the duplication of results already determined by previous optimization runs.

In summary, in this thesis, simulation-based methods were introduced for data preparation, float time determination and schedule optimization for construction projects. The purpose of the conducted research was to raise the competitiveness and acceptance of simulation-based scheduling techniques in the construction industry. This has been achieved by the introduction of new approaches. These are the software Preparator, the float time determination under resource constraints, and the reasonable swap-based heuristic optimization strategies.

To achieve further success with the simulation-based scheduling, first its scheduling algorithm should be enhanced so that it is capable of determining active schedules in addition to its current ability to identify non-delay schedules. Thus it could also be used for deterministic optimization purposes. Furthermore, additional methods should be developed that automate and also accelerate the data preparation process for the simulation. For the float time determination a solution for the serialization problem would result in a technique that is capable of determining the correct amount of float time for every individual task in the resource constrained case. The further development of the priority swap-based heuristic optimization technique would also likely further reduce the number of duplicate schedules. Additional research should extend the application field of the CBDES to generate feasible solutions for the multi-mode RCPSP, which would support the analysis of different scenarios for construction projects. One further important research topic should address the application of time and space constraints for single tasks that define a frame in which the corresponding task or tasks must be executed.

The introduced approaches are mostly applicable in the later design phases of a construction project when there are enough information about the construction site, the boundary conditions such as restrictions in time and space and the building or bridge itself available. Through the developed methods one can accelerate the design phase of a construction project and so the designer has the opportunity to investigate higher number of scenarios for the project, or to prepare the selected design more detailed in a higher quality than before.

References

Abdelmaguid, T. F. (2009). "Permutation-induced acyclic networks for the job shop scheduling problem." *Applied Mathematical Modelling* 33 (3): 1560–72.

AbouRizk, S., and Hajjar, D. (1998). "A Framework for Applying Simulation in the Construction Industry." *Can. J. Civ. Eng* (253): 604–17.

AbouRizk, S., Halpin, D., Mohamed, Y., and Hermann, U. (2011). "Research in Modeling and Simulation for Improving Construction Engineering Operations." *Journal of Construction Engineering and Management ASCE* 137 (10): 843–52.

AbouRizk, S. M., and Hague, S. (2009). "An Overwiew of The Cosye Environment for Construction Simulation." *Proceedings of the 2009 Winter Simulation Conference*, 2624–34.

AbouRizk, S. M., Halpin, D. W., and Lutz, J. D. (1992). "State of Art in Construction Simulation." *Proceedings of the 1992 Winter Simulation Conference*, 1271–77.

AbouRizk, S. M., Mohamed, Y., Taghaddos, H., Saba, F., and Hague, S. (2010). "Developing Complex Distributed Simulation for Industrial Plant Construction Using High Level Architecture." *Proceedings of the 2010 Winter Simulation Conference*, 3177–88.

AbouRizk, S. M., and Yasser, M. (2002). "Optimal Construction Project Planning." *Proceedings of the 2002 Winter Simulation Conference*, 1704–08.

Afshar, A., Ziaraty, A., Kaveh, A., and Sharifi, F. (2009). "Nondominated Archiving Multicolony Ant Algorithm in Time–Cost Trade-Off Optimization." *Journal of Construction Engineering and Management* 135 (7): 668–74.

Al Sarraj, Z. M. (1990). "Formal Developement of Line-of-Balance Technique." *Journal of Construction Engineering and Management* 116: 689–704.

Anagnostopoulos, K., and Koulinas, G. (2012). "Resource-Constrained Critical Path Scheduling by a GRASP-Based Hyperheuristic." *Journal of Computing in Civil Engineering* 26 (2): 204–13.

Arditi, D., Tokdemir, O. B., and Suh, K. (2002). "Challenges in Line-of-Balance Scheduling." *Journal of Construction Engineering and Management* (128): 545–56.

Artigues, C., Brucker, P., Knust, S., Koné, O., Lopez, P., and Mongeau, M. (2013). "A Note on "Event-Based MILP Models for Resource-Constrained Project Scheduling Problems." *Computers & Operations Research* 40 (4): 1060–63.

Artigues, C., Sophie Demassey, and Emmanuel Néron (2010). *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications:* ISTE Ltd and John Wiley & Sons Inc.

Artigues, C., Koné, O., Lopez, P., Mongeau, M., Néron, E., and Rivreau, D. (2008)."Computational Experiments." In *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*, 98–102.

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): 249–67.

Baar, T., Brucker, P., and Knust, S. (1999). "Tabu Search Algorithms and Lower Bounds for the Resource-Constrained Project Scheduling Problem." In *Meta-Heuristics*. Edited by Stefan Voß, Silvano Martello, IbrahimH. Osman, and Catherine Roucairol, 1–18: Springer US. http://dx.doi.org/10.1007/978-1-4615-5775-3_1.

Baker, K. (1974). "*Introduction to sequencing and scheduling"* Wiley.

Baker, R. D. (1991). "Time-Cost relationship in Construction: Master-Thesis." University of Florida.

Banks, J. (1998). *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice.* A Wiley-Interscience publication: Wiley.

Banks, J., and J.S. Carson (2009). *Discrete-event System Simulation: 5th Edition.* Prentice-Hall international series in industrial and systems engineering: Prentice-Hall.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H. (1998). "Branch-and-Price: Column Generation for Branch-and-Proce: Column Generation for Solving Huge Integer Programs." *Operations research letters* 46: 316–29.

Barták, R. (1999). "On the Boundary of Planning and Scheduling: A Study." *In Proceedings of the 18th Workshop of the UK Planning and Scheduling SIG*, 28–39.

Beißert, U. (2012). *Constraint-basierte Simulation zur Terminplanung von Ausführungsprozessen: Repräsentation baubetrieblichen Wissens mittels Soft Constraints.* Weimar, Germany: Verlag der Bauhaus-Universität Weimar.

Beißert, U., König, M., and Bargstädt, H.-J. (2007a). "Generation and Local Improvement of Execution Schedules Using Constraint-Based Simulation." *Tsinghua Science and Technology* 12 (1): 1–8.

Beißert, U., König, M., and Bargstädt, H.-J. (2010). "Soft Constraint-based simulation of execution strategies in building engineering." *J Simulation* 4 (4): 222–31.

Beißert, U., König, M., and Bargstaedt, H. J. (2007b). "Constraint-Based Simulation of Outfitting Processes in Building Engineering." *Proceedings of CIB-W78 24th International Conference on Information Technology in Construction*, 491–98.

Bell, P. C., and O´Keefe, R. M. (1986). "Visual interactive Simulation - History, Recent Developments, and Major Issues." *Simulation* (49): 109–16.

Bellman, R. (1954). "An introduction to the theory of dynamic programming: Bull. Amer. Math. Soc. 60 (1954), 503-515." *Bulletin of the American Mathematical Society* 60: 503–15.

Benders, J. F. (1962). "Partitioning Procedurs for Solving Mixed-Variables Programming Problems." *Numerische Mathematik* 4: 201–19.

Berkhahn, V., Klinger, A., Rüppel, U., Meißner, U. F., Greb, S., and Wagenknecht, A. (2005). "Processes Modelling in Civil Engineering based on Hierarchical Petri Nets." In *Proc. of the 22nd CIB-W78 Conf.*

Berthaut, F., Pellerin, R., Perrier, N., and Hajji, A. (2011). "Time-Cost Trade-Offs in Resource-Constraint Project Scheduling Problems with Overlapping Modes." *CIRRELT* 10: 0–24.

Berthold, T., Heinz, S., Lübbecke, M. E., Möhring, R. H., and Schulz, J. (2010a). "A Constraint Integer Programming Approach for Resource-Constrained Project Scheduling." *CPAIOR 2010*, 313–17.

Berthold, T., S. Heinz, M. E. Lübbecke, R. H. Möhring, and J. Schulz (2010b). *A Constraint Integer Programming Approach for Resource-Constrained Project Scheduling: ZIBReport 10-03*.

Blazewicz, J., K. Ecker, G. Schmidt, and J. Weglarz (2007). *Handbook on Scheduling: From Theory to Applications.* Berlin: Springer Berlin.

Blazewicz, J., Lenstra, K. J., and Rinnooy Kan, A. (1983). "Scheduling projects to resource constraints: Classification and complexity." *Discrete Applied Mathematics* (5): 11.24.

Blazewicz, J., Pesch, E., and Sterna, M. (2000). "The Disjunctive Graph Machine Representation of the Job Shop Scheduling Problem." *European Journal of Operational Research* 127 (2): 317–31.

Bouleimen, K., and Lecocq, H. (2003). "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version." *European Journal of Operational Research* 149 (2): 268–81.

Bowers, J. (2000). "Interpreting Float in Resource Constrained Projects." *International Journal of Project Management* 18: 385–92.

Brucker, P. (2007). *Scheduling Algorithms:* Springer-Verlag Berlin Heidelberg.

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: 3–41.

Brucker, P., Jurisch, B., and Sievers, B. (1994). "A branch and bound algorithm for the job-shop scheduling problem." *Discrete Applied Mathematics* 49 (1–3): 107–27.

Brucker, P., Knust, S., Schoo, A., and Thiele, O. (1998). "A branch and bound algorithm for the resource-constrained project scheduling problem." *Operational Research* 107: 272–88.

Burghardt, M. (2007). *Einführung in Projektmanagement: Definition, Planung, Kontrolle und Abschluss:* Publicis.

Bügler, M., Borrmann, A. (2014). "Using Swap-Based Search Trees to obtain Solutions for Resource Constrained Project Scheduling Problems" Proceedings *in Applied Mathematics and Mechanics*, Erlangen, Germany, 2014

Caccetta, L., and Hill, S. P. (2001). "Branch and Cut Methods for Network Optimization." *Mathematical and Computer Modelling* 33: 517–32.

Chahrour, R. (2006). "Integration von CAD und Simulation auf Basis von Produktmodellen im Erdbau."

Chahrour, R., and Franz, V. (2002). "Computersimulation … Warum nicht im Bauwesen?" *Tiefbau, Ingenieurbau, Strassenbau* (10): 21–26.

Chaleshtari, A. S., and Shadrokh, S. (2012). "A Branch and Bound Algorithm for Resource Constrained Project Scheduling Problem subject to Cumulative Resources." *World Academy of Science, Engineering and Technology* 62: 27–32.

Chang, D. (1989). *Resque: A Resource Based Simulation System for Construction Process Planning: Ph.D. Dissertation.* Michigan: University of Michigan.

Chen, P.-H., and Weng, H. (2009). "A two-phase GA model for resource-constrained project scheduling." *Automation in Construction* 18 (4): 485–98.

Chen, R.-M. (2011). "Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem." *Expert Systems with Applications* 38 (6): 7102–11.

Cheng, F., Li, H., Wang, Y.-W., Skitmore, M., and Forsythe, P. (2013). "Modeling resource management in the building design process by information constraint Petri nets." *Automation in Construction* 29: 92–99.

Cheng, F., Wang, Y., Ling, X., and Bai, Y. (2011). "A Petri net simulation model for virtual construction of earthmoving operations." *Automation in Construction* 20 (2): 181–88.

Christofides, N., Alvarez-Valdés, R., and Tamarit, J. M. (1987). "Project Scheduling With Resource Constraints: A Branch and Bound Approach." *European Journal of Operational Research* 29 (3): 262–73.

Chua, D. K. H., and Li, G. M. (2002). "RISim Resource-Interacted Simulation Modeling in Construction." *Journal of Construction Engineering and Management ASCE* 128 (3): 195–202.

Concannon, K. H., Hunter, K. I., and Tremble, J. M. (2003). "SIMUL8-Planner Simulation-Based Planning and Scheduling." *Winter Simulation Conference 2003*, 1488–93.

da Costa, N. C. A., Doria, F. A., and Bir, E. (2007). "On the metamathematics of the P vs. NP question." *Applied Mathematics and Computation* 189: 1223–40.

Dantzig, G. B. (1951). "Maximization of a linear function of variables subject to linear inequalities." In *Activity Analysis of Production and allocation*, 339–47.

Dawood, N., Sriprasert, E., Mallasi, Z., and Hobbs, B. (2003). "Development of an Integrated Information Resource Base for 4D/VR Construction Process Simulation." *Automation in Construction* 12 (2): 123–31.

Day, J. E., and Hottenstein, M. P. (1970). "Review of Sequencing Research." *Naval Research Logistics Quarterly* 17 (1): 11–39.

De Reyck, B., and Herroelen, W. (1998). "A Branch-and-Bound Procedure for the Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations." *European Journal of Operational Research* 111: 152–74.

Deb, K. (2004). *Optimization for engineering design: Algorithms and examples:* PHI Learning Pvt. Ltd.

Demeulemeester, E., Herroelen, W., Simpson, W., Baroum, S., Patterson, J. H., and Yang, K. K. (1994). "On a Paper by Christofides et al. for Dolving the Multiple-Resource Constrained, Simple Project Scheduling Problem." *European Journal of Operational Research* 76: 218–28.

Demeulemeester, E. L., and Herroelen, W. S. (1997). "New Benchmark Results for the Resource-Constrained Project Scheduling Problem: Management Science November 1997 vol. 43 no. 11 1485-1492." *Management Science* 43 (11): 1485–92.

Demeulemeester, E. L., Herroelen, W. S., and Elmaghraby, S. E. (1996). "Optimal procedures for the discrete time/cost trade-off problem in project networks." *European Journal of Operational Research* 88 (1): 50–68.

Dijkstra, E. W. (1959). "A Note on Two Problems in Connexion with Graphs." *Numerische Mathematik* 1: 269–71.

Dori, G., and Borrmann, A. (2010). "Bauablaufsimulation und -animation für die Planung von Brückenbauvorhaben." *Proceedings of the 22nd Forum Bauinformatik.*

Dori, G., and Borrmann, A. (2011). "Automatic Generation of Complex Bridge Construction Animation Sections by Coupling Constraint-based Discrete-Event Simulation with Game Engines." *Proceedings of the International Conference on Construction Applications of Virtual Reality 2011*.

Dorigo, M., Maniezzo, V., and Colorni, A. (1991). "The ant system: An autocatalytic optimizing process." 016.

Drake, G. R., Smith, J. S., and Peters, B. A. (1995). "Simulation as a Planning and Scheduling Tool for Flexible Manufacturing Systems." *Winter Simulation Conference 1995*, 805–12.

Duan, Q., and Liao, T. W. (2010). "Improved ant colony optimization algorithms for determining project critical paths." *Automation in Construction* 19 (6): 676–93.

Eastman, C. M., P. Teicholz, R. Sacks, and K. Liston (2011). *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors.* 2nd ed. Hoboken, NJ: Wiley.

Elbeltagi, E., Hegazy, T., and Grierson, D. (2005). "Comparison among five evolutionary-based optimization algorithms." *Advanced Engineering Informatics* 19 (1): 43–53.

Elmaghraby, S. (1993). "Resource allocation via dynamic programming in activity networks." *European Journal of Operational Research* 64: 199–215.

ElNimr, A. A., and Mohamed, Y. (2011). "Application of gaming engines in simulation driven visualization of construction operations." *Journal of Information Technology in Construction* 16 (3): 23–38.

Evans, R. V. (1964). "Queueing When Jobs Require Several Services Which Need Not be Sequenced." *Management Science* 10 (2): 298–315.

Feo, T. A., and Resende, M. G. C. (1989). "A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem." *Operations research letters* 8: 67–71.

Festa, P., and Resende, M. G. C. (2001). "GRASP: An Annotated Bibliography." *Essays and Surveys in Metaheuristics, Kluwer Academic Press, Boston*, 325–67.

Fisher, M. L. (1970). *Optimal Solution of Resource Constrained Network Scheduling Problems:* Operations Research Center, MIT.

Fisher, M. R. (1981). "The Lagrangian Relaxation Method for Solving Integer Programming Problems." *Management Science* 27: 1–18.

Fishman, G. (2001). *Discrete-Event Simulation: Modeling, Programming, and Analysis.* Springer Series in Operations Research and Financial Engineering: Springer.

Fondahl, J. W. (1961). *A non-computer approach to the critical path method for construction industry.* Technical report (Stanford University. Dept. of Civil Engineering): Dept. of Civil Engineering, Stanford University.

Fondahl, J. W. (1987). "Precedence Diagramming Methods: Origins and Early Development." *Project Management* 18 (2): 33–36.

Galati, M. (2010). *Decomposition Methods for Integer Programming: Dissertation*.

Gantt, H. L. (1903). "A Graphical Daily Balance in Manufacture." *ASME Transactions* 24: 1322–36.

Gantt, H. L. (1919). *Work, Wages, and Profits.* 2nd ed., revised and enlarged. New York: The Engineering Magazine Co.

Garey, M., and D.S Johnson (1979). *Computers and intractability:* Freeman San Francisco, CA.

Gass, S., and C.M. Harris (2001). *Encyclopedia of Operations Research and Management Science: Centennial Edition.* SpringerLINK ebook collection: Kluwer Academic.

Genova, K., and Guliashki, V. (2011). "Linear Integer Programming Methods and Approaches – A Survey." *Cybernetics and Information Technologies* 11 (1): 3–25.

Geoffrion, A. M. (1974). "Lagrangian Relaxation and its Uses in Integer Programming." *Mathematical Programming Study* 2: 82–114.

Gilmore, P. C., and Gomory, R. E. (1963). "A Linear Programming Approach to the Cutting Stock Problem." *Operations Research* 11 (6): 863–88.

Glover, F. (1989). "Tabu search-part I." *ORSA Journal on computing* 1 (3): 190-206.

Glover, F. (1990). "Tabu search-part II." *ORSA Journal on computing* 2 (1): 4-32.

Glover, F., and G. A. Kochenberger, eds. (2003). *Handbook of metaheuristics: Chapter 13 - A Classification of Hyper-heuristic Approaches.* Boston, MA, USA: Kluwer Academic Publishers.

Goldratt, E. (1997). *Critical Chain:* North River Press.

Gomory, R. E. (1958). "Outline of an Algorithm for Integer Solutions to Linear Programs." *Bulletin of the American Mathematical Society* 64: 275–78.

Gonzalez-Quevedo, A. A., AbouRizk, S. M., Iseley, D. T., and Halpin, D. W. (1993). "Comparison of Two Simulation Methodologies in Construction." *Journal of Construction Engineering and Management ASCE* 119 (3): 573–89.

Gordon, V., Proth, J.-M., and Chu, C. (2002). "A survey of the state-of-the-art of common due date assignment and scheduling research." *European Journal of Operational Research* 139: 1–25.

Günthner, W., and A. Borrmann, eds. (2011). *Digitale Baustelle- innovativer Planen, effizienter Ausführen: Werkzeuge und Methoden für das Bauen im 21. Jahrhundert.* 1st ed. Berlin: Springer Berlin.

Hajjar, D., and AbouRizk, S. M. (1996). "Building a Special Purposes Simulation Tool for Earth Moving Operations." *Proceedings of the 1996 Winter Simulation Conference*, 1313–20.

Hajjar, D., and AbouRizk, S. M. (1998). "Modeling and Analysis of Aggregate Production Operations." *Journal of Construction Engineering and Management ASCE* 124 (5): 390–401.

Hajjar, D., and AbouRizk, S. M. (1999). "Simphony: an Environment for Building Special Purpose Construction Simulation Tools." *Proceedings of the 1999 Winter Simulation Conference*, 998–1006.

Hajjar, D., AbouRizk, S. M., and Xu, J. (1998). "Construction Site Dewatering Analysis Using a Special Purpose Simulation-based Framework." *Canadian Journal of Civil Engineering* 25: 819–28.

Halpin, D. W. (1977). "CYCLONE: Method for modeling of job site processes: Vol. 103, No. 3, September 1977, pp. 489-499." *Journal of the Construction Division* 3 (103): 489–99.

Halpin, W. D. (1990). *Micro-CYCLONE: User's Manual.* West Lafayette: Department of Civil Engineering, Purdue University.

Hamm, M., and König, M. (2010). "Constraint-based Multi-objective Optimization of Construction Schedules." In *Proceedings of the International Conference Computing in Civil and Building Engineering*. Vol. 30, 243.

Hartmann, S. (1997). "A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling." *Naval Research Logistics* 45: 733–50.

Hartmann, S. (2001). "Project Scheduling with Multiple Modes: A Genetic Algorithm." *Annals of Operations Research* 102: 111–35.

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.

Hartmann, V., Beucke, K. E., Shapir, K., and König, M. (2012). "Model-based Scheduling for Construction Planning." *Proceedings of the 14th International Conference on Computing in Civil and Building Engineering*.

Hegazy, T. (1999). "Optimization of resource allocation and leveling using genetic algorithms." *Journal of Construction Engineering and Management* 125 (3): 167–75.

Hegazy, T., and Menesi, W. (2010). "Critical Path Segments Scheduling Technique." *Journal of Construction Engineering and Management ASCE* 136 (10): 1078–85.

Hegazy, T., and Menesi, W. (2012). "Heuristic Method for Satisfying Both Deadlines and Resource Constraints." *J. Constr. Eng. Manage.* 138 (6): 688–96.

Henderson, S. G., and B. L. Nelson, eds. (2006). *Handbook in Operation Research and Management Science: Simulation: Vol. 13 – Chapter 21:* North Holland.

Hendrickson, C., and T. Au (1989). *Project management for construction: fundamental concepts for owners, engineers, architects, and builders.* Prentice-Hall civil engineering and engineering mechanics series: Prentice Hall.

Herroelen, W., E. Demeulemeester, and B. De Reyck. (1997). "A Classification Scheme for Project Scheduling Problems: Technical Report."

Hertz, A., and Widmer, M. (2003). "Guidelines for the Use of Meta-Heuristics in Combinatorial Optimization." *European Journal of Operational Research* 151: 247–52.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence:* MIT Press.

Hollermann, S., Melzner, J., and Bargstaedt, J. H. (2012). "Concept for Scheduling of Bridge Construction Processes Based on Distributed Simulation." *Proc. of the 19th Intl. EG-ICE Workshop*.

Hooper, J. W. (1986). "Strategy-related characteristics of discrete-event languages and models." *Simulation* 46 (4): 153–59.

Horenburg, T., Wimmer, J., and Günthner, W. A. (2012). "Resource Allocation in Construction Scheduling Based on Multi-Agent Negotiation." *Proceedings of the 14th International Conference on Computing in Civil and Building Engineering*.

IEEE 1516 - (2010). *IEEE Std 1516'-2010, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)„Framework and Rules"*.

Igelmund, G., and Radermacher, F. J. (1983a). "Algorithmic approaches to preselective strategies for stochastic scheduling problems." *Networks* 13: 29–48.

Igelmund, G., and Radermacher, F. J. (1983b). "Preselective strategies for the optimization of stochastic project networks under resource constraints." *Networks* 13: 1–28.

Ioannou, P. G. (1989). "UM-CYCLONE Discrete Event Simulation System - Reference Manual."

ISO 16739:2013 . *Industry Foundation Classes (IFC) for Data Sharing in the Construction and Facility Management Industries*, no. ISO 16739:2013.

Iwata, S. K., and Murota, K. (2001). "Combinatorial Relaxation Algorithm for Mixed Polinomial Matrices." *Math. Program.* 90: 353–71.

Jackson, J. R. (1957). "Simulation research on job shop production." *Naval Research Logistics Quarterly* 4 (4): 287–95.

Jarboui, B., Damak, N., Siarry, P., and Rebai, A. (2008). "A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems." *Applied Mathematics and Computation* 195 (1): 299–308.

Jensen, K. (1991). "Coloured Petri Nets: A High Level Language for System Design and Analysis." *Lecture Notes in Computer Science* 483: 342–416.

Jensen, K. (1996). *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use.* Coloured Petri Nets: Springer.

Jensen, P. A., and Jonathan F. Bard (2003). *Operations Research Models and Methods:* Wiley.

Kalk, A., and S.A. Douglas (1980). *Insight: Interactive Simulation of Construction Operations Using Graphical Techniques.* Technical report. Stanford University. Department of Civil Engineering: Stanford University, Department of Civil Engineering.

Kamat, V. R., and Martinez, J. C. (2003). "Validating Complex Construction Simulation Models Using 3D Visualization." *Systems Analysis Modelling Simulation* 43 (4): 455–67.

Kang, D. B., and Myint, Y. M. (1999). "Time, Cost and Quality Trade-off in Project Management: a Case Study." *International Journal of Project Management* 17 (4): 249–56.

Karmarkar, N. (1984). "A new polinomial-time algoritm for linear programming." *Combinatorica* 4 (4): 373–95.

Kelley, J. E., and Walker, M. R. (1961). "Critical-Path Planning and Scheduling: Mathematical Basis: Operations Research May/June 1961 9:296-320." *Operations Research* 9 (3): 296–320.

Kelley, J. E., and Walker, M. R. (1989). "The Origins of CPM: a Personal History." *PM Network* 3 (2): 7–22.

Kennedy, J., and Eberhart, R. (1995). "Particle swarm optimization." In *IEEE International Conference on Neural Networks*. Vol. 4, 1942-1948.

Kerzner, H. (2003). *Project Management: A Systems Approach to Planning, Scheduling, and Controlling, 8th Edition.* Hoboken, New Jersey: John Wiley & Sons, Inc.

Khachiyan, L. G. (1979). "A Polynomial Algorithm in Linear Programming." *Doklady Akademiia Nauk SSSR* 244: 1093–96.

Khisty, C. J. (1970). "The Application of the Line of Balance Technique to the Construction Industry." *Indian Concrete Journal* 44 (7): 297-300, 319-320.

Kim, K., and de la Garza, J. (2003). "Phantom Float." *Journal of Construction Engineering and Management ASCE* 129 (5): 507–17.

Kim, K., and de la Garza, J. M. (2005). "Evaluation of the Resource-Constrained Critical Path Method Algorithms." *Journal of Construction Engineering and Management ASCE* 131 (5): 522–32.

Kim, K., and Kim, K. J. (2010). "Multi-agent-based simulation system for construction operations with congested flows." *Automation in Construction* 19 (7): 867–74.

Kirkpatrick, S., Gelatt Jr, C., and Vecchi, M. (1983). "Optimization by simulated annealing." *science* 220 (4598): 671-680.

Klein, R. (2000). *Scheduling of Resource-constrained Projects.* Operations Research-Computer Science Interfaces Series: Kluwer Academic Publishers.

Knotts, G., Dror, M., and Hartman, B. C. (2000). "Agent-Based Project Scheduling." *IIE Transactions* 32: 387–401.

Kolisch, R. (1996). "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation." *European Journal of Operational Research* 90 (2): 320–33.

Kolisch, R., and Hartmann, S. (1999). "Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis." In *Handbook on Recent Advances in Project Scheduling*. http://link.springer.com/chapter/10.1007/978-1-4615-5533-9_7.

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., Sprecher, R., and Drexl, A. (1995). "Characterization and Generation of a General Class of a Resource-Constrained Project Scheduling Problems." *Management Science* 41: 1693–703.

Koné, O., Artigues, C., Lopez, P., and Mongeau, M. (2011). "Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources." *Computers & Operations Research* 38: 1–17.

König, M., and Beißert, U. (2009). "Construction Scheduling Optimization by Simulated Annealing." *International Symposium on Automation and Robotics in Construction*, 183–90.

König, M., Beißert, U., and Bargstädt, H.-J. (2009a). "Knowlegde-Based Simulation and Optimization of Construction Processes." *EG-ICE Conference 2009*.

König, M., Beißert, U., and Bargstaedt, H. J. (2007a). "Ereignis diskrete Simulation von Trockenbauarbeiten - Konzept, Implementierung und Anwendung." *Schriftreihe Bauwirtschaft - III Tagungen und Berichte 4 - 1. IBW Workshop Simulation in der Bauwirtschaft*, 15–28.

König, M., Beißert, U., and Bargstaedt, H. J. (2009b). "Knowledge-Based Simulation and Optimization of Construction Processes." *Proceedings of the EG-ICE Conference 2009*.

König, M., Beißert, U., Steinhauer, D., and Bargstaedt, H. J. (2007b). "Constraint-Based Simulation of Outfitting Processes in Ship Building and Civil Engineering." *6th Eurosim Congress in Modelling and Simulation*.

Koo, B., and Fischer, M. (2003). "Formalizing Construction Sequencing Constraints for Rapid Generation of Schedule Alternatives."

Kooragamage, R., Elhag, T., Kelsey, J., and Julier, S. (2013). "Using Agent-Based Simulation to Manage Logistics for Earthmoving Operations in Construction." *CIB World Building Congress, Kuala Lumpur, Malaysia*.

Kugler, M., and Franz, V. (2007). "Entwurf eines multiagentenbasierten Referenzmodells für Simulationen im Hochbau." *Schriftreihe Bauwirtschaft - III Tagungen und Berichte 4 - 1. IBW Workshop Simulation in der Bauwirtschaft*, 69–84.

Ladányi, L., Ralphs, T. K., and Trotter Jr., L. E. (2001) "Branch, Cut, and Price: Sequential and Parallel: Lecture Notes in Computer Science." In *Computational Combinatorical Optimization*, 223–60.

Lai, K. K., and Li, L. (1999). "A Dynamic Approach to Multiple-Objective Resource Allocation Problem." *European Journal of Operational Research* 117: 293–309.

Land, A. H., and Doig, A. G. (1960). "An Automatic Method for Solving Discrete Programming Problems." *Econometrica: journal of the Econometric Society* 28.

Lim, A., Ma, H., Rodrigues, B., Teck Tan, S., and Xiao, F. (2011). "New Concepts for Activity Float in Resource-constrained Project Management." *Computers & Operations Research* 38 (6): 917–30.

Lin, J. T., and Lee, C.-C. (1993). "A Three-phase Discrete Event Simulation with EPNSim Graphs." *Simulation* 60 (6): 382–92.

Liu, L. Y., and Ioannou, P. G. (1992). “Graphical Object-Oriented Discrete Event Simulation System.” *Proceedings of the 1992 Winter Simulation Conference*, 1285–91.

Long, L. D., and Ohsato, A. (2009). “A genetic algorithm-based method for scheduling repetitive construction projects.” *Automation in Construction* 18 (4): 499–511.

Lu, M. (2003). “Simplified Discrete-Event Simulation Approach for Construction Simulation.” *Journal of Construction Engineering and Management* 129 (5): 537–46.

Lu, M., Kim, K., and de la Garza, J. M. (2006). “Discussion and Closure to "Evaluation of the Resource-Constrained Critical Path Method Algorithms" by Kyunghwan Kim and Jesús M. de la Garza.” *Journal of Construction Engineering and Management ASCE* 132 (10): 1126–28.

Lu, M., and Lam, H.-C. (2008). “Critical Path Scheduling under Resource Calendar Constraints.” *Journal of Construction Engineering and Management*, 25–31.

Lu, M., and Li, H. (2003). “Resource-Activity Critical-Path Method for Construction Planning.” *Journal of Construction Engineering and Management* 129 (4): 412–20.

Lumsden, P. (1968). *The Line of Balance Method.* London, UK: Pergamon Press, Inc.

Lutz, D. J. (1990). *Planning of Linear Construction Projects using Simulation and Line of Balance: Ph.D. Dissertation.* Lafayette: School of Civil Engineering, Purdue University.

M. König. (2004). “Ein Prozessmodell für die kooperative Gebäudeplanung.” *Ph.D. Dissertation* Ruhr Universität Bochum, Shaker Verlag.

Martinez, J., and Ioannou, P. G. (1994). “General Purpose Simulation with Stroboscope.” *Winter Simulation Conference 1994*, 1159–66.

Martinez, J., Ioannou, P. G., and Carr, R. I. (1994). “State And REsource Based Construction Process Simulation.” *Proceedings of the First Congress on Computing in Civil Engineering, ASCE, Washington, D.C.*, 177–84.

Martinez, J. C. (2001). “EZStrobe - General-Purpose Simulation System Based on Activity Cycle Diagrams.” *Proceedings of the 2001 Winter Simulation Conference*, 1556–64.

Martinez, J. C., and Ioannou, P. G. (1999). “General-Purpose Systemes for Effective Construction Simulation.” *Journal of Construction Engineering and Management ASCE* 125: 265–76.

Marx, A., and König, M. (2011). “Preparation of Constraints for Construction Simulation.” *Proceedings of the 2011 ASCE International Workshop on Computing in Civil Engineering*, 462–69.

Mati, Y. (2010). “Minimizing the makespan in the non-preemptive job-shop scheduling with limited machine availability.” *Computers & Industrial Engineering* 59 (4): 537–43.

McCahill, D. F., and Bernold, L. E. (1993). “Resource-Oriented Modeling and Simulation in Construction.” *Journal of Construction Engineering and Management ASCE* 119 (3): 590–606.

Melzner, J., Hollermann, S., and Bargstaedt, H. J. (2011). “Detailed Input data Source for Construction Process Simulation.” *Proceedings of the Third International Conference on Advances in System Simulation*, 132–35.

Melzner, J., Hollermann, S., Elmahdi, A., Hong, H. L., and Bargsaedt, H. J. (2012). “Pattern-Based Process Modeling for Construction Management.” *Proceedings of the 14th International Conference on Computing in Civil and Building Engineering*.

Merkle, D., Middendorf, M., and Schmeck, H. (2002). "Ant colony optimization for resource-constrained project scheduling." *Evolutionary Computation, IEEE Transactions on* 6 (4): 333–46.

Michalewicz, Z., and David B. Fogel (2004). *How to solve it: modern heuristics // How to solve it: Modern heuristics.* 2nd ed. Berlin,, New York: Springer-Verlag New York Inc; Springer.

Mikulakova, E., König, M., Tauscher, E., and Beucke, K. (2010). "Knowledge-based schedule generation and evaluation." *Advanced Engineering Informatics* 24 (4): 389–403.

Ming Lu, H.-C. L. F. D. (2008). "Respurce-Constrained Critical Path Analysis Based on Discrete Event Simulation and Particle Swarm Optimization." *Automation in Construction* (17): 670–81.

Mingozzi, A., and Maniezzo, V. (1998). "An Exact Algorithm for the Resource Constrained Project Scheduling Problem Based on a New Mathematical Formulation." *Management Science* 44: 714-729.

Mitchell, J. E. (2002). *Branch-and-Cut Algorithms for Combinatorial Optimization Problems:* Oxford University Press: 65–77.

Mohamed, Y., and AbouRizk, S. (2005). "Framework for Building Intelligent Simulation Models of Construction Operations.".

Möhring, R. H., Skutella, M., and Stork, F. (2004). "Scheduling with AND/OR Precedence Constraints." *SIAM J. Comput* 33 (2): 393–415.

Montoya-Torres, J. R., Gutierrez-Franco, E., and Pirachicán-Mayorga, C. (2010). "Project scheduling with limited resources using a genetic algorithm." *International Journal of Project Management* 28 (6): 619–28.

NAVMAT (1962). "Line of Balance Technology." *Washington DC: Naval Material Command: NAVMAT.*

Nelson, R. T. (1958). "Waiting-Time Distributions for Application to a Series of Service Centers." *Operations Research* 6 (6): 856–62.

Ng, S., and Zhang, Y. (2008). "Optimizing Construction Time and Cost Using Ant Colony Optimization Approach." *Journal of Construction Engineering and Management* 134 (9): 721–28.

O´Brian, J. J. (1975). "VPM Scheduling for High-Rise Buildings." *Journal of Construction Division* (101:4): 895–905.

O'Brien, J. J., and F. L. Plotnick (2009). *CPM in Construction Management, Seventh Edition:* McGraw-Hill.

Olaguibel, R. A.-V., and Goerlich, J. M. T. (1993). "The Project Scheduling Polyhedron: Dimension, Facets and Lifting Theorem." *European Journal of Operational Research* 67: 204–20.

Ören, T. I., Numrich, S. K., Uhrmacher, A. M., Wilson, L. F., and Gelenbe, E. (2000). "Agent-Directed Simulation - Challenges to Meet Defense and Civilian Requirements." *Proceedings of the 2000 Winter Simulation Conference*, 1757–62.

Page, B., and Kreutzer, W. (2005). *The Java Simulation Handbook - Simulating Discrete Event Systems with UML and Java.* Aachen: Shaker Verlag.

Papadimitriou, C. H., and Steiglitz, I. (1998). *Combinatorial Optimization - Algorithms and Complexity*. United States of America: Dover Publications Inc.

Patterson, J., Slowinski, R., Talbot, F. B., and Weglarz, J. (1989). "An Algorithm for a General Class of Precedence and Resource Constrained Scheduling Problems." *Advances in Project Scheduling*, 3–28.

Petri, C. A. (1962). "Kommunikation mit Automaten." Dissertation, Institut für Angewandte Mathematik, Universität Bonn.

Petri, C. A. (1966). "Communication with Automata." In *Technical Report RADC-TR-65-377, 1, 1*.

Powell, W. B., Shapiro, J. A., and Simão, H. P. (2002). "An Adaptive Dynamic Programming Algorithm for the Heterogeneous Resource Allocation Problem." *Transportation Science* 36 (2): 231–49.

Pritsker, A., Watters, L., and Wolfe, P. (1969). "Multi-Project Scheduling with Limited Resources: A Zero-One Programming Approach." *Management Science* 16: 93–108.

R. Alvarez-Valdes, E. Crespo, J.M. Tamarit, and F. Villa (2008). "GRASP and path relinking for project scheduling under partially renewable resources." *European Journal of Operational Research* 189 (3): 1153–70.

Rahm, T., Sadri, K., Thewes, M., and König, M. (2012). "Multi-Method Simulation of the Excavation Process in Mechanized Tunneling." *Proc. of the 19th Intl. EG-ICE Workshop*.

Ranjbar, M. (2012). "A Hybrid GRASP Algorithm for Minimizing Total Weighted Resource Tardiness Penalty Costs is Scheduling of Project Networks." *International Journal of Industrial Engineering & Production Research* 23 (3): 231–43.

Ranjbar, M., Kianfar, F., and Shadrokh, S. (2008). "Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm." *Applied Mathematics and Computation* 196 (2): 879–88.

Rao, S. S. (2009). *Engineering optimization: Theory and Practice:* John Wiley & Sons, New Jersey. 4th Edition.

Raz, T., and Marshall, B. (1996). "Effect of Resource Constraints in Float Calculations in Project Networks." *International Journal of Project Management* 14 (4): 241–48.

Rossi, F., P. van Beek, and T. Walsh (2006). *Handbook of Constraint Programming.* Amsterdam: Elsevier. Foundations of Artificial Intelligence.

Roy, M., and M. Sussmann (1969). *Les problèmes d'ordonnancement avec contraintes disjonctives.* Paris: SEMA notes DS.

Sawhney, A. (1997). "Petri net based simulation of construction schedules." *Proceedings of the 1997 Winter Simulation Conference*, 1111–18.

Sawhney, A., and Vamadevan, A. (2000). "Petri Net-Based Scheduling of a Bridge Project." *Construction Congress VI*, 107–14.

Sawhney, A., Walsh, K., and Mulky, A. R. (2003). "Agent-Based Modeling and Simulation in Construction." *Proceedings of the 2003 Winter Simulation Conference*, 1541–47.

Shannon, R. (1975). *Systems Simulation: The Art and Science.* Englewood Cliffs, N.J: Prentice Hall.

Shen, W., and Norrie, D. H. (1999). "Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey." *Knowledge and Information Systems* 1: 129–56.

Shi, J. J. (1999). "Activity-Based Construction (ABC) Modeling and Simulation Method." *Journal of Construction Engineering and Management ASCE* 125 (5): 354–60.

Shmuel, A. (1954). "The relaxation method for linear inequalities." *Canadian Journal of Mathematics* 6: 382–92.

Skowronski, M. E., Myszkowski, P. B., Adamski, M., and Kwiatek, P. (2013). "Tabu Search approach for Multi-Skill Resource-Constrained Project Scheduling Problem." *Proceedings of the 2013 Federal Conference on Computer Science and Information Systems*, 153–58.

Smith, J. S. (2003). "Survey on the Use of Simulation for Manufacturing System Design and Operation." *Journal of Manufacturing Systems* 22 (2): 157–71.

Snyder, J. R. (1987). "Modern Project Management: How Did We Get Here - Where Do We Go?" *Project Management Journal* 18 (1): 28–29.

Sprecher, A., Kolisch, R., and Drexl, A. (1995). "Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem." *European Journal of Operational Research* 80 (1): 94–102.

Sriprasert, E., and Dawood, N. (2002). "Requirements Identification for 4D Constraint-based Construction Planning and Control System." *Proceedings of CIB-W78 International Conference on Information Technology in Construction*.

Sriprasert, E., and Dawood, N. (2003). "Genetic Algorithms for Multi-Constraint Scheduling: An Application for the Construction Industry." *Proceedings of CIB-W78 International Conference on Information Technology in Construction*.

Stinson, J. P., Davis, E. W., and Khumawala, B. H. (1978). "Multiple Resource-Constrained Scheduling Using Branch and Bound." *A I I E Transactions* 10: 252–59.

Stinson, J. P., Davis, E. W., and Khumawala, B. M. (1978). "Multiple Resource–Constrained Scheduling Using Branch and Bound." *A I I E Transactions* 10 (3): 252–59.

Stork, F., and Uetz, M. (2005). "On the generation of circuits and minimal forbidden sets." *Math. Program* 102 (1): 185–203.

Stretton, A. (2007). "A Short History of Modern Project Management." *PM World Today* 9 (10): 1–18.

Syswerda, G. (1991). "Schedule Optimization Using Genetic Algorithms." In *Handbook of Genetic Algorithms*. Edited by Lawrence Davis. New York, NY: Van Nostrand Reinhold.

Szczesny, K., Hamm, M., Koch, C., and König, M. (2012). "A rank-based crossover operator for evolutionary algorithms for simulation-based optimization of construction schedules." In *Proc. of EG-ICE 2012*.

Tareghian, H. R., and Taheri, S. H. (2006). "On the discrete time, cost and quality trade-off problem." *Applied Mathematics and Computation* 181 (2): 1305–12.

Tauscher, E., Mikulakova, E., König, M., and Beucke, K. (2007). "Generating Construction Schedules with Case-Based Reasoning Support." *Computing in Civil Engineering ASCE*, 119–26.

Taylor, F. W. (1903). "Shop Management." *ASME Transactions* 24: 1337–480.

Tommelein, I. (1998). "Pull-Driven Scheduling for Pipe-Spool Installation: Simulation of Lean Construction Technique." *Journal of Construction Engineering and Management ASCE* 124 (4): 279–88.

Tommelein, I. D., and Odeh, A. M. (1994). "Knowledge-Based Assembly of Simulation Networks Using Construction Designs, Plans, and Methods." *Proceedings of the 1994 Winter Simulation Conference*, 1145–52.

Tulke, J. (2010). "Kollaborative Terminplanung auf Basis von Bauwerksinformationsmodellen." PhD Thesis, Bauhaus Universität Weimar.

Tulke, J., and Hanff, J. (2007). "4D Construction sequence planning." New-Process and Data Model. *Proceedings of CIB-W78 24th International Conference on Information Technology in Construction* (24th int. conf. Maribor, Slov. 2007): 79–84.

Valls, V., Ballestín, F., and Quintanilla, S. (2005). "Justification and RCPSP: A technique that pays." *European Journal of Operational Research* 165 (2): 375–86.

Wakefield, R. R., and Sears, G. A. (1997). "Petri nets for simulation and modeling of construction systems." *Journal of Construction Engineering and Management ASCE* 123 (2): 105–12.

Wales, R., and AbouRizk, S. (1996). "An integrated simulation model for construction." *Simulation Practice and Theory* 3: 401–20.

Wang, H., Zhang, J., Chau, K., and Anson, M. (2004). "4D dynamic management for construction planning and resource utilization." *Automation in Construction* 13 (5): 575–89.

Weaver, P. (2007). "The Origins of Project Management." *Fourth Annual PMI College of Scheduling Conference*.

Wiest, J. D. (1982). "Precedence Diagramming Method: Some Unusual Characteristics and Their Implications for Project Managers." *Journal of Operations Management* 1 (3): 121–30.

Willis, R. J. (1985). "Critical path analysis and resource constrained project scheduling — Theory and practice." *European Journal of Operational Research* 21: 149–55.

Wilson, J. M. (2003). "Gantt charts: A centenary appreciation." *European Journal of Operational Research* 149 (2): 430–37.

Wu, I.-C., Borrmann, A., Beißert, U., König, M., and Rank, E. (2010a). "Bridge construction schedule generation with pattern-based construction methods and constraint-based simulation." *Advanced Engineering Informatics* 24: 379–88.

Wu, I.-C., Borrmann, A., Beißert, U., König, M., and Rank, E. (2010b). "Bridge construction schedule generation with pattern-based construction methods and constraint-based simulation." *Advanced Engineering Informatics* 24 (4): 379–88.

Xie, H., AbouRizk, S. M., and Fernando, S. (2011). "Integrating realtime project progress input into a construction simulation model." *Proceedings of the 2011 Winter Simulation Conference*, 3443–54.

Xu, J., and AbouRizk, S. M. (1999). "Product - based Model Fepresentation for Integrating 3D CAD with Computer Simulation." *Proceedings of the 1999 Winter Simulation Conference*, 971–77.

Yang, Z.-y., and Wang, Z.-f. (2010). "Comparison between AON and AOA Network Diagrams." *IEEE International Conference on Industrial Engineering and Engineering Management*, 1507–09.

Zapata, J. C., Hodge, B. M., and Reklaitis, G. V. (2008). "The Multimode Resource Constrained Multiproject Scheduling Problem: Alternative Formulations: AIChE Journal, 54(8):2101–2119, 2008." *AIChE Journal* 54 (8): 2101–19.

Zhang, H., Li, H., and Tam, C. M. (2006). "Particle swarm optimization for resource-constrained project scheduling." *International Journal of Project Management* 24 (1): 83–92.

Zhang, H., Tam, C., and Li, H. (2005). "Activity Object-Oriented Simulation Strategy for Modeling Construction Operations." *Journal of Computing in Civil Engineering ASCE* 19 (3): 313–22.

Zhang, H., Xu, H., and Peng, W. (2008). "A Genetic Algorithm for Solving RCPSP." In *Proceedings of the 2008 International Symposium on Computer Science and Computational Technology - Volume 02*, 246–49. ISCSCT '08. Washington, DC, USA: IEEE Computer Society. http://dx.doi.org/10.1109/ISCSCT.2008.255.

Zhang, J., Anson, M., and Wang, Q. (2000). "A New 4DManagement Approach to Construction Planning and Site Space Utilization. (2000): pp. 15-22." *Computing in Civil and Building Engineering*, 15–22.

Zheng, D., Ng, S., and Kumaraswamy, M. (2004). "Applying a Genetic Algorithm-Based Multiobjective Approach for Time-Cost Optimization." *Journal of Construction Engineering and Management* 130 (2): 168–76.

Zhou, J., Love, P. E. D., Wang, X., Teo, K. L., and Irani, Z. (2013). "A Review of Methods and Algorithms for Optimizing Construction Scheduling." *Journal of the Operational Research Society* 64 (8): 1091–105.

Zhou, W., Heesom, D., Georgakis, P., Nwagboso, C., and Feng, A. (2009). "An Interactive Approach to Collaborative 4D Construction Planning." *Journal of Information Technology in Construction* 14: 30–47.

Appendix

List of new constraints for the in Figure 5-21 introduced test case

Superstructure1/Girder2/Transport depends on Abutment2/Pour Concrete
Superstructure1/Girder2/Transport depends on Pier1/ Pour Concrete
Superstructure1/Girder2/Transport depends on Abutment1 /Pour Concrete
Superstructure1/Girder2/Transport depends on Superstructure1/Scaffold

Superstructure1/Girder1/Transport depends on Abutment2/Abutment2/Pour Concrete
Superstructure1/Girder1/Transport depends on Pier1/Pier1/Pour Concrete
Superstructure1/Girder1/Transport depends on Abutment1/Abutment1/Pour Concrete

Pier/Foundation depends on Abutment1/Construct Formwork
Pier/Foundation depends on Abutment1/Tie Rebar
Pier/Foundation depends on Abutment2/Remove Formwork

Abutment2/Foundation depends on Pier1/Tie Rebar
Abutment2/Foundation depends on Pier1/Create Formwork
Abutment2/Foundation depends on Abutment1/Remove Formwork

Abutment2/Tie rebar depends on Pier1/Pour Concrete

Abutment1/Foundation depends on Pier1/Pier1/Tie Rebar

Abutment1/Foundation depends on Pier1/Pier1/Create Formwork
Abutment1/Foundation depends on Abutment2/Remove Formwork

Abutment2/Create Formwork depends on Pier1/Pour Concrete
Abutment2/Create Formwork depends on Abutment1/Pour Concrete

Pier1/Head Beam depends on Abutment2/Remove Formwork
Pier1/Head Beam depends on Superstructure1/Bearings
Pier1/Head Beam depends on Abutment1/Remove Formwork

Abutment2/Pour Concrete depends on Pier1/Remove Formwork
Abutment2/Pour Concrete depends on Abutment1/Remove Formwork

Abutment1/Formwork depends on Superstructure1/Scaffold

Abutment2/Remove Formwork depends on Superstructure1/Girder2/Lift
Abutment2/Remove Formwork depends on Superstructure1/Girder1/Lift

Abutment1/Remove Formwork depends on Parapet2/Pour Concrete
Abutment1/Remove Formwork depends on Parapet1/Pour Concrete

Superstructure1/Girder2/Lift depends on Parapet1/Remove Formwork

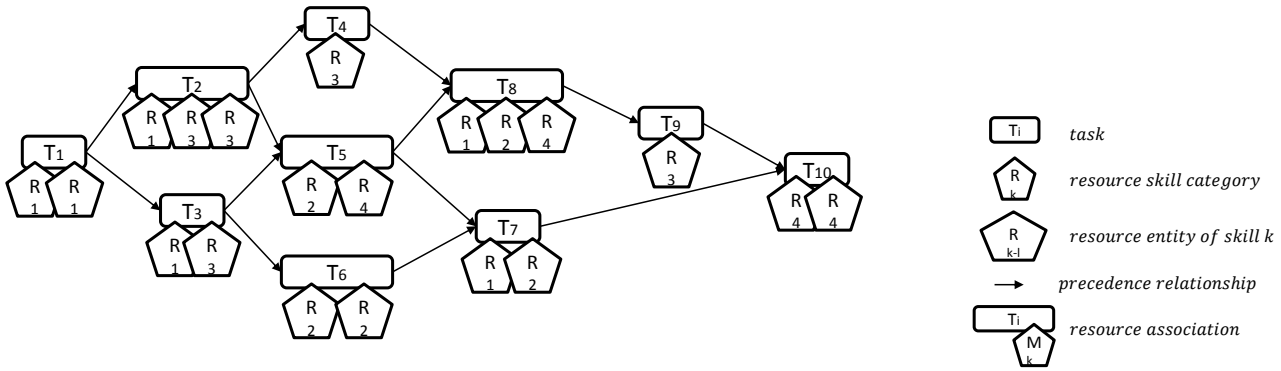
Superstructure1/Girder1/Lift depends on Parapet2/Remove Formwork

Parapet1/Pour Concrete depends on Superstructure1/Girder Heads

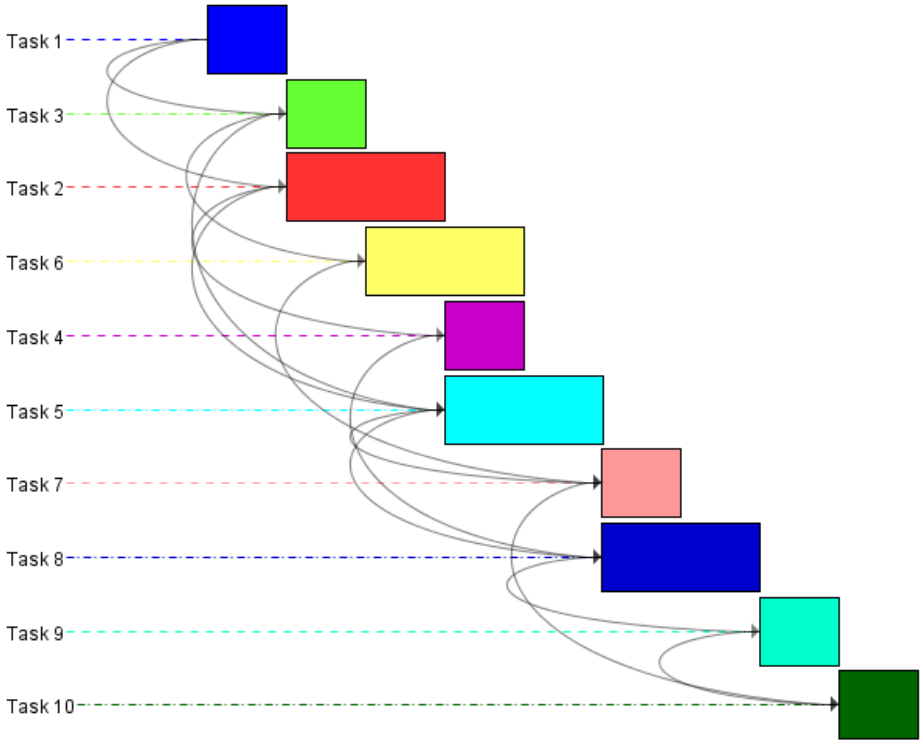
Results to the verification and validation of the introduced total float determination methods in Section 5.3 and 5.4

Precedence Graph:

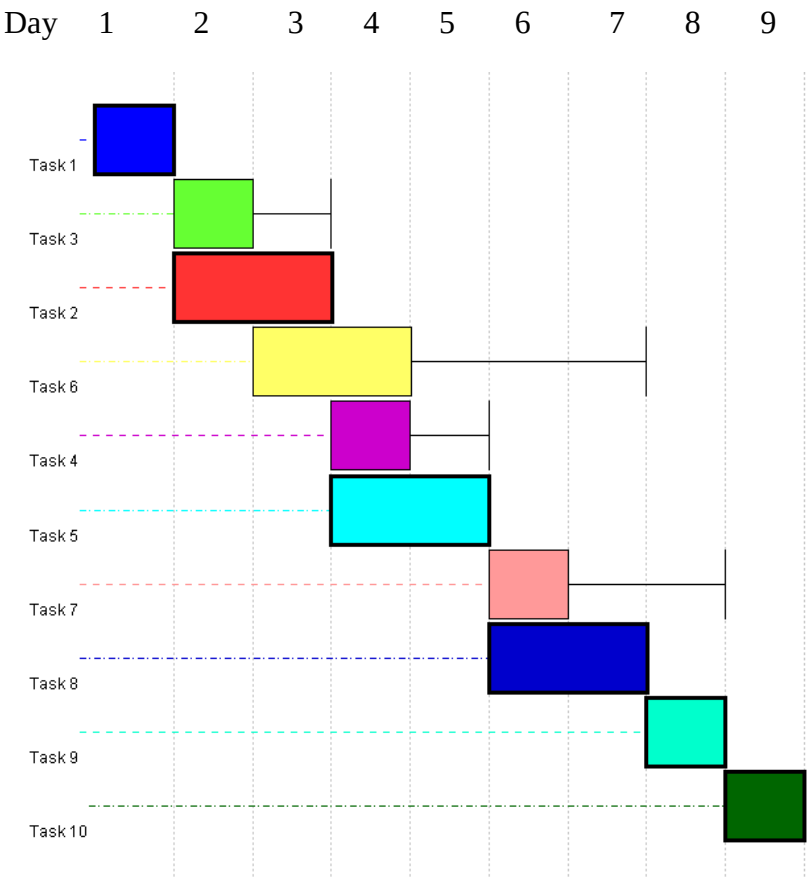
construction project scheduling problem under resource constraints



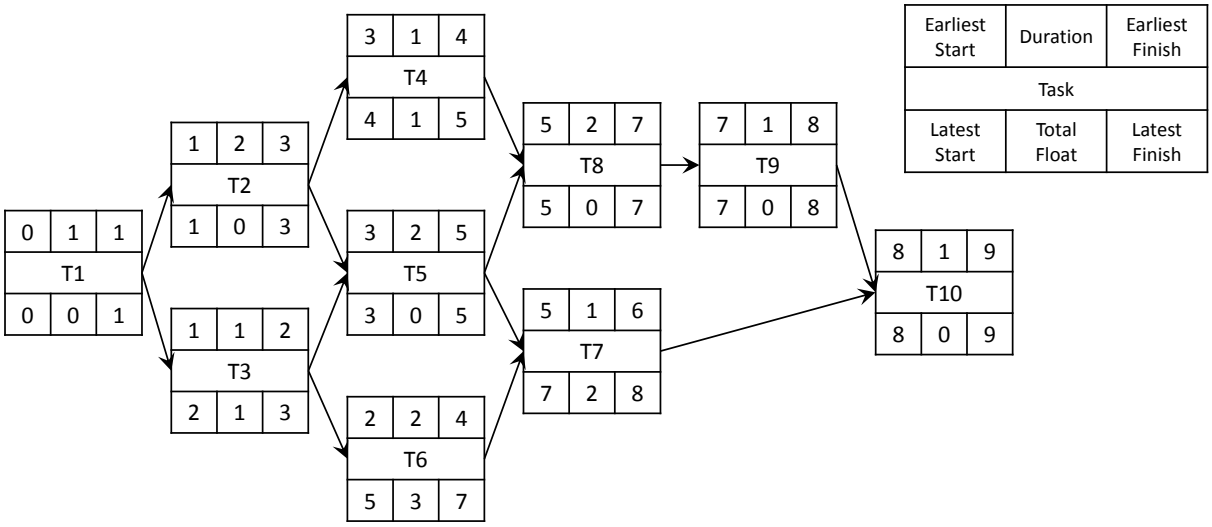
Results of the forward simulation (available resources: unlimited):



Results of the total float determination:

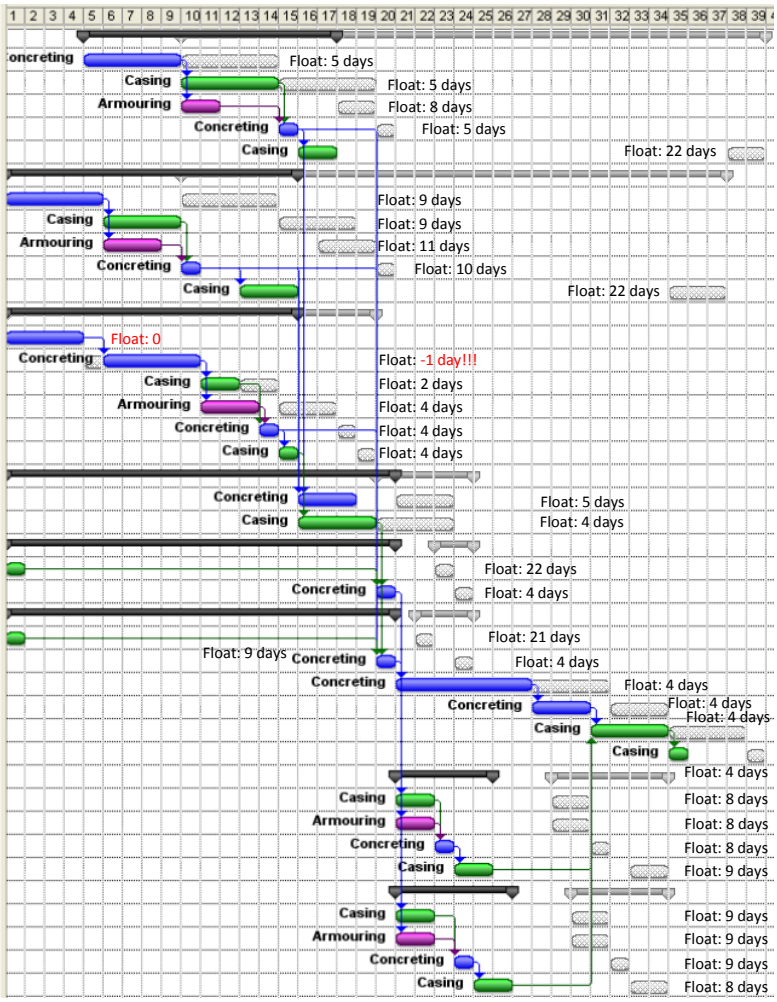


Results determined by the PDM:



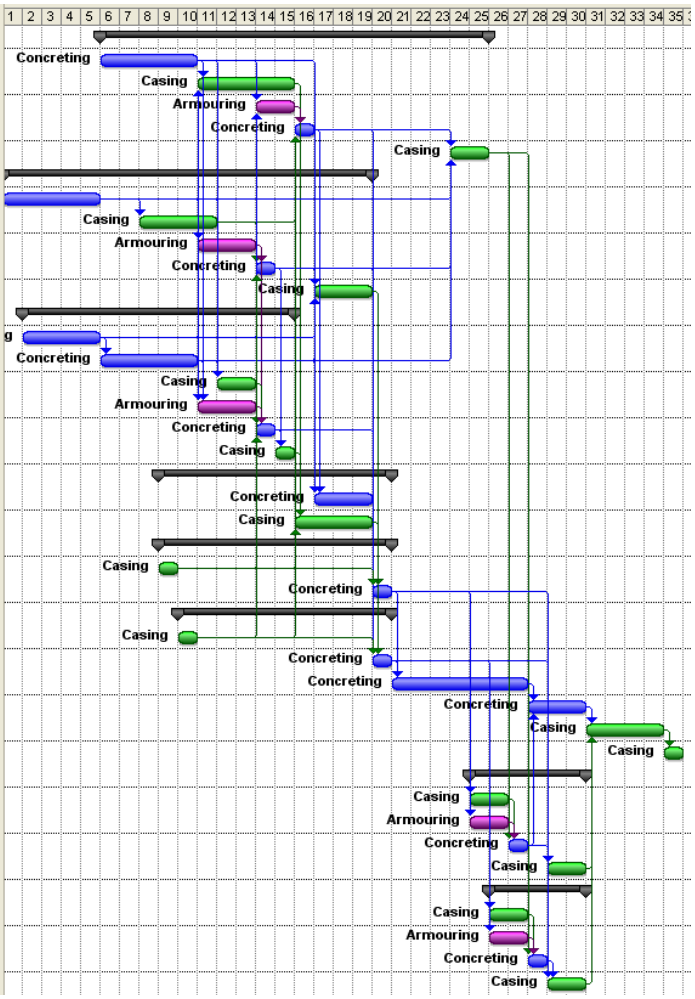
Results of the total float determination without the sequence enforcement constraints for the test case introduced in Figure 5-21

	Task Name	Duration	Resource Names
1	**Abutment 1**	**13 days**	
2	Foundation	5 days	Concreting
3	Construct Formwork	5 days	Casing
4	Tie Rebar	2 days	Armouring
5	Pour Concrete	1 day	Concreting
6	Remove Formwork	2 days	Casing
7	**Abutment 2**	**15 days**	
8	Foundation	5 days	Concreting
9	Construct Formwork	4 days	Casing
10	Tie Rebar	3 days	Armouring
11	Pour Concrete	1 day	Concreting
12	Remove Formwork	3 days	Casing
13	**Pier**	**15 days**	
14	Foundation	4 days	Concreting
15	Head Beam	5 days	Concreting
16	Construct Formwork	2 days	Casing
17	Tie Rebar	3 days	Armouring
18	Pour Concrete	1 day	Concreting
19	Remove Formwork	1 day	Casing
20	**Superstructure**	**20 days**	
21	Bearings	3 days	Concreting
22	Scaffold	4 days	Casing
23	**Girder 1**	**20 days**	
24	Transport	1 day	Casing
25	Lift	1 day	Concreting
26	**Girder 2**	**20 days**	
27	Transport	1 day	Casing
28	Lift	1 day	Concreting
29	Deck	7 days	Concreting
30	Girder Heads	3 days	Concreting
31	Sealing	4 days	Casing
32	Remove Scaffold	1 day	Casing
33	**Parapet 1**	**5 days**	
34	Construct Formwork	2 days	Casing
35	Tie Rebar	2 days	Armouring
36	Pour Concrete	1 day	Concreting
37	Remove Formwork	2 days	Casing
38	**Parapet 2**	**6 days**	
39	Construct Formwork	2 days	Casing
40	Tie Rebar	2 days	Armouring
41	Pour Concrete	1 day	Concreting
42	Remove Formwork	2 days	Casing



Results of the backward simulation with the sequence enforcement constraints for the case study introduced in Figure 5-21

	Task Name	Duration	Resource Names	Predecessors
1	**Abutment 1**	**20 days**		
2	Foundation	5 days	Concreting	
3	Construct Formwork	5 days	Casing	2;14
4	Tie Rebar	2 days	Armouring	2;14
5	Pour Concrete	1 day	Concreting	3;4;24;27;9
6	Remove Formwork	2 days	Casing	5;8;15;11
7	**Abutment 2**	**19 days**		
8	Foundation	5 days	Concreting	
9	Construct Formwork	4 days	Casing	8
10	Tie Rebar	3 days	Armouring	8
11	Pour Concrete	1 day	Concreting	9;10;24;27
12	Remove Formwork	3 days	Casing	11;14;2;15
13	**Pier**	**14 days**		
14	Foundation	4 days	Concreting	
15	Head Beam	5 days	Concreting	14
16	Construct Formwork	2 days	Casing	15;8;2
17	Tie Rebar	3 days	Armouring	15;8;2
18	Pour Concrete	1 day	Concreting	16;17;24;27;10;9
19	Remove Formwork	1 day	Casing	18;11
20	**Superstructure**	**12 days**		
21	Bearings	3 days	Concreting	5;11;15
22	Scaffold	4 days	Casing	19;27;3
23	**Girder 1**	**12 days**		
24	Transport	1 day	Casing	
25	Lift	1 day	Concreting	24;5;18;22;12
26	**Girder 2**	**11 days**		
27	Transport	1 day	Casing	
28	Lift	1 day	Concreting	27;18;11;22;12
29	Deck	7 days	Concreting	25;28
30	Girder Heads	3 days	Concreting	29;36
31	Sealing	4 days	Casing	30;37;42
32	Remove Scaffold	1 day	Casing	31
33	**Parapet 1**	**6 days**		
34	Construct Formwork	2 days	Casing	25
35	Tie Rebar	2 days	Armouring	25
36	Pour Concrete	1 day	Concreting	34;35;6
37	Remove Formwork	2 days	Casing	36;28
38	**Parapet 2**	**5 days**		
39	Construct Formwork	2 days	Casing	28
40	Tie Rebar	2 days	Armouring	28
41	Pour Concrete	1 day	Concreting	39;40;6
42	Remove Formwork	2 days	Casing	41;25



List of priorities for the case study in Section 7.5.4

Abutment 1/ Pour concrete	19
Abutment 1/ Create Formwork	26
Abutment 1/ Remove Formwork	17
Abutment 1/ Tie Rebar	25
Abutment 1/ Foundation	30
Abutment 2/ Pour concrete	24
Abutment 2/ Create Formwork	29
Abutment 2/ Remove Formwork	21
Abutment 2/ Tie Rebar	28
Abutment 2/ Foundation	32
Parapet 1/ Pour concrete	7
Parapet 1/ Create Formwork	11
Parapet 1/ Remove Formwork	6
Parapet 1/ Tie Rebar	10
Parapet 2/ Pour concrete	5
Parapet 2/ Create Formwork	9
Parapet 2/ Remove Formwork	4
Parapet 2/ Tie Rebar	8
Pier 1/ Pour concrete	20
Pier 1/ Create Formwork	23
Pier 1/ Remove Formwork	18
Pier 1/ Tie Rebar	22
Pier 1/ Foundation	31
Pier 1/ Head Beam	27
Superstructure/ Bearings	16
Superstructure/ Scaffold	15
Superstructure/ Girder 1/ Transport	34
Superstructure/ Girder 1/ Lift	14
Superstructure/ Girder 2/ Transport	33
Superstructure/ Girder 2/ Lift	13
Superstructure/ Deck	12
Superstructure/ Girderheads	3
Superstructure/ Sealing	2
Superstructure/ Remove Scaffold	1

Input data for the second optimization case study in Section 7.6.2 as XML file

See https://www.cms.bgu.tum.de/de/team/46-team/161-dori/...