Modeling and Optimizing the Evacuation of Hospitals based on the RCPSP with Resource Transfers

Doctoral Thesis
(Dissertation)

to be awarded the degree
Doctor rerum naturalium (Dr. rer. nat.)

submitted by

Jens Poppenborg
from Nordhorn, Germany

approved by the Faculty of
Mathematics/Computer Science and Mechanical Engineering,
Clausthal University of Technology,

Date of oral examination
11th March 2014

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.dnb.de.

D 104

1. Auflage, 2014

ISBN 978-3-86948-336-8

Contents

1 Introduction

The partial or complete evacuation of a hospital or another healthcare facility might become necessary in various situations. These include, for example, internal dangers such as a fire or a hazardous materials spill, as well as external dangers such as a flood or a bomb disposal. In particular the latter is frequently the case in Germany if an aircraft bomb from World War II has to be disposed, e.g. in April 2013 when a hospital as well as a nursing home had to be evacuated due to a bomb disposal in Duisburg or in November 2013 when a hospital as well as two nursing homes had to be evacuated due to a bomb disposal in Dortmund. These scenarios generally involve a lot of pre-evacuation planning both inside as well as outside the healthcare facilities because assistants, transport vehicles, aids, as well as sheltering facilities for the patients have to be organized.

In recent years, a multitude of papers have been published that deal with various aspects of evacuation planning in different situations (cf., for example, Hamacher and Tjandra (2002) and Bretschneider (2013)). These evacuation problems range from the evacuation of a single building to the evacuation of an entire region. At the same time, the evacuation of a hospital or another healthcare facility has received only a limited amount of attention despite the additional challenges encountered when evacuating these facilities. In particular, the patients inside a healthcare facility may often not be able to help themselves, i.e. they require assistance in order to reach a safety zone.

Some papers have been published, however, that deal with different aspects of evacuation planning for healthcare facilities. Here, the majority of these papers deal with the problem of optimizing the transportation of patients from staging areas outside the healthcare facilities to sheltering facilities. In contrast to this, the problem of optimizing the evacuation of the patients from their sick rooms inside the hospital to these staging areas (or, alternatively, to safety zones) has been considered in only a few papers. For example, Wolf (2001) and Golmohammadi and Shimshak (2011) have introduced different approaches to estimate the time required to evacuate patients from their sick rooms inside the hospital to safety zones.

In this thesis, we deal with the problem of modeling and optimizing the evacuation of patients from their sick rooms to staging areas (or safety zones) inside or outside the hospital. Here, as Sternberg et al. (2004) and Lipp et al. (1998) remark, many hospitals seldom or never stage evacuation exercises such that no figures may be available regarding the time required to perform a partial or even a complete evacuation of a hospital. For this reason, the solution approaches for the problem of hospital evacuations considered in this thesis may offer results that can then be used, for example, to estimate the time required to evacuate patients from the hospital as well as to better plan the further evacuation of the patients to sheltering facilities.

In the following, we assume that only a limited amount of assistants as well as a limited amount of aids are available for the evacuation of the patients. Furthermore, the infrastructure of the hospital is taken into account in order to estimate the time required to evacuate the patients by the available assistants and aids. For example, only a limited number of patients can be evacuated through the same corridors or in the same elevator at any time. While additional requirements or conditions may apply for different healthcare facilities (e.g. based on public building emergency regulations in different regions), these base requirements remain the same.

We model the problem of hospital evacuations considered in this thesis as a multi-mode resource-constrained project scheduling problem (multi-mode RCPSP or MRCPSP). For this, the evacuation of a single patient is regarded as a job that requires certain amounts of scarce resources (i.e. assistants, aids, as well as a sufficient amount of space along the evacuation route). These jobs can be processed in one out of several modes. For example, the same patient may either be evacuated by one assistant and one wheelchair or by two assistants and one stretcher. Also, different evacuation routes might be available for the evacuation of a patient. Finally, resource transfers as well as blockings are integrated into the model. Here, transfer times have to be taken into account whenever an assistant or an aid has to be transferred to the initial location of a patient (e.g. the sick room of the patient) before the patient can actually be evacuated. Furthermore, the transfer of an aid between two locations within the hospital has to be supported by a certain number of assistants. On the other hand, blockings can be regarded as delays during the evacuation of the patients. These delays occur, for example, if multiple patients have to wait in front of an elevator.

To the best of our knowledge, both of these extensions have received only limited attention in the context of the RCPSP. In particular, while Krüger

(2009) as well as Krüger and Scholl (2010) have presented a model for the RCPSP with resource transfers where some resources are required to support the transfer of other resources, no practical approaches to solve this problem have been introduced. Similarly, blocking constraints in the context of the RCPSP have only been considered in some works dealing with practical applications, e.g. in the master's thesis by Kröger (2013) in which blockings occur in a rail-rail transshipment problem. For this reason, we will especially consider the integration of these two extensions into the RCPSP.

In order to solve the problem of hospital evacuations considered in this thesis, we present two distinct solution approaches: one solution approach based on priority rules as well as one solution approach based on resource flows. For the latter approach, we first focus on the resource-constrained project scheduling problem with resource transfers and develop a solution approach based on resource flows for this problem. To do this, we describe a solution representation based on resource flows that incorporates resource transfers of resources that require the support of other resources and define modifications based on these resource flows. Furthermore, we report theoretical results related to both, the solution representation as well as the neighborhoods based on these modifications. Finally, in order to develop a solution approach for the problem of hospital evacuations based on resource flows, we extend this solution representation as well as the neighborhoods for the RCPSP with resource transfers to incorporate blockings.

The remainder of this thesis is divided as follows. First of all, in Chapter 2, we describe the problem of hospital evacuations in more detail. In particular, we give a short overview of literature concerned with typical solution approaches used in general evacuation problems (e.g. for the evacuation of a building). Furthermore, we survey literature related to the problem of evacuation planning for healthcare facilities. In this context, we then describe the problem tackled in this thesis and discuss why we have opted to model this problem as a multi-mode RCPSP instead of using a more typical approach used in evacuation problems.

Afterward, we give a formal definition of the classical resource-constrained project scheduling problem in Chapter 3 and describe different extensions of the problem. In particular, we introduce the multi-mode variant as well as the RCPSP with setup times. Also, in this chapter, different exact as well as heuristic solution approaches for the classical RCPSP are discussed. Next, in Chapter 4, we model the problem of hospital evacuations as a multi-mode resource-constrained project scheduling problem with resource transfers and blockings and present a mixed-integer linear programming formulation of the problem. Furthermore, a first solution approach for the problem is introduced. In this solution approach, a tabu search algorithm modifies the modes of the jobs while actual schedules are generated by schedule generation schemes based on priority rules.

Due to some shortcomings of this first solution approach related to resource transfers, we consider the resource-constrained project scheduling problem with resource transfers in more detail in Chapter 5. In particular, we deal with the problem of resource transfers that require other resources in order to support the transfer. Apart from introducing a model as well as a mixed-integer linear programming formulation for this problem, we discuss differences between our model as well as the model presented by Krüger (2009) and Krüger and Scholl (2010). Then, in Chapter 6, we present a solution approach based on resource flows for this problem. For this, we describe a resource flow representation that incorporates resource transfers of resources that require the support of other resources. Additionally, we define modifications based on these resource flows and report theoretical results related to both, the solution representation as well as the neighborhoods. At the end of this chapter, we introduce a tabu search algorithm for this problem in which solutions are represented as resource flows.

Based on the results from the previous chapter, we describe a second solution approach for the problem of hospital evacuations based on resource flows in Chapter 7. For this, we extend the resource flow representation as well as the neighborhoods to incorporate blockings. The problem is then solved by a two-stage solution approach such that solutions for the first stage represent the order in which the patients are evacuated by the available assistants and aids while solutions for the second stage represent complete solutions including the actual evacuation of the patients through the hospital.

In Chapter 8, we report computational results for the different solution approaches presented in this thesis. This includes results for the tabu search algorithm that has been introduced for the RCPSP with resource transfers as well as the two solution approaches for the problem of hospital evacuations. As far as such results are available, we have compared the results obtained by our algorithms to those reported in literature. For the problem of hospital evacuations, we compare the results obtained by both solution approaches with each other. Finally, in Chapter 9, we close this thesis with some concluding remarks.

2 Hospital Evacuations

Evacuation planning as a scientific research topic has received an increasing amount of attention since one of the first papers by Chalmet et al. (1982) has been published in the early 1980s dealing with the evacuation of a building. While a lot of research focuses on the evacuation of buildings, cities, or entire regions, however, the evacuation of healthcare facilities such as hospitals has received significantly less attention despite the additional difficulties encountered when evacuating these facilities. Most importantly, the patients within healthcare facilities often can not help themselves in order to reach a safety zone but rely on the help of other people.

In the following, a general overview of literature dealing with evacuation planning is given in Section 2.1. The most common evacuation planning approaches are based on either network flow problems or simulation and differ in how they model the problem. While network flow problems are often macroscopic models that are used to optimize a given objective but neglect the individual behavior of the affected people, simulation approaches are often microscopic models that focus on human behavior (e.g. panic).

Afterward, in Section 2.2, the problem of hospital evacuations is described, which is abbreviated as HEP (hospital evacuation problem) in the remainder of this thesis. Here, apart from defining the problem tackled in this thesis, this section also gives an overview of existing literature dealing with the problem of evacuation planning for healthcare facilities.

2.1 Evacuation Planning

The field of evacuation planning deals with a broad range of situations where evacuations might be necessary, ranging from building evacuations (e.g. due to fire) to evacuations of complete cities and regions (e.g. due to natural disasters). Generally, evacuations can be classified as either precautionary or life-saving evacuations (cf. Hamacher and Tjandra (2002)). On the one hand, for precautionary evacuations, the hazard is known in advance and the evacuation can be planned in detail. In particular, it is also possible to estimate the evacuation time as well as the risk to the people in the affected area. On the other hand, life-saving evacuations are usually necessary if hazards occur with insufficient warning and the affected people have to be evacuated without pre-emergency planning. In this case, the problems that arise are more direct (e.g. the rescue of injured people, clearing routes for the evacuation, fire-fighting, etc.) and often have to be dealt with in real time.

An important part of evacuation planning is to estimate the time required to evacuate all people from an area (e.g. a building, city, or region) while ensuring the safety of the people at the same time. Here, the time affected people require in order to move to safe regions is referred to as egress time and can be regarded as a lower bound on the actual evacuation time, which also consists of behavioral and organizational factors such as recognizing a danger and then deciding on a course of action (cf. Hamacher and Tjandra (2002)). Depending on the type of the hazard as well as the area that has to be evacuated, different aspects have to be taken into consideration. In the case of a chemical accident, for example, it might be more safe for the affected people to be sheltered in-place (e.g. inside homes, schools, etc.) instead of being evacuated (cf. Müller (1998)).

In the following, an overview of different approaches to evacuation planning is given. Here, Section 2.1.1 deals with (dynamic) network flow models that have been used for evacuation planning while Section 2.1.2 deals with other approaches, in particular those based on simulation.

2.1.1 Network Flow Models

Some of the earliest approaches for evacuation planning are based on network flow models where the evacuation of people is modeled by flows from source nodes (i.e. nodes where the people are initially located) to sink nodes (i.e. nodes representing safe regions) in a network. In these models, people are generally treated as homogeneous groups without individual behavior.

As described by Hoppe and Tardos (1994), dynamic network flow models are often used for evacuation planning. For these models, a network $G = (N, A)$ is given which consists of a set N of nodes as well as a set A of directed arcs between these nodes. Here, the set of nodes contains source nodes (i.e. nodes with a supply > 0), sink nodes (i.e. nodes with a demand < 0), as well as transshipment nodes (i.e. nodes with neither supply nor demand). Furthermore, with each directed arc $(i, j) \in A$ between two nodes $i \in N$ and $j \in N$, a transit time t_{ij} denoting the time required to move from node i to node j as well as a capacity c_{ij} denoting the amount of flow that can traverse the arc at any time t are associated. It should be noted that these parameters are not necessarily constant for all times t. Instead, transit times or capacity constraints might change over time, or some nodes might not be available at all times (e.g. due to smoke or fire). A feasible flow for this problem satisfies the capacity constraints (i.e. the flow from node $i \in N$ $i \in N$ to node $j \in N$ may not exceed the capacity c_{ij} of the arc $(i, j) \in A$ at any time t) as well as the transit times (i.e. a flow from node $i \in N$ to node $j \in N$ that leaves node i at time t arrives at node j at time $t + t_{ij}$).

One of the earliest dynamic network flow problems has been introduced by Ford and Fulkerson (1958). Here, the network consists of exactly one source and one sink node, as well as an arbitrary number of transshipment nodes. Based on such a network, Ford and Fulkerson (1958) compute the maximum flow from the source to the sink node in a specified time horizon T. This problem, referred to as the maximum dynamic flow problem, has been extended by Gale (1959) by additionally requiring the cumulative amount of flow to reach the sink node in each time period to be maximal. Gale (1959) refers to this extended problem as the universal maximum flow problem or the earliest arrival flow problem. In the context of evacuation planning, these problems can be used to estimate the maximum number of people that can be evacuated from a danger zone (source node) to a safety zone (sink node) within a given time horizon T if the actual number of affected people is not known.

A related problem to the maximum dynamic flow problem is the quickest flow problem that has been introduced by Burkard et al. (1993). In this problem, a specific supply v is assigned to the source node. The problem then consists of calculating the shortest time in which the amount v can flow from the source node to the sink node. An extension of the quickest flow problem with multiple sources and multiple sinks is, for example, discussed by Hoppe and Tardos (1994, 1995) who also present a polynomial time algorithm for this problem. Hoppe and Tardos (1994) also refer to the quickest flow problem with multiple sources and a single sink as the evacuation problem that can be used to model the evacuation of a building. Here, people are located in different rooms (i.e. multiple sources) and have to be evacuated from the building (i.e. the outside is modeled as a single sink).

As already noted above, some attributes of the network might change over time. For example, transit times between two adjacent nodes, arc capacities, or node capacities might change as time progresses, e.g. due to spreading smoke or fire. Here, such time-dependent attributes in the context of evacuation planning are discussed by Tjandra (2003) for the maximum dynamic flow problem, the universal maximum flow problem, as well as the quickest flow problem. Another variant of the problem are flow-dependent transit times. For example, Köhler et al. (2002) consider the problem of flow-dependent transit times where the transit time on an arc $(i, j) \in A$ depends on the flow on the arc at a specific time point t.

Dynamic network flow models have, for example, been applied to evacuation planning for buildings by Chalmet et al. (1982) as well as Choi et al. (1988). Here, Chalmet et al. (1982) use a dynamic network flow model as described above in order to minimize the average time required for a person to leave the building while Choi et al. (1988) extend this problem by flow-dependent arc capacities, i.e. the capacity of an arc depends on the number of people on that arc at a given time point t. In a more recent approach, Chen and Miller-Hooks (2008) incorporate shared information into the building evacuation problem, i.e. information and instructions that are provided to the people at specific locations and at specific times during the evacuation.

Surveys dealing with various dynamic network flow problems have, for example, been published by Hamacher and Tjandra (2002) as well as Kotnyek (2003). Additionally, the PhD thesis by Bretschneider (2013) offers a more recent overview of flow-based optimization models for evacuation planning.

2.1.2 Other Approaches

While the dynamic network flow models described above are macroscopic models treating people as homogenous groups and thus neglecting individual behavior, other approaches to evacuation planning based on simulation are often microscopic models that model each affected person as a separate flow object. In these simulations, people make individual choices regarding the route they will take in each node (decision point) until they reach their destination (safety zone) as displayed in Figure 2.1.

For example, Løvås (1998) uses different probabilistic rules in order to model the wayfinding behavior of people, i.e. at decision points (nodes) they have to choose between different routes to reach their destination. These rules

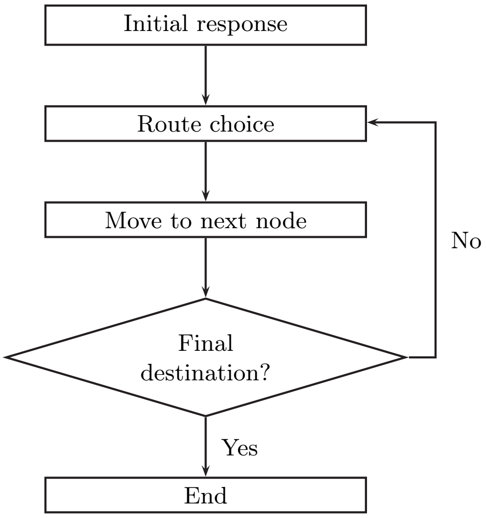


Figure 2.1: The evacuation process for one person (cf. Løvås (1998)). Starting with an initial response (e.g. an initially selected route), people make individual choices regarding the route they follow in each decision point (e.g. in each room) until they reach their destination (e.g. the safety zone). These choices can, for example, depend on a perceived danger.

have then been tested in a simulation framework and include, amongst others, a simple random walk rule as well as rules where an underlying optimal evacuation route is given (e.g. based on shortest paths) and people might stray from this route, for example if they are more familiar with other routes. More recently, Pan et al. (2007) introduced a multi-agent simulation framework for emergency evacuations where, on a microscopic level, people are modeled as agents that make individual decisions while, at a macroscopic level, the framework also takes into account interactions of individual agents as groups.

Another simulation approach is based on cellular automatons. For example, Kirchner and Schadschneider (2002) use a cellular automaton model for pedestrian dynamics in order to simulate the evacuation of a large room. Here, they estimate the evacuation times for different parameters relating to the behavior of the affected people, e.g. in case of panic. Similarly, Zheng et al. (2010) describe a simulation based on a cellular automaton that is used to evaluate the effect a partitioning wall has on jamming (i.e. the blocking of an exit) during the evacuation of a square.

Apart from these approaches, many other models have been introduced for the simulation of evacuation processes, including lattice gas models, social force models, and fluid-dynamic models. A recent overview of these as well as other models is, for example, given by Zheng et al. (2009).

2.2 The Problem of Hospital Evacuations

While the previous section gave a general overview of evacuation planning with a focus on various macroscopic and microscopic modeling approaches that are often used for evacuation planning of buildings or areas, this section deals with the problem of hospital evacuations. Here, an overview of existing literature on the topic of hospital evacuations, which has received an increasing amount of attention in the last few years, is given in Section 2.2.1. Afterward, the problem of hospital evacuations tackled in this thesis is described in Section 2.2.2. This includes a definition of the scenario and the resulting evacuation problem, as well as a description of the various details that are included in the model proposed for this problem.

2.2.1 Literature Review

As stated above, this section deals with evacuation planning for healthcare facilities such as hospitals or nursing homes. A recent survey regarding this issue has been published by Childers and Taaffe (2010). In this survey, they state that the problem of evacuation planning for healthcare facilities has received significantly less attention than other evacuation scenarios and highlight various research opportunities to help improve this situation. These include, for example, estimating the time required for the evacuation, as well as planning the transport of patients both inside as well as outside the healthcare facility.

First of all, as Childers and Taaffe (2010) point out, it is important to note that the planning of an evacuation is strongly influenced by the nature of the emergency. On the one hand, some threats such as floods are known in advance and allow the affected facilities to prepare (i.e. plan) their emergency response (e.g. an evacuation of the patients). Depending on the available time window, detailed plans regarding the transport of the patients as well as the required resources can be made. On the other hand, threats such as fires occur without prior notice and do not allow for additional evacuation planning. In this case, the response of the affected facility primarily depends on the preparedness of its staff, e.g. based on an evacuation plan that details how such emergencies should be handled.

An evaluation of 257 reported hospital evacuations (these include full as well as partial evacuations) in the United States between 1971 and 1999 by Sternberg et al. (2004) revealed that the majority of these evacuations were caused by fire, while natural disasters caused the most severe problems due to the large area affected by these disasters. In particular, Sternberg et al. (2004) note that the preparedness of hospitals is often insufficient due to a lack of exercises as well as a disregard of evacuation plans in the case of an actual emergency. A similar study of 522 hospitals in Germany by Lipp et al. (1998) showed that only 83.5% of these hospitals had any emergency plan and 51.2% of these hospitals have never staged an emergency exercise. Compared to this, 14.5% of these hospitals already had to activate their emergency plans.

The majority of papers dealing with evacuation planning for healthcare facilities focus on decision making procedures as well as emergency preparedness training. For example, an emergency management system that can be used for decision making in the case of a hospital evacuation is the Hospital Incident Command System (HICS) introduced by the California Emergency Medical Services Authority. Similarly, papers dealing with the decision making process with a focus on the situation in Germany have been published by Gretenkort and Harke (2001) as well as Urban et al. (2006). Furthermore, a number of papers deal with experiences from past evacuations. For example, Gray and Hebert (2007) describe the evacuation of multiple hospitals in New Orleans due to the Hurricane Katrina in August 2005, while Katter et al. (2008) describe the evacuation of a hospital in Germany after a bomb from World War II was discovered in the vicinity of the hospital.

Finally, the amount of papers dealing with evacuation planning for healthcare facilities using optimization or simulation techniques as described in Section 2.1 is very limited. Here, first of all, Taaffe et al. (2005) give an overview of issues and modeling approaches regarding the problem of evacuation planning for healthcare facilities. They point out that the nature of the threat influences the risks to patients and staff, as well as the availability of resources. For example, an external disaster such as a flood might also bind some of the available transportation resources that are no longer available for the evacuation of the healthcare facility. Finally, Taaffe et al. (2005) describe project scheduling models, mathematical programming, and simulation as possible modeling approaches for subproblems in hospital evacuations.

A simulation approach used to estimate the time required to evacuate all patients from one hospital to sheltering facilities (e.g. other hospitals) is described by Taaffe et al. (2006). This model focuses on the transport of the patients from the affected hospital to sheltering facilities and incorporates a limited number of transporting vehicles as well as support staff for the transport. Additionally, only a limited amount of space in the staging area (i.e. the area where the patients enter the transporting vehicles) as well as a limited number of beds in the sheltering facilities might be available. In this approach, the transport of the patients within the hospital is modeled as a stochastic delay. An extension of this approach is described by Tayfur and Taaffe (2007) in which the influence of traffic on the transporting vehicles is added to the model. An integer linear programming formulation for this problem is described by Tayfur and Taaffe (2009). Similarly, Bish et al. (2011) introduce an integer linear program for the problem of transporting patients from a hospital to other receiving hospitals while minimizing the risk to the patients.

Next, a prediction model used to estimate the time required to evacuate all patients from one or multiple floors of a hospital is introduced by Golmohammadi and Shimshak (2011). Here, the time required to evacuate all patients from their initial locations within the hospital to a safe location inside or outside the hospital is calculated while staff members that are available to aid in the evacuation as well as elevators are regarded as bottleneck resources. A simulation approach for estimating the time required to evacuate patients during an internal danger (e.g. a fire) is presented by Wolf (2001). In this approach, the section of the hospital that has to be evacuated is modeled as a network and assistants as well as aids (e.g. stretchers, wheelchairs, etc.) are preassigned to evacuate specific patients. In this model, a particular emphasis is put on representing real-world parameters such as the floor plan of the building as well as evacuation times of the patients using various aids.

Finally, while not directly connected to hospital evacuations, Hanne et al. (2009) as well as Beaudry et al. (2010) tackle the problem of patient transportation within large hospitals. In this problem, requests to transport patients between different locations within a hospital arrive in real-time and have to be inserted into current vehicle routes, where a vehicle can, for example, be a wheelchair or a stretcher. This problem has been modeled as a dial-a-ride problem and solved using a heuristic algorithm by both, Hanne et al. (2009) as well as Beaudry et al. (2010).

2.2.2 Problem Description

In this section, the problem of hospital evacuations as it is tackled in this thesis is defined. Here, as already noted by Childers and Taaffe (2010), a threat that might lead to an evacuation can either be predictable or unpredictable. On the one hand, if a threat is predictable (i.e. if the threat is known in advance), it is possible to plan the evacuation in more or less detail. Floods, hurricanes (primarily in the US), or - as is frequently the case in Germany - the disposal of a bomb from World War II, are examples for predictable threats that might make it necessary to evacuate a hospital. Often in these cases, the entire hospital has to be evacuated and patients have to be transported to sheltering facilities, e.g. other hospitals. On the other hand, if a threat is unpredictable (i.e. if the threat occurs without prior notice), the evacuation can not be planned in advance. In this case, real-time decision making based on more general evacuation plans is used to evacuate the patients. Unpredictable threats include, for example, a fire or a hazardous materials spill within the hospital, or a bomb threat.

Furthermore, the situation inside the hospital can either be described as static or dynamic. Here, external (and often predictable) threats such as a flood or a bomb disposal generally do not change the conditions inside the hospital during the evacuation, i.e. they can be classified as static scenarios. For example, the capacity of the corridors does not change during the evacuation and it is less likely that a panic will occur during a planned evacuation. On the other hand, internal (and often unpredictable) threats such as a fire or a hazardous materials spill may change the conditions under which patients are evacuated from the hospital, i.e. they can be classified as dynamic scenarios. For example, whole corridors might become impassable due to a spreading fire or smoke and it is more likely that people will start to panic due to an approaching threat.

This thesis focuses on evacuation planning for hospitals under predictable and static conditions. In contrast to unpredictable and dynamic situations, these conditions make it possible to deterministically plan the evacuation because all required data can be collected prior to the evacuation (e.g. the number of patients that have to be evacuated, available staff and aids, etc.) and the general conditions under which patients have to be evacuated from the hospital remain the same during the evacuation. Also, in such a situation, it might be possible to close the hospital to visitors on the day of the evacuation (i.e. only the patients have to be evacuated) and some patients that are due to be discharged soon might be discharged early. In Germany, such a situation might occur if an unexploded bomb from World War II has to be disposed and the area surrounding the bomb has to be evacuated. In this case, if a hospital or another healthcare facility such as a nursing home is affected, the patients have to be evacuated from the facility and transported to sheltering facilities outside the affected area.

The evacuation of a patient in this scenario can be broken down into two subproblems: the transport of the patient from a location within the hospital to a staging area inside or outside the hospital, as well as the further transport of the patient from the staging area to the sheltering facility using transportation vehicles. Here, as described above, the assignment of patients to transportation vehicles for the evacuation of the patients from a staging area to sheltering facilities is, for example, tackled by Tayfur and Taaffe (2007, 2009). This thesis, on the other hand, focuses on the transport of the patients within the hospitals from their initial locations (e.g. their sickrooms) to one or multiple staging areas or safety zones.

Unlike the majority of other evacuation scenarios described above where people can move towards the safety zone on their own, the patients in this scenario rely on the help of assistants (e.g. hospital staff, firemen, etc.) as well as aids (e.g. stretchers, wheelchairs, hospital beds, etc.) for the transport. Additionally, depending on the severeness of the disease, the patients have to be evacuated using different resources. For example, some patients can walk on their own or with the help of one assistant, while other patients can only be transported in a wheelchair or on a stretcher. Finally, intensive care patients can maybe only be transported in hospital beds and require constant supervision by a doctor. In the model proposed for this problem, both, assistants as well as aids, are regarded as scarce resources that are required for the evacuation of the patients.

Apart from these two types of resources, the infrastructure of the hospital is regarded as a third type of resource. Here, the infrastructure is divided into building sections (e.g. sickrooms, corridors, stairs, elevators, etc.) with specific capacities corresponding to the space available for the transport of patients through these sections as well as possibly other characteristics. Then, starting from the initial location of a patient, the patient has to be evacuated through a series of building sections until he reaches a safety zone. Similar to the resources introduced above, multiple alternatives might be available in order to evacuate a patient from his initial location to a safety zone, for example if a patient can be transported to one of several safety zones, or if various routes are available to one safety zone. Additionally, the type of aids used for the evacuation of a patient might also influence the evacuation routes available for the evacuation of the patient.

Now, depending on the resources selected for the evacuation of a patient (e.g. a wheelchair and one assistant or a stretcher and two assistants) as well as on the type of building section the patient is evacuated through (e.g. a corridor or a stair), the patient can be transported with a specific speed (cf. Wolf (2001)). For example, if the patient is evacuated through a building section representing a corridor, the time required to evacuate the patient through this section depends on the speed of the aids and assistants used for the evacuation as well as on the length of the building section. Other types of building sections (e.g. stairs and elevators) might impose additional constraints on how quickly patients can be evacuated through them. Depending on the number of patients evacuated through the same building sections simultaneously, a congestion of these sections can occur. In this case, all patients evacuated through these building sections at the time of the congestion can only be evacuated with a reduced speed.

Finally, assistants and aids that have been used to evacuate one patient to a safety zone first have to be transferred to the location of the next patient they have to evacuate. Here, depending on the distance between the safety zone as well as the location of the next patient, a transfer time has to be taken into account. Additionally, aids have to be transported by assistants between different locations within the hospitals. For this, it may be necessary to first transfer an assistant to the location of the required aid and then, from there, to transport the aid to the location of the patient. Only after all required assistants and aids have arrived at the location of the patient can the evacuation of the patient begin.

Now, in order to plan the evacuation of one patient, first of all, the types of resources required for the evacuation of the patient as well as the evacuation route along which the patient will be evacuated have to be chosen. Then, specific assistants and aids have to be transferred to the location of the patient. The evacuation of the patient itself can only begin when all of these resources have arrived at the initial location of the patient. Here, the evacuation might consist of multiple steps: a preparation of the patient for the evacuation, the evacuation itself along the selected evacuation route, as well as a possible postprocessing of the patient in the staging area, e.g. to prepare the patient for the further transport to a sheltering facility. Each of these steps has to be scheduled at specific times such that all required assistants and aids are available for the length of the evacuation and sufficient space is available on the required sections along the evacuation route. After the last step has been finished, the required assistants and aids are available again to be used for the evacuation of other patients.

These steps described above have to be performed for each patient such that the time required to evacuate all patients from their initial locations to a safety zone is minimized. In order to solve this problem, it is first modeled as a resource-constrained project scheduling problem and then solved using heuristic algorithms. Unlike the more common approaches for evacuation planning discussed in Section 2.1, this approach is better suited for the problem of hospital evacuations tackled here for the following reasons.

First of all, network flow problems generally focus on the movement of groups of people inside a building towards safety zones in order to estimate the time required until the building is evacuated. Here, network flow problems can not be used to model the problem of hospital evacuations because the patients inside the hospital can often not move towards safety zones independently but rather depend on resources like assistants and aids to help them (i.e. each patient has to be evacuated independently). On the other hand, simulation approaches generally focus on the individual behavior of the affected people in the situation of an evacuation. Here, simulation approaches are generally feasible as a means to evaluate a given evacuation plan. They can not be used, however, in order to calculate and optimize a detailed plan for the evacuation of the patients.

For these reasons, the resource-constrained project scheduling problem offers a good alternative because it incorporates activities (i.e. the evacuation of patients) that require multiple types of resources (i.e. assistants, aids, and building sections) which have to be scheduled so as to optimize a given objective function (e.g. to minimize the time required to evacuate all patients). In the following, a general introduction of the resource-constrained project scheduling problem is given in Chapter 3 while a first solution approach based on the resource-constrained project scheduling problem for the problem of hospital evacuations is introduced in Chapter 4.

3 Resource-Constrained Project Scheduling

The resource-constrained project scheduling problem (RCPSP) has been extensively studied in scientific literature since the early 1960s with some of the earliest works contributed, for example, by Wiest (1963, 1964) as well as Kelley (1963). Since this time, a large number of publications concerning the resource-constrained project scheduling problem and its extensions have been published, including various surveys (e.g. by Herroelen et al. (1998), Brucker et al. (1999), Kolisch and Padman (2001), Hartmann and Briskorn (2010), etc.) and books (e.g. by Demeulemeester and Herroelen (2002), Brucker and Knust (2006), Artigues et al. (2010), etc.).

In the following, the resource-constrained project scheduling problem is introduced in Section 3.1. Apart from a definition of the classical problem, this section includes a short overview of the activity-on-node network representation, critical paths, as well as mixed-integer linear programming (MILP) formulations of the problem. Furthermore, the section deals with extensions of the classical problem, of which the multi-mode RCPSP as well as the RCPSP with setup times are of particular interest for this thesis.

Afterward, Section 3.2 deals with solution approaches for the resource-constrained project scheduling problem. Here, after discussing the time complexity of the RCPSP and introducing a classification scheme for schedules, a short overview of various solution approaches with a focus on heuristic algorithms for solving the RCPSP is given.

3.1 The Resource-Constrained Project Scheduling Problem

In the classical problem formulation, as described, for example, by Brucker and Knust (2006) and Artigues et al. (2010), n activities $i = 1, \dots, n$ have to be scheduled under precedence and resource constraints such that a given objective function is optimized. For this, r renewable resources $k = 1, \dots, r$ with limited resource capacities R_k are given. Then, at any time t (or in any time interval $[t, t+1[$, respectively), an amount of exactly R_k units of each resource k is available. In the following, the set $V = \{1, \ldots, n\}$ contains all activities while the set $\mathcal{R} = \{1, \ldots, r\}$ contains all resources.

Now, each activity $i \in V$ requires a certain amount of time (denoted by the processing time p_i of the activity) as well as resources (denoted by resource requirements r_{ik} for resources $k \in \mathcal{R}$) in order to be processed. Here, all resources required by an activity have to be available simultaneously for the complete duration of processing the activity. Additionally, precedence constraints $i \rightarrow j$ with $i \neq j$ may be given between pairs of activities $i \in V$ and $j \in V$ such that activity j can only start after activity i has been completed.

A schedule for an instance of the resource-constrained project scheduling problem is then defined by the starting times S_i of the activities $i \in V$ (for this problem without preemption, the corresponding completion times are given by $C_i = S_i + p_i$) such that the precedence constraints $i \rightarrow j$ are observed and, at any time t, the amount of each resource $k \in \mathcal{R}$ used to process all activities $i \in V$ with $S_i \leq t < S_i + p_i$ is not larger than the available amount R_k of this resource. If both of these conditions are fulfilled, the schedule S is referred to as precedence- and resource-feasible or simply as a feasible schedule. An optimal schedule S^* for a given problem instance is a feasible schedule that optimizes the given objective function. For example, for the makespan objective function C_{max} with

$$C_{max} = \max_{i=1}^{n}\{S_i + p_i\}$$

that denotes the time required to complete all activities $i = 1, \ldots, n$ in a given schedule, the following condition holds for all feasible schedules S:

$$C_{max}(S^*) \leq C_{max}(S)$$

In the following, a dummy start activity 0 as well as a dummy end activity $n + 1$ with processing times $p_0 = p_{n+1} = 0$ and resource requirements $r_{0k} = r_{n+1,k} = 0$ for all resources $k = 1, \ldots, r$ are added to each project. These activities represent the first and the last activity processed in any feasible schedule, i.e. the starting time S_0 of activity 0 is the time t_0 (often, $t_0 = 0$ is used) at which the schedule starts while the completion time C_{n+1} (or, alternatively, S_{n+1}) of activity $n + 1$ is the time at which all activities have been completed, i.e. it represents the makespan C_{max}. For this, additional precedence constraints $0 \rightarrow j$ are introduced for all activities $j \in V$ that do not have a predecessor activity and additional precedence constraints $i \to n+1$ are introduced for all activities $i \in V$ that do not have a successor activity. Now, the resulting set $V_{\text{all}} = \{0, 1, \ldots, n, n+1\}$ contains all real activities $i = 1, \ldots, n$ as well as the two dummy activities 0 and $n+1$ while the set A contains all precedence constraints $i \to j$ with $A = \{(i,j) \,|\, i, j \in V_{\text{all}}, i \to j\}$.

3.1.1 Activity-On-Node Network

The structure of a given instance of the resource-constrained project scheduling problem is often represented using an activity-on-node (AON) network or precedence diagram as it has been introduced by Fondahl (1961). In an activity-on-node network, the activities $i \in V_{\text{all}}$ are represented as nodes while the precedence constraints $(i,j) \in A$ are represented as directed arcs between these nodes.

Example 3.1 We consider a small project consisting of $n = 6$ real activities $i = 1, \ldots, 6$ as well as the two dummy activities 0 and 7. Between these activities, the following precedence constraints are given: $0 \to 1$, $0 \to 2$, $1 \to 3$, $1 \to 4$, $2 \to 5$, $4 \to 6$, $5 \to 6$, $3 \to 7$, and $6 \to 7$. Furthermore, $r = 2$ renewable resources with capacities $R_1 = 3$ and $R_2 = 2$ are available. The processing times p_i and resource requirements r_{ik} of the activities are given in Table 3.1.

i	0	1	2	3	4	5	6	7
p_i	0	2	1	1	2	2	3	0
r_{i1}	0	1	2	0	3	1	2	0
r_{i2}	0	0	1	2	0	0	1	0

Table 3.1: Processing times p_i and resource requirements r_{ik} of the $n = 6$ real activities $i = 1, \ldots, 6$ as well as the two dummy activities 0 and 7 for the project from Example 3.1.

An activity-on-node network for this project is displayed in Figure 3.1. In this network, the processing times p_i of the activities are given above the respective nodes. While an activity-on-node network for a project represents the logical order in which the activities have to be processed with respect to the given precedence constraints in any feasible schedule, an actual schedule for a project is often displayed using a Gantt-chart (Gantt, 1903, 1910). Here, a precedence- and resource-feasible schedule for this project is shown

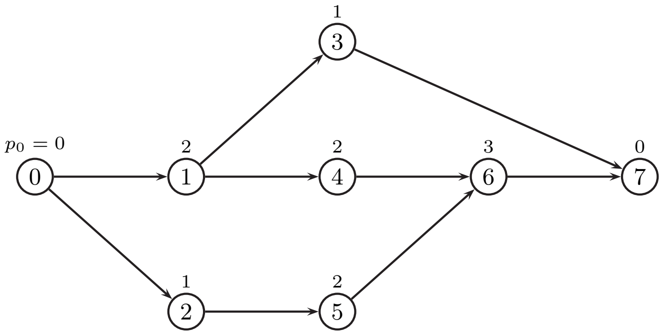


Figure 3.1: Representation of the $n = 6$ real activities $i = 1, \ldots, 6$ as well as the two dummy activities 0 and 7 in an activity-on-node network. The directed arcs between the nodes represent the precedence constraints $(i, j) \in A$ between the activities.

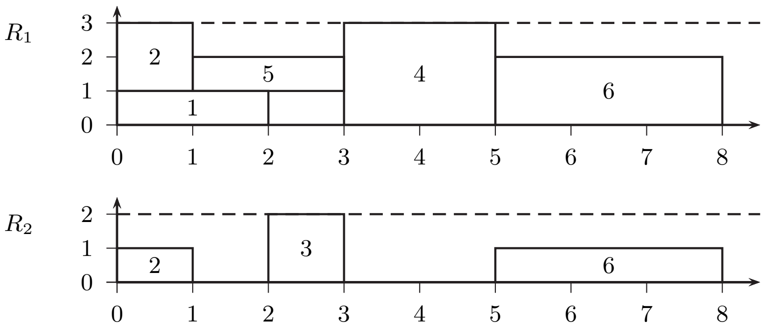


Figure 3.2: Representation of a feasible schedule for the project using a Gantt-chart. Here, the first chart displays the usage of resource $k = 1$ over the time $t = 1, \ldots, 8$ while the second chart displays the usage of resource $k = 2$.

in Figure 3.2. The makespan of this schedule is given by the completion time of activity 6 (the last real activity to be completed), which coincides with the earliest possible starting time of dummy end activity 7 for this schedule, i.e. the makespan is $C_{max} = 8$. □

3.1.2 Critical Path

Based on the network structure of a project, some important values including the earliest start time ES_i of an activity $i \in V_{\text{all}}$, the earliest finish time EF_i, the latest start time LS_i, as well as the latest finish time LF_i can be calculated. These values are described by Kelley and Walker (1959) and Malcolm et al. (1959) and are used in two early methods for project scheduling, the critical path method (CPM) as well as the program evaluation and review technique (PERT). It should be noted that these four values are calculated without considering the given resource constraints.

The values ES_i, EF_i, LS_i and LF_i can be calculated using a forward and a backward pass over all activities $i \in V_{\text{all}}$ as described, for example, by Demeulemeester and Herroelen (2002). Here, for the forward pass, the earliest start time ES_0 as well as the earliest finish time EF_0 of the dummy start activity 0 are initialized with $ES_0 = EF_0 = 0$. Then, the remaining activities $j = 1, \ldots, n+1$ are sorted in a topological order (i.e. all predecessor activities i of an activity j have to be in front of activity j in the topological order) and the values ES_j and EF_j can be calculated for these activities according to this order as follows:

$$
\begin{aligned}
ES_j &= \max\{EF_i \,|\, (i,j) \in A\} \\
EF_j &= ES_j + p_j
\end{aligned}
$$

Similarly, the latest start and finish times can be calculated in a backward pass. For this, the latest start time LS_{n+1} as well as the latest finish time LF_{n+1} of the dummy end activity $n+1$ are set to an upper bound UB on the latest allowed completion time of the project, i.e. $LS_{n+1} = LF_{n+1} = UB$. Then, the remaining activities $i = 0, \ldots, n$ are sorted in a reverse topological order (i.e. all successor activities j of an activity i have to be in front of activity i in the reverse topological order) and the values LS_i and LF_i can be calculated for these activities according to this order as follows:

$$
\begin{aligned}
LF_i &= \min\{LS_j \,|\, (i,j) \in A\} \\
LS_i &= LF_i - p_i
\end{aligned}
$$

Now, in any feasible schedule, an activity $i \in V$ always has to be processed between its earliest start time ES_i and its latest finish time LF_i. The total slack TS_i of an activity i is then defined as the time $TS_i = LS_i - ES_i = LF_i - EF_i$ and defines the time by which an activity can be moved in any feasible schedule with the given upper bound UB (Demeulemeester and

Herroelen, 2002). By setting the upper bound UB to the earliest finish time of the dummy end activity $n+1$, i.e. by setting $LF_{n+1} = EF_{n+1}$, some activities $i \in V$ of the project might have a total slack of $TS_i = 0$, i.e. they can not be moved in a feasible schedule. These activities are referred to as critical activities. A path of critical activities between the dummy start activity 0 and the dummy end activity $n+1$ is referred to as a critical path of the project and corresponds to a longest path from dummy start activity 0 to dummy end activity $n+1$ with respect to the sum of the processing times of all activities on the path.

Example 3.2 For the small project considered in Example 3.1, the values ES_i, EF_i, LS_i, LF_i, and TS_i described above have been calculated for all activities $i = 0, \ldots, 7$ as displayed in Table 3.2. The upper bound used for the calculation of the latest start and finish times has been set to $UB = 7$.

i	0	1	2	3	4	5	6	7
ES_i	0	0	0	2	2	1	4	7
EF_i	0	2	1	3	4	3	7	7
LF_i	0	2	2	7	4	4	7	7
LS_i	0	0	1	6	2	2	4	7
TS_i	0	0	1	4	0	1	0	0

Table 3.2: The earliest start time ES_i, the earliest finish time EF_i, the latest start time LS_i, the latest finish time LF_i, as well as the total slack TS_i of the real activities $i = 1, \ldots, 6$ as well as the two dummy activities 0 and 7 for the project from Example 3.1. The values have been calculated based on an upper bound $UB = 7$.

In Figure 3.3, the activity-on-node network for this project has been extended by time windows $[ES_i, LF_i]$ for all activities $i \in V_{\text{all}}$. For each activity $i \in V_{\text{all}}$, the value ES_i denotes the earliest time at which the activity can start and LF_i denotes the latest time at which the activity can finish, i.e. the activity has to be processed between these two times in order to ensure that no more than $UB = 7$ time units are required to process all activities. Also, the critical path $0 \rightarrow 1 \rightarrow 4 \rightarrow 6 \rightarrow 7$ of the project is highlighted in this figure. □

The length of a critical path of a project (i.e. the sum of the processing times of the activities on the critical path) can also be regarded as a simple lower bound LB_0 on the optimal makespan if all resource constraints are

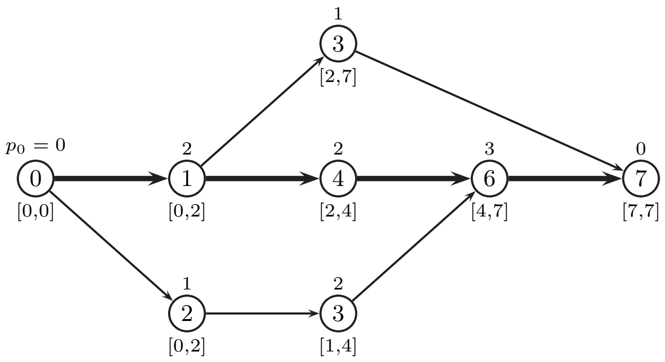


Figure 3.3: Representation of the activity-on-node network for the project from Example 3.1 including the time windows $[ES_i, LF_i]$ for all activities $i \in V_{\text{all}}$ based on an upper bound $UB = 7$. The unique critical path of the project is highlighted in the network.

neglected and only the precedence constraints between the activities are considered (cf. Brucker and Knust (2006)). Thus, for a given project, no feasible schedule S with a makespan $C_{max}(S) < LB_0$ can exist.

3.1.3 Mixed-Integer Linear Programming Formulations

As stated by Davis (1973), one of the earliest integer linear programming formulations for the resource-constrained project scheduling problem has been introduced by Wiest (1963) as an adaption of an integer linear programming formulation for the job-shop problem by Bowman (1959). Another formulation more widely used in scientific literature, however, has been introduced by Pritsker et al. (1969) and uses time-indexed binary variables. Below, this time-indexed formulation as well as a flow-based formulation that has been introduced by Artigues et al. (2003) are described.

Time-Indexed Formulation

First of all, for the time-indexed mixed-integer linear programming formulation described by Pritsker et al. (1969), a time horizon T is required that denotes an upper bound on the available time to complete all activities. Then, time-indexed binary variables x_{it} are introduced for all activities $i \in V_{\text{all}}$ as well as all time points $t = 0, \ldots, T$ such that

$$x_{it} = \begin{cases} 1, \text{ if activity } i \text{ is completed at time } t \\ 0, \text{ otherwise.} \end{cases}$$

The classical resource-constrained project scheduling problem can then be modeled by the mixed-integer linear programming formulation (3.1) to (3.5) (cf. Brucker and Knust (2006)).

$$\min \quad \sum_{t=0}^{T} t x_{n+1,t} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{t=0}^{T} x_{it} = 1 \qquad (i \in V_{\text{all}}) \tag{3.2}$$

$$\sum_{t=0}^{T} t x_{it} - \sum_{t=0}^{T} (t - p_j) x_{jt} \le 0 \qquad ((i,j) \in A) \tag{3.3}$$

$$\sum_{i=1}^{n} r_{ik} \sum_{\tau=t+1}^{\min\{t+p_i,T\}} x_{i\tau} \le R_k \qquad (k \in \mathcal{R};\ t = 0, \ldots, T) \tag{3.4}$$

$$x_{it} \in \{0,1\} \qquad (i \in V_{\text{all}};\ t = 0, \ldots, T) \tag{3.5}$$

Here, the makespan as denoted by the completion time C_{n+1} of dummy end activity $n + 1$ is minimized in (3.1). In order to ensure that all activities are scheduled once (without preemption), constraints (3.2) are introduced. Next, constraints (3.3) ensure that all precedence constraints $(i, j) \in A$ between activities $i \in V_{\text{all}}$ and $j \in V_{\text{all}}$ are observed while constraints (3.4) ensure that, at any time t, no more than the available amount of R_k units of any resource $k \in \mathcal{R}$ are required to process the activities that are active at this time (i.e. all activities $i \in V_{\text{all}}$ with $S_i \le t < S_i + p_i$ that have been scheduled to be processed in the time interval $[t, t + 1[$). Together, constraints (3.3) and (3.4) ensure that any solution for this mixed-integer linear programming formulation is precedence- as well as resource-feasible.

Flow-Based Formulation

Now, an alternative mixed-integer linear programming formulation for the classical RCPSP based on resource flows is introduced as it has been described, for example, by Artigues et al. (2003). This representation is based on the transfer of resources between activities such that, for each real activity $i \in V$, the amount of incoming resource units as well as the amount of outgoing resources units of each resource $k \in \mathcal{R}$ has to be equal to the resource requirements r_{ik} of the activity. A resource flow that satisfies this condition is called resource-feasible. Additionally, any feasible resource flow has to be acyclic and observe the given precedence constraints $(i, j) \in A$ between the activities (i.e. it has to be precedence-feasible). It should be noted that the resource requirements r_{0k} and $r_{n+1,k}$ of the dummy activities 0 and $n + 1$ for all resources $k \in \mathcal{R}$ are set to $r_{0k} = r_{n+1,k} = R_k$ for this alternative representation, i.e. all resource units are initially located at dummy start activity 0 and have to be transferred to dummy end activity $n + 1$ at the end of the project.

Example 3.3 We consider a small project consisting of $n = 4$ activities with precedence constraints $1 \to 2$ and $3 \to 4$ between real activities as well as $r = 1$ renewable resource with a capacity of $R_1 = 6$. The processing times and resource requirements of the activities are given in Table 3.3.

i	0	1	2	3	4	5
p_i	0	2	4	1	3	0
r_{i1}	6	2	3	3	2	6

Table 3.3: Processing times p_i and resource requirements r_{i1} of the activities $i = 0, \ldots, 5$ for the project considered in Example 3.3.

For this project, a so-called AON-flow network consisting of a feasible resource flow as well as the activity-on-node network of the project is displayed in Figure 3.4. As described above, a feasible flow is defined as a flow that is acyclic and precedence-feasible, and for which the amount of incoming as well as outgoing resource units to and from an activity $i \in V$ is equal to the resource requirements r_{ik} of this activity for each resource $k \in \mathcal{R}$. It can be seen that all resource units are initially located at dummy activity 0 and are collected at dummy activity 5 at the end of the project. This also includes one unused resource unit that is transferred directly from activity 0 to activity 5.

Finally, a schedule corresponding to this AON-flow network is shown in Figure 3.5. It should be noted that this schedule is left-justified, i.e. all activities start as early as possible with respect to the resource flow as well

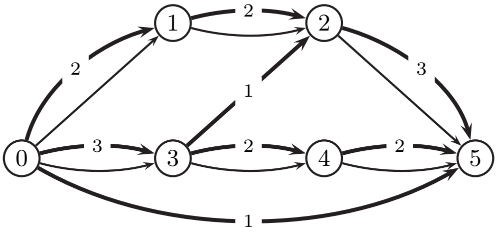


Figure 3.4: AON-flow network consisting of a feasible resource flow as well as the activity-on-node network for the project considered in Example 3.3. In this network, thick arcs represent resource transfers while thin arcs represent precedence constraints.

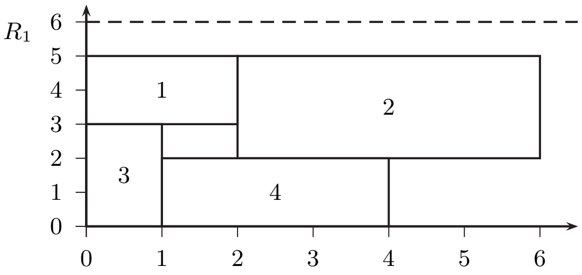


Figure 3.5: A left-justified schedule with the makespan $C_{max} = 6$ corresponding to the AON-flow network displayed in Figure 3.4.

as to the given precedence constraints. A left-justified schedule can be calculated based on the longest paths in the AON-flow network where each arc (i, j) is weighted with the processing time p_i of activity $i \in V_{\text{all}}$. The earliest starting time S_i of an activity $i \in V_{\text{all}}$ then corresponds to the length of a longest path from dummy start activity 0 to activity i. □

For this alternative mixed-integer linear programming formulation, two additional sets $V_0 = \{0, 1, \ldots, n\}$ and $V_* = \{1, \ldots, n, n+1\}$ are introduced. Additionally, the following variables are defined: Integer variables f_{ijk} for activities $i \in V_0$ and $j \in V_*$ as well as resources $k \in \mathcal{R}$ denoting the amount of resource k that is transferred from activity i to activity j, integer variables S_i for all activities $i \in V_{\text{all}}$ denoting the starting time of activity i, and binary variables x_{ij} for activities $i \in V_0$ and $j \in V_*$ with

$$x_{ij} = \begin{cases} 1, \text{ if activity } j \text{ is constrained to start after the end of activity } i \\ 0, \text{ otherwise.} \end{cases}$$

It should be noted that $x_{ij} = 1$ only has to hold if a precedence constraint $(i,j) \in A$ or a resource transfer $f_{ijk} > 0$ for some resource $k \in \mathcal{R}$ from activity $i \in V_0$ to activity $j \in V_*$ is given. Otherwise, i.e. if neither a precedence constraint nor a resource transfer is given, $x_{ij} = 0$ can also hold even if activity j only starts after the completion of activity i. On the other hand, if activity j starts before activity i has finished, $x_{ij} = 0$ always has to hold. Finally, M and N are two large integer values. Now, based on the paper by Artigues et al. (2003), the mixed-integer linear programming formulation (3.6) to (3.14) can be defined.

$$\begin{aligned}
\min \quad & S_{n+1} && && (3.6)\\
\text{s.t.} \quad & x_{ij} = 1 && ((i,j) \in A) && (3.7)\\
& S_j - (S_i + p_i) + M(1 - x_{ij}) \geq 0 && (i \in V_0;\, j \in V_*) && (3.8)\\
& f_{ijk} - N x_{ij} \leq 0 && (i \in V_0;\, j \in V_*;\, k \in \mathcal{R}) && (3.9)\\
& \sum_{i \in V_0} f_{ijk} = r_{jk} && (j \in V_*;\, k \in \mathcal{R}) && (3.10)\\
& \sum_{j \in V_*} f_{ijk} = r_{ik} && (i \in V_0;\, k \in \mathcal{R}) && (3.11)\\
& S_i \in \mathbb{N} && (i \in V_{\text{all}}) && (3.12)\\
& x_{ij} \in \{0,1\} && (i \in V_0;\, j \in V_*) && (3.13)\\
& f_{ijk} \in \mathbb{N} && (i \in V_0;\, j \in V_*;\, k \in \mathcal{R}) && (3.14)
\end{aligned}$$

Here, the makespan as denoted by the starting (and completion) time S_{n+1} of dummy end activity $n + 1$ is minimized in (3.6). Next, constraints (3.7) ensure that, if a precedence constraint $(i, j) \in A$ between two activities $i \in V_{\text{all}}$ and $j \in V_{\text{all}}$ is given, activity j can only start after activity i has been completed. The starting times of activities $j \in V_*$ are then calculated by constraints (3.8). These constraints ensure that activity $j \in V_*$ can only start at time $S_j \geq S_i + p_i$ after the completion of activity $i \in V_0$ if $x_{ij} = 1$ holds for activities i and j, i.e. if a precedence constraint $(i, j) \in A$ is given or if a resource transfer $f_{ijk} > 0$ for some resource $k \in \mathcal{R}$ exists.

Next, constraints (3.9) link the resource transfers f_{ijk} with the binary variables x_{ij} such that $x_{ij} = 1$ has to hold if $f_{ijk} > 0$ holds for activities $i \in V_0$

and $j \in V_*$ and some resource $k \in \mathcal{R}$. Finally, constraints (3.10) and (3.11) ensure that the amount of incoming resource units to an activity $j \in V_*$ as well as the amount of outgoing resource units from an activity $i \in V_0$ have to be equal to the corresponding resource requirements r_{jk} and r_{ik} for each resource $k \in \mathcal{R}$, respectively. It should be noted that here, activity 0 only has outgoing resource transfers while activity $n+1$ only has incoming resource transfers.

3.1.4 Multi-Mode RCPSP

An important generalization of the classical resource-constrained project scheduling problem is the multi-mode RCPSP or MRCPSP. The problem of multiple activity execution modes under resource constraints has first been considered in the 1970s, for example by Elmaghraby (1977). For this generalization, the RCPSP is extended by execution modes such that each real activity $i \in V$ can be executed in M_i different modes $m = 1, \ldots, M_i$. Then, for each mode m, the activity has mode-dependent processing times p_{im} as well as mode-dependent resource requirements r_{ikm} for each resource $k \in \mathcal{R}$. It should be noted that the dummy start activity 0 as well as the dummy end activity $n+1$ can only be executed in $M_0 = M_{n+1} = 1$ mode each. In the following, the set $\mathcal{M}(i) = \{1, \ldots, M_i\}$ contains all available modes for an activity $i \in V_{\text{all}}$.

Similar to the MILP formulation for the RCPSP by Pritsker et al. (1969), a time-indexed mixed-integer linear programming formulation for the MR-CPSP has been presented by Talbot (1982). In this formulation, binary variables x_{imt} are introduced for all activities $i \in V_{\text{all}}$, their respective modes $m \in \mathcal{M}(i)$, as well as all time points $t = 0, \ldots, T$ such that

$$x_{imt} = \begin{cases} 1, \text{ if activity } i \text{ is executed in mode } m \text{ and completed at time } t \\ 0, \text{ otherwise.} \end{cases}$$

Now, the multi-mode RCPSP can be modeled as the mixed-integer linear program (3.15) to (3.19) (cf. Brucker and Knust (2006)).

$$\min \quad \sum_{t=0}^{T} t x_{n+1,1,t} \tag{3.15}$$

$$\text{s.t.} \quad \sum_{m=1}^{m_i} \sum_{t=0}^{T} x_{imt} = 1 \qquad (i \in V_{\text{all}}) \tag{3.16}$$

$$\sum_{m=1}^{m_i}\sum_{t=0}^{T} t x_{imt} - \sum_{m=1}^{m_j}\sum_{t=0}^{T}(t-p_j)x_{jmt} \leq 0 \qquad ((i,j) \in A) \tag{3.17}$$

$$\sum_{i=1}^{n}\sum_{m=1}^{m_i} r_{imk} \sum_{\tau=t+1}^{\min\{t+p_i,T\}} x_{im\tau} \leq R_k \qquad \begin{pmatrix} k \in \mathcal{R}; \\ t = 0,\ldots,T \end{pmatrix} \tag{3.18}$$

$$x_{imt} \in \{0,1\} \qquad \begin{pmatrix} i \in V_{\text{all}}; \\ m \in \mathcal{M}(i); \\ t = 0,\ldots,T \end{pmatrix} \tag{3.19}$$

As before, the makespan (3.15) as denoted by the completion time C_{n+1} of dummy end activity $n+1$ is minimized. Next, equations (3.16) ensure that each activity $i \in V$ is scheduled once (without preemption) and executed in a mode $m \in \{1,\ldots,M_i\}$ while inequalities (3.17) and (3.18) ensure that all precedence and resource constraints are adhered to.

A further generalization of the MRCPSP is the mode identity resource-constrained project scheduling problem introduced by Salewski et al. (1997). In this extension, the modes of the activities can not be selected separately. Instead some activities might have to be executed in the same mode, for example if these activities have to use the same resources.

3.1.5 Setup Times

Another extension of the classical resource-constrained project scheduling problem are setup times. Setup times generally occur before an activity can be processed by the required resources and model the time needed to prepare these resources for the execution of the activity. Here, as pointed out by Mika et al. (2006), setup times in the context of the resource-constrained project scheduling problem have received significantly less attention than setup times in machine scheduling problems (cf. Allahverdi et al. (2008) for a recent survey on machine scheduling problems with setup times).

According to Mika et al. (2006), setup times in project scheduling have first been considered in an unpublished paper by Kaplan (1991) in which she tackles the resource-constrained project scheduling problem with preemption where setup times are required whenever a preempted activity is restarted. Apart from this, for example, Vanhoucke (2008) considers the RCPSP with preemption and sequence-independent setup times while Neumann et al. (2003) deal with the RCPSP with sequence-dependent (and resource-dependent) setup (or changeover) times and time windows. Similarly, the resource-constrained project scheduling problem with sequence- and resource-dependent setup (or transfer) times is the main focus of the PhD thesis of Krüger (2009). Additionally, Krüger (2009) introduces the RCPSP with generalized setup times as described in Section 5.1.

In the following, a classification of different types of setup times in the context of the resource-constrained project scheduling problem is given. In this classification as it has been introduced by Mika et al. (2006), setup times are classified according to various categories including activity vs. class setups, separable vs. inseparable setups, as well as sequence-independent, sequence-dependent, and schedule-dependent setups. Afterward, due to its importance to this thesis, the RCPSP with sequence- and resource-dependent setup times is described as it has been considered by Krüger (2009) in her PhD thesis.

Classification of Setup Times

According to the classification of setup times by Mika et al. (2006), if a setup is required by an individual activity, it is referred to as an activity setup. Otherwise, if a setup is required by a group of activities, it is referred to as a class setup. In this case, it is sufficient to execute the setup only once before all activities of the respective group can be processed. Next, a setup is called inseparable if the activity has to be started directly after the setup has been executed. Similarly, a setup is called separable if the activity does not have to start immediately after the setup has been performed. It should be noted, however, that also in this case no other activities can be executed on a resource between the setup of the resource and the activity for which the setup has been performed.

Next, if sequence-independent setup times s_j for activities $j \in V_*$ are given, these setup times only depend on activity j itself (i.e. they are independent of the sequence in which the activities are processed on the required resource). On the other hand, sequence-dependent setup times s_{ij} depend on the activity $i \in V_0$ that has been processed directly before activity $j \in V_*$ on the required resource (i.e. they depend on the sequence in which the activities are processed on the required resource). Finally, if more than one type of resources is available, the setup times might also be resource-dependent.

Example 3.4 We consider a small project consisting of $n = 3$ real activities with unit processing times (i.e. $p_i = 1$ holds for $i = 1,2,3$) as well as

$r = 1$ renewable resource with a capacity of $R_1 = 1$ such that each real activity $i = 1,2,3$ requires $r_{i1} = 1$ units of this resource. Additionally, the precedence constraints $1 \rightarrow 2$ and $1 \rightarrow 3$ between real activities as well as the sequence-independent setup times $s_1 = 1$, $s_2 = 2$, and $s_3 = 1$ are given. Two feasible schedules for this project with sequence-independent setup times are displayed in Figure 3.6.

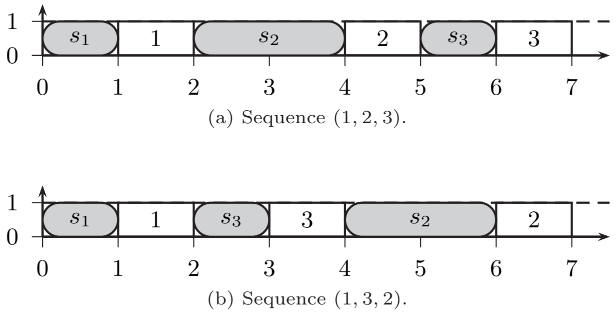


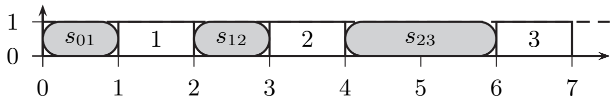
Figure 3.6: Two schedules for the project from Example 3.4 with sequence-independent setup times for the sequences $(1, 2, 3)$ and $(1, 3, 2)$.

Here, it can be seen that the sequence in which the activities are processed on the resource has no influence on the setup times of the activities and both schedules have a makespan of $C_{max} = 7$. Now, the sequence-independent setup times are replaced by sequence-dependent setup times s_{ij} as given in Table 3.4.

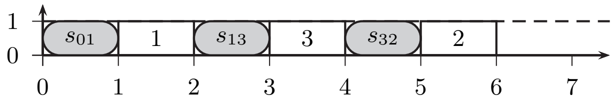
	0	1	2	3	4
0	0	1	1	1	0
1	0	0	1	1	0
2	0	0	0	2	0
3	0	0	1	0	0
4	0	0	0	0	0

Table 3.4: Sequence-dependent setup times s_{ij} between all pairs of activities $i,j = 0, \ldots, 5$ for the project from Example 3.4.

Two schedules for the resulting problem with sequence-dependent setup times are displayed in Figure 3.7. Unlike in the case of sequence-independent



(a) Sequence $(1, 2, 3)$.



(b) Sequence $(1, 3, 2)$.

Figure 3.7: Two schedules for the project from Example 3.4 with sequence-dependent setup times for the sequences $(1, 2, 3)$ and $(1, 3, 2)$.

setup times, the setup times of the activities are influenced by the direct predecessor of each activity on the resource, i.e. they depend on the sequence in which the activities are processed on the resource. As a result of this, the makespan of the schedule for the sequence $(1, 2, 3)$ is $C_{max} = 7$ while it is only $C_{max} = 6$ for the sequence $(1, 3, 2)$. $\square$

Finally, schedule-dependent setup times for activities not only depend on the sequence in which activities are processed on a specific resource but also on the assignment of resources to certain predecessors of the activity. Here, Mika et al. (2006) refer to the resources by which the activities are actually processed as multi-purpose resources (i.e. resources that can be used to process several different activities) and introduce an additional type of resources, referred to as setup-required resources. These setup-required are required to process particular activities on multi-purpose resources (i.e. each setup-required resource is dedicated to one particular activity) and can either be available in specific locations from the start of the project or, alternatively, they can be the product of some activities. Now, schedule-dependent setup times depend on when and where these setup-required resources are available to be transferred to the multi-purpose resource required by the corresponding activity.

Next, Mika et al. (2006) distinguish between precedence-independent and precedence-dependent setup times. In the case of a precedence-independent setup, the setup of the resource required by an activity can already be started before all predecessors of this activity have been completed. Alternatively, in the case of a precedence-dependent setup, all predecessors of an activity have to be completed before the resources required by this activity can be set up.

Another classification can be made if activities require multiple units of a resource. In this case the setup can be performed as an undivided setup (i.e. all resource units have to be set up at the same time) or as a divided setup (i.e. the resource units can be set up separately). Also, an activity might require multiple types of resources. In this case, the setup times can be classified as either synchronous setups (i.e. all setups have to be performed in parallel and require the time of the longest setup time), semi-synchronous setups (i.e. the setups are still performed in parallel but retain their individual setup times), or asynchronous setups (i.e. the setups can be performed independently).

Finally, Mika et al. (2006) introduce auxiliary resources as an additional type of resources used only to set up other resources. In contrast to the setup-required resources introduced above for the schedule-dependent setup times, auxiliary resources are not only used to set up resources for exactly one activity but can be used for setting up resources for various activities. If more than one type of auxiliary resources is available and the resources can be set up in alternative ways by these auxiliary resources, setup modes can be introduced in order to model these alternatives.

RCPSP with Sequence- and Resource-Dependent Setup Times

Now, the resource-constrained project scheduling problem with sequence- and resource-dependent setup (or transfer) times is introduced in more detail due to its importance to this thesis. This problem has, for example, been considered by Krüger (2009) in her PhD thesis as well as in a paper by Krüger and Scholl (2009). It should be noted that Krüger (2009) refers to setup times as transfer times Δ_{ijk} that occur if resource $k \in \mathcal{R}$ is transferred from activity $i \in V_0$ to activity $j \in V_*$. For this, she models the transfer of resources between activities as resource flows.

Example 3.5 We consider the project described in Example 3.3 and extend it by transfer times Δ_{ijk} between all pairs of activities $i \in V_{\text{all}}$ and $j \in V_{\text{all}}$ for resource $k = 1$. These transfer times Δ_{ijk} are given in Table 3.5.

Now, based on the feasible AON-flow network displayed in Figure 3.4, a left-justified schedule for this project can be calculated based on the longest

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	2	1	3	0
2	0	2	0	2	3	0
3	0	1	2	0	4	0
4	0	3	3	4	0	0
5	0	0	0	0	0	0

Table 3.5: Transfer times Δ_{ijk} between all pairs of activities $i \in V_{\text{all}}$ and $j \in V_{\text{all}}$ for resource $k = 1$ for the project from Example 3.5.

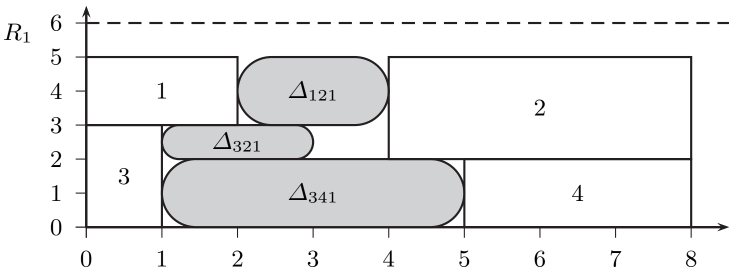


Figure 3.8: A left-justified schedule based on the feasible AON-flow network from Figure 3.4 for the project considered in Example 3.5.

paths as described in Example 3.3. For this, each arc (i, j) representing a resource transfer of resource 1 from activity $i \in V_0$ to activity $j \in V_*$ is weighted with the time $p_i + \Delta_{ij1}$ while all remaining arcs (i, j) representing precedence constraints between activities $i \in V_{\text{all}}$ and $j \in V_{\text{all}}$ are weighted with the processing time p_i of activity i. Then, the left-justified schedule with the makespan $C_{max} = 8$ displayed in Figure 3.8 can be calculated for this project. □

In the following, a mixed-integer linear programming formulation for this problem is introduced as it has been modeled by Krüger (2009). For this formulation, additional subsets V_{r_i} and V_{s_i} are introduced for each activity $i \in V_{\text{all}}$ such that the set V_{r_i} contains all activities to which resource units can be transferred from activity i (i.e. all activities $j \in V_*$ that are no direct or indirect predecessors of activity i) while the set V_{s_i} contains all activities from which resource units can be transferred to activity i (i.e. all activities $h \in V_0$ that are no direct or indirect successors of activity i).

Additionally, the following variables are required for this mixed-integer linear programming formulation: Integer variables C_i for all activities $i \in V_{\text{all}}$ denoting the completion time of activity i, integer variables f_{ijk} for all activities $i \in V_0$ and $j \in V_{r_i}$ as well as for all resources $k \in \mathcal{R}$ denoting the amount of resource k transferred from activity i to activity j, and binary variables x_{ijk} for all activities $i \in V_0$ and $j \in V_{r_i}$ as well as for all resources $k \in \mathcal{R}$ such that

$$x_{ijk} = \begin{cases} 1, & \text{if resource } k \text{ is transferred from activity } i \text{ to activity } j \\ 0, & \text{otherwise.} \end{cases}$$

Using these variables, the mixed-integer linear programming formulation (3.20) through (3.29) is defined by Krüger (2009).

$$\begin{aligned}
\min \quad & C_{n+1} && && (3.20)\\
\text{s.t.} \quad & C_j - C_i \geq p_j && ((i,j) \in A) && (3.21)\\
& C_i + \Delta_{ijk} + p_j \leq C_j + T(1 - x_{ijk}) && \begin{pmatrix} i \in V_0;\, j \in V_{r_i}; \\ k \in \mathcal{R} \end{pmatrix} && (3.22)\\
& f_{ijk} \leq \min\{r_{ik}, r_{jk}\} \cdot x_{ijk} && \begin{pmatrix} i \in V_0;\, j \in V_{r_i}; \\ k \in \mathcal{R} \end{pmatrix} && (3.23)\\
& x_{ijk} \leq f_{ijk} && \begin{pmatrix} i \in V_0;\, j \in V_{r_i}; \\ k \in \mathcal{R} \end{pmatrix} && (3.24)\\
& \sum_{h \in V_{s_i}} f_{hik} = r_{ik} && (i \in V_*;\, k \in \mathcal{R}) && (3.25)\\
& \sum_{j \in V_{r_i}} f_{ijk} = r_{ik} && (i \in V_0;\, k \in \mathcal{R}) && (3.26)\\
& C_i \in \mathbb{N} && (i \in V_{\text{all}}) && (3.27)\\
& f_{ijk} \in \mathbb{N} && \begin{pmatrix} i \in V_0;\, j \in V_{r_i}; \\ k \in \mathcal{R} \end{pmatrix} && (3.28)\\
& x_{ijk} \in \{0, 1\} && \begin{pmatrix} i \in V_0;\, j \in V_{r_i}; \\ k \in \mathcal{R} \end{pmatrix} && (3.29)
\end{aligned}$$

As before, the objective used in this mixed-integer linear programming formulation is to minimize the makespan (3.20) as denoted by the completion time C_{n+1} of dummy end activity $n+1$. Next, Inequalities (3.21) ensure that all precedence constraints $(i,j) \in A$ between activities $i \in V_{\text{all}}$ and $j \in V_{\text{all}}$ are observed while inequalities (3.22) ensure that activity $j \in V_*$ can only start at time $S_j = C_j - p_j \geq C_i + \Delta_{ijk}$ if at least one unit of resource $k \in \mathcal{R}$ is transferred from activity $i \in V_0$ to activity j (i.e. if $x_{ijk} = 1$ holds).

Inequalities (3.23) and (3.24) are introduced in order to ensure that $x_{ijk} = 1$ holds for $i \in V_0$, $j \in V_{r_i}$, and $k \in \mathcal{R}$ if and only if at least one unit of resource k is transferred from activity i to activity j. Finally, equations (3.25) ensure that the amount of incoming resource units of resource $k \in \mathcal{R}$ to activity $i \in V_*$ is equal to the resource requirements r_{ik} of this activity while equations (3.26) ensure that the amount of outgoing resource units of resource $k \in \mathcal{R}$ from activity $i \in V_0$ is equal to the resource requirements r_{ik} of this activity.

3.1.6 Further Extensions

Apart from the multi-mode RCPSP and the RCPSP with setup times introduced above, various other extensions for the classical RCPSP have been researched. In the following, a short selection of these extensions is described. A more detailed overview of these as well as other extensions is given in the surveys and books mentioned at the beginning of this chapter.

First of all, it might be possible that activities that have been started do not have to be processed without interruption in their entirety but can instead be interrupted during their execution and continued at a later time. This is referred to as preemption of activities and, in the context of the RCPSP, has been described, for example, by Kaplan (1988).

Next, apart from the renewable resources used throughout this chapter which have a constant availability for each time period of the planning horizon, various other types of resources might be used. These include, for example, non-renewable resources of which only a certain quantity is available for the duration of the whole project, as well as doubly-constrained resources which are limited both, for each time period as well as for the whole project (cf. Słowiński (1981)). It should be noted that doubly-constrained resources can also be incorporated by one renewable as well as one non-renewable resource. If the availability of resources is not constant over the planning horizon, time-dependent resource profiles can be used as described, for example, by Bartusch et al. (1988). Finally, for partially-renewable resources, the planning horizon is partitioned into subsets of time periods such that each of these subsets is assigned a total amount of available resource units (cf. Böttcher et al. (1999)). As stated by Brucker and Knust (2006), partially-renewable resources are a generalization of the concepts of renewable and non-renewable resources, which can both be modeled as partially-renewable resources.

In the classical resource-constrained project scheduling problem, if a precedence constraint $(i, j) \in A$ between two activities $i \in V_{\text{all}}$ and $j \in V_{\text{all}}$ is given, inequality $S_i + p_i \leq S_j$ has to hold, i.e. activity i has to be completed before activity j can be started. More generalized precedence constraints are described by Bartusch et al. (1988) such that, between two activities $i \in V_{\text{all}}$ and $j \in V_{\text{all}}$, a minimum time lag l_{ij}^{min} as well as a maximum time lag l_{ij}^{max} might be given. In this case, the start-start relations $S_i + l_{ij}^{min} \leq S_j$ and $S_i + l_{ij}^{max} \geq S_j$ have to hold, i.e. activity j has to be started in the time window $[S_i + l_{ij}^{min}; S_i + l_{ij}^{max}]$ after the start of activity i. Similarly, this concept can be extended to start-finish, finish-start, and finish-finish relations between the activities. Additionally, release dates and deadlines can be modeled using generalized precedence constraints between the dummy start activity 0 and other activities $j \in V$ such that activity j can only start after its release date $r_j = l_{0j}^{min}$ and has to be finished before its deadline $d_j = l_{0j}^{max} + p_j$ (cf. Bartusch et al. (1988)).

Finally, various other objective functions beside the makespan have been used in the context of the RCPSP. For example, some important time-based objective functions are based on the lateness $L_i = C_i - d_i$, the tardiness $T_i = \max\{0, C_i - d_i\}$, or the earliness $E_i = \max\{0, d_i - C_i\}$ of activities $i \in V_{\text{all}}$ (cf. Brucker and Knust (2006)). For all of these values, d_i is the due date of activity $i \in V_{\text{all}}$. Similar to the deadlines introduced above, due dates also refer to the latest time an activity should be finished. Unlike deadlines, however, due dates are not hard constraints but can be violated which results in a penalty. A classification of alternative objective functions is, for example, given by Neumann et al. (2006).

3.2 Solution Approaches

In this section, an overview of various solution approaches for the classical resource-constrained project scheduling problem and its extensions is given. Here, first of all, some results concerning the time complexity of the RCPSP are discussed in Section 3.2.1. Then, Section 3.2.2 introduces a schedule classification scheme for the RCPSP. Finally, Section 3.2.3 deals with heuristic solution approaches for the RCPSP while Section 3.2.4 describes other solution approaches, including exact methods as well as lower bounds.

3.2.1 Time Complexity

The classical RCPSP with the makespan objective function is a strongly NP-hard problem. This follows from the fact that the RCPSP is a generalization of the well-known job-shop scheduling problem, which has been shown to be strongly NP-hard by Garey et al. (1976) for the makespan objective function. As a result of this, it is unlikely that an exact polynomial time algorithm exists to solve the problem. Moreover, Blazewicz et al. (1983) show that already the problem with only $r = 2$ renewable resources with capacities of $R_1 = 1$ and $R_2 = 2$ as well as fixed resource requirements $r_{i2} = 1$ for all real activities $i \in V$ (i.e. at most two activities can be processed simultaneously), unit processing times, and precedence chains is NP-hard in the strong sense by a transformation of the problem 3-PARTITION. Only some even more simple problems without precedence constraints have been shown to be polynomially solvable (cf. Blazewicz et al. (1983)).

While feasible (not necessarily optimal) solutions for the classical RCPSP can be calculated in polynomial time, for example by scheduling all activities with respect to a topological order, Bartusch et al. (1988) show that even deciding whether a feasible solution exists is NP-complete for the RCPSP with maximum time lags. Similarly, Kolisch and Drexl (1997) show that the problem of deciding whether a feasible schedule exists for the multi-mode RCPSP with at least two non-renewable resources is also NP-complete.

3.2.2 Classification of Schedules

Schedules for instances of the resource-constrained project scheduling problem generated by different algorithms can often be classified as belonging to different subsets of schedules. Based on a classification scheme introduced by Conway et al. (1967) for the job-shop scheduling problem, Sprecher et al. (1995) classify schedules for the RCPSP as feasible, semi-active, active, and non-delay schedules.

Here, while the subset of feasible schedules contains all schedules that are feasible with respect to the given precedence and resource constraints as introduced above, the remaining classifications are based on the definition of left shifts of activities. For a feasible schedule S, a left shift of an activity $i \in V$ transforms the schedule into a feasible schedule S' with $S'_i < S_i$ and $S'_j = S_j$ for all other activities $j \in V$ with $j \neq i$ (cf. Wiest (1964)). Based on this definition Wiest (1964) defines an one-period left shift of an activity $i \in V$ as a left shift such that $S_i - S'_i = 1$ holds for activity i and a local left shift as a series of one-period left shifts such that each intermediate schedule is feasible. If one of the intermediate schedules obtained by one-period left shifts of an activity i is not feasible but the resulting schedule S' is feasible, this is referred to as a global left shift.

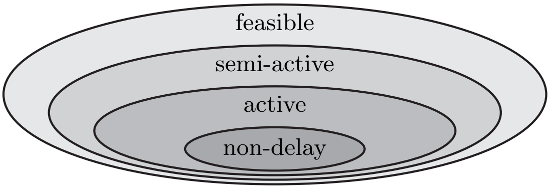


Figure 3.9: The relation between the four subsets of feasible schedules, active schedules, semi-active schedules, as well as non-delay schedules for the RCPSP as described by Sprecher et al. (1995).

Based on these definitions, Sprecher et al. (1995) refer to a schedule in which no local left shift is possible for any activity $i \in V$ as a semi-active schedule. Similarly, an active schedule is a schedule in which no local or global left shift is possible for any activity $i \in V$. Finally, a schedule is referred to as a non-delay schedule if no activities can be shifted to the left using either local or global left shifts even if preemption is allowed. The relation between these four subsets of schedules is displayed in Figure 3.9.

Example 3.6 In order to visualize the different classes of schedules introduced above, Sprecher et al. (1995) introduce an example instance for the resource-constrained project scheduling problem consisting of $n = 5$ real activities as well as $r = 1$ renewable resource with a capacity of $R_1 = 2$. The processing times and resource requirements of the activities are given in Table 3.6 while the precedence constraints are displayed in the activity-on-node network in Figure 3.10.

i	1	2	3	4	5
p_i	2	1	1	1	2
r_{i1}	1	2	1	2	1

Table 3.6: Processing times p_i and resource requirements r_{ik} of the $n = 5$ real activities $i = 1, \dots, 5$ for the project from Example 3.6.

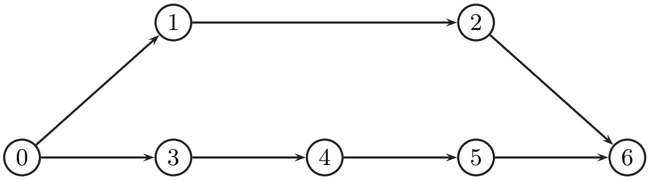


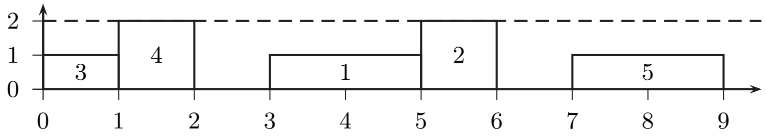
Figure 3.10: Activity-on-node network for the project considered in Example 3.6 consisting of $n = 5$ real activities $i = 1, \dots, 5$ as well as the two dummy activities 0 and 6.

Based on this example instance, the four schedules displayed in Figure 3.11 visualize the four different subsets of feasible, active, semi-active, and non-delay schedules. According to Sprecher et al. (1995), the active schedule displayed in Figure 3.11(c) is the only optimal schedule for this project. For this reason, because this schedule is not a non-delay schedule (the first part of activity 1 could feasibly start at time $t = 0$ if preemption were allowed) it can be concluded that the subset of non-delay schedules does not always contain an optimal solution. □

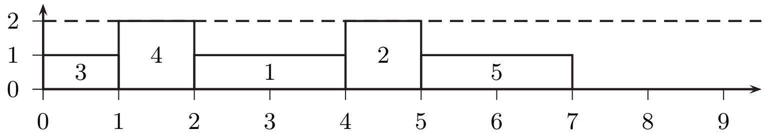
Finally, it is important to note that the set of active schedules always contains an optimal solution for regular objective functions such as the makespan objective function (cf., for example, Brucker and Knust (2006)).

3.2.3 Heuristic Approaches

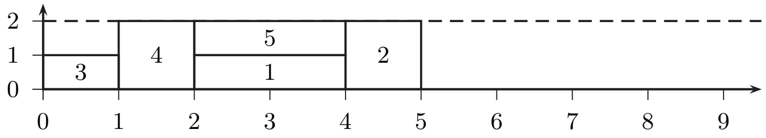
Some of the earliest heuristic methods for the resource-constrained project scheduling problem have been introduced by Kelley (1963) and are priority rule based scheduling schemes. These heuristics consist of a schedule generation scheme used to actually schedule the activities, as well as a priority rule that is used to select the next activity to be scheduled. Here, two schedule



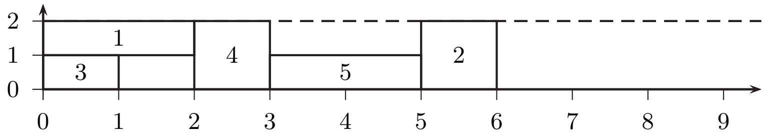
(a) Feasible schedule.



(b) Semi-active schedule.



(c) Active schedule.



(d) Non-delay schedule.

Figure 3.11: Schedules for the project considered in Example 3.6 visualizing the four different subsets of feasible schedules (cf. Figure (a)), semi-active schedules (cf. Figure (b)), active schedules (cf. Figure (c)), and non-delay schedules (cf. Figure (d)) as described by Sprecher et al. (1995).

generation schemes have been introduced by Kelley (1963), referred to as the serial schedule generation scheme as well as the parallel schedule generation scheme. As pointed out by Kolisch (1996a), however, the parallel schedule generation scheme for the RCPSP most often used in scientific literature has been published by Brooks (Bedworth and Bailey, 1982).

In the following, the two schedule generation schemes are described in more detail. Here, first of all, the serial schedule generation scheme is outlined in Algorithm 3.1 (cf. Brucker and Knust (2006)).

Algorithm 3.1: Serial Schedule Generation Scheme

```
1  Let $E_1$ be the set of all activities without predecessor;
2  for $\lambda := 1$ to $n$ do
3      Choose an activity $j \in E_\lambda$;
4      $t := \max_{(i,j)\in A} \{S_i + p_i\}$;
5      while a resource $k$ with $r_{jk} > R_k(\tau)$ for a $\tau \in [t, t + p_j[$ exists do
6          Calculate the smallest time $t_k > t$ such that $j$ can be scheduled in
           the interval $[t_k, t_k + p_j[$ if only resource $k$ is considered and set
           $t := t_k$;
7      end
8      Schedule $j$ in the interval $[S_j, C_j[ := [t, t + p_j[$;
9      Update the current resource profile by setting $R_k(\tau) := R_k(\tau) - r_{jk}$ for
       all $k = 1, \dots, r$ and $\tau \in [t, t + p_j[$;
10     Let $E_{\lambda+1} := E_\lambda \setminus \{j\}$ and add to $E_{\lambda+1}$ all successors $i \notin E_\lambda$ of $j$ for
       which all predecessors are scheduled;
11 end
```

The main idea of the serial schedule generation scheme is to schedule exactly one activity in each iteration, i.e. a total of n iterations (or stages) are required in order to schedule all real activities $i \in V$. In this algorithm, the set E_λ denotes the set of all eligible activities that can be scheduled in iteration λ. For the serial schedule generation scheme, an eligible activity $j \in E_\lambda$ is an activity for which all predecessors $i \in V_{\text{all}}$ with $(i, j) \in A$ have already been scheduled, i.e. in each iteration λ, the set E_λ contains all precedence-feasible activities. Furthermore, $R_k(t)$ denotes the amount of units of resource $k \in \mathcal{R}$ available in the time interval $[t, t + 1[$. Here, the amount of available resource units $R_k(t)$ in the time interval $[t, t + 1[$ depends on activities $i \in V$ with $S_i \leq t < S_i + p_i$ that have already been scheduled to be processed in this time interval.

Now, in each iteration λ, an activity $j \in V$ is selected from the set of eligible activities E_λ based on a priority rule in line 3 of the algorithm. The earliest precedence-feasible start time of this activity is then calculated as the latest time at which one of its predecessor activities is completed in line 4. Then, in lines 5 to 7, the earliest precedence- and resource-feasible time at which activity j can be started is calculated. Finally, the activity is scheduled to start at this time in line 8, the resource profiles are updated in line 9 and the new set of eligible activities $E_{\lambda+1}$ for the next iteration is calculated in line 10. Kolisch (1996a) has shown that the serial schedule generation scheme using any priority rule generates schedules belonging to the set of active schedules.

The time complexity of the serial schedule generation scheme primarily depends on how the time points t are calculated in line 6. Here, if all possible time points t_k are tested (i.e. by iteratively incrementing the values by one until a feasible time point has been found), the computational complexity of the algorithm is given by $\mathcal{O}(nTr)$, i.e. the algorithm runs in pseudo-polynomial time. A more efficient approach is to store the jump points at which the resource profiles change and only test these time points t_k. Then, because each activity that is scheduled adds at most two new jump points, the time complexity of the algorithm is reduced to $\mathcal{O}(n^2 r)$, i.e. the algorithm runs in polynomial time.

Next, the parallel schedule generation scheme is described. The main idea of the parallel schedule generation scheme is to consider decision points t_λ such that $t_{\lambda-1} < t_\lambda$ holds. Then, in each iteration λ (or, respectively, at each decision point t_λ), a subset of eligible activities is scheduled. For this algorithm, a set A_λ of active activities (i.e. activities $i \in V$ with $S_i \leq t_\lambda < S_i + p_i$ that have already been scheduled and are still active at decision point t_λ) as well as a set E_λ of precedence- and resource-feasible activities that have not yet been scheduled are associated with each iteration λ. Additionally, $R_k(t)$ again denotes the amount of units of resource $k \in \mathcal{R}$ available in the time interval $[t, t+1[$. The parallel schedule generation scheme is outlined in Algorithm 3.2 (cf. Brucker and Knust (2006)).

Here, in each iteration λ (i.e. at each decision point t_λ), the parallel schedule generation scheme schedules eligible activities from the set E_λ until either all activities from the set have been scheduled or until no more activities can be scheduled resource-feasibly at decision point t_λ. For this, a priority rule is used to select the next eligible activity $j \in E_\lambda$ in line 5, which can then be scheduled immediately to start at time $S_j = t_\lambda$ in line 6. Then, in line 7, the resource profiles are updated and activity j is added to the set of active activities A_λ. Afterward, all activities $i \in E_\lambda$ that can not be scheduled resource-feasibly at time t_λ any more are removed from the current set E_λ in line 8. After no more eligible activities can be scheduled, the next decision point $t_{\lambda+1}$ is calculated in line 10 as the earliest time at which an active activity from the set A_λ will be completed. Finally, the sets $A_{\lambda+1}$ and $E_{\lambda+1}$ for the next iteration are calculated in line 12. Unlike the serial schedule generation scheme, the parallel schedule generation scheme generates schedules from the set of non-delay schedules for any priority rule (cf. Kolisch (1996a)). For this reason, the parallel schedule generation scheme may not always be able to generate an optimal solution.

Algorithm 3.2: Parallel Schedule Generation Scheme

```
1  $\lambda := 1$; $t_1 := 0$; $A_1 := \emptyset$;
2  Let $E_1$ be the set of all activities $i$ without predecessor and $r_{ik} \leq R_k(\tau)$ for
   $k = 1, \ldots, r$ and all $\tau \in [0, p_i[$;
3  while not all activities are scheduled do
4      while $E_\lambda \neq \emptyset$ do
5          Choose an activity $j \in E_\lambda$;
6          Schedule $j$ in the interval $[S_j, C_j[ := [t_\lambda, t_\lambda + p_j[$;
7          Update the current resource profiles and add $j$ to $A_\lambda$;
8          Update the set $E_\lambda$ by eliminating $j$ and all activities $i$ with
           $r_{ik} > R_k(\tau)$ for some resource $k$ and a $\tau \in [t_\lambda, t_\lambda + p_i[$;
9      end
10     Let $t_{\lambda+1} = \min_{i \in A_\lambda} \{S_i + p_i\}$ be the minimal completion time of all active
       activities;
11     $\lambda := \lambda + 1$;
12     Calculate the new sets $A_\lambda$ and $E_\lambda$;
13 end
```

Similar to the serial schedule generation scheme, the parallel schedule generation scheme also has a time complexity of $\mathcal{O}(n^2 r)$ because at most n decision points t_λ have to be considered (i.e. one for each activity). Further results related to the serial as well as the parallel schedule generation scheme are reported by Kolisch (1996a).

It should be noted that both, the serial as well as the parallel schedule generation scheme, can generate the schedules either in a forward direction, in a backward direction, or bidirectional (cf. Brucker and Knust (2006)). Here, the forward scheme schedules the activities from left to right as described above while the backward scheme schedules the activities from right to left. The bidirectional scheme schedules the activities either from left to right or from right to left and then transforms the schedule into an active schedule at the end.

Some important priority rules that can be used together with the schedule generation schemes introduced above are summarized by Kolisch (1996b) and Klein (2000). These include, for example, the priority rules SPT (shortest processing time) and LPT (longest processing time) which select activities based on their processing times p_i, as well as the priority rules MIS (most immediate successors), LIS (least immediate successors), MTS (most total successors), and LTS (least total successors) which select activities based on the number of immediate or total successors. Finally, the priority rules LFT (latest finish time), EST (earliest start time), or MSLK (minimum slack) use values based on the network structure as well as the critical path of the project in order to prioritize the activities.

When using schedule generation schemes in order to generate schedules, single-pass and multi-pass methods can be distinguished. On the one hand, single-pass methods employ one schedule generation scheme (i.e. either the serial or the parallel schedule generation scheme using either the forward, backward, or bidirectional scheme) as well as one specific priority rule in order to generate exactly one schedule for a given problem instance. Computational results for these schemes and various priority rules are for example reported by Kolisch (1996a) and Klein (2000).

On the other hand, multi-pass methods generally use a combination of schedule generation schemes and priority rules in order to generate multiple schedules and select the best solution from these schedules. For example, Boctor (1990) presents a multi-pass heuristic using multiple priority rules in order to generate schedules. An alternative multi-pass heuristic using a forward-backward scheme has been introduced by Li and Willis (1992). Finally, sampling methods again use only one schedule generation scheme and one priority rule but are extended by a random factor such that activities $j \in V$ are selected from the set of eligible activities based on the computed priority value $v(j)$ as well as on a selection probability $p(j)$. An overview of different types of sampling methods is, for example, given by Kolisch (1996a) as well as Kolisch and Drexl (1996).

Apart from these solution approaches based on priority rules, another important class of solution approaches are metaheuristic approaches. These include, for example, local search algorithms such as tabu search or simulated annealing, as well as genetic algorithms and various other approaches. Often, the serial or parallel schedule generation scheme is used to generate starting solutions for these metaheuristics.

First of all, local search algorithms based on tabu search for the classical resource-constrained project scheduling problem have been introduced, for example, by Baar et al. (1998) as well as Artigues et al. (2003). While Baar et al. (1998) describe a tabu search algorithm that tries to eliminate critical arcs as well as another tabu search algorithm based on schedule schemes, Artigues et al. (2003) present a tabu search algorithm based on a resource flow representation. In this algorithm, they implement neighborhoods based on an insertion algorithm that deletes activities from a resource flows and then reinserts them into the resource flow. Another local search approach based on a resource flow representation as well as as an insertion algorithm for the RCPSP with sequence-dependent and resource-independent setup times has been introduced by Quilliot and Toussaint (2012). Next, a tabu search algorithm for the multi-mode RCPSP with schedule-dependent setup times is, for example, presented by Mika et al. (2008). Local search algorithms based on simulated annealing for the RCPSP as well as for the MRCPSP are described, for example, by Bouleimen and Lecocq (2003).

A genetic algorithm for the resource-constrained project scheduling problem based on an activity-list representation has been introduced by Hartmann (1998). This algorithm is then extended by Hartmann (2002) to include a gene that denotes if the serial or the parallel schedule generation scheme is used to transform the activity list into a schedule. Another genetic algorithm based on activity-lists is introduced by Valls et al. (2008). This genetic algorithm is extended by a double-justification procedure introduced by Valls et al. (2005) that first shifts all activities in a schedule to the right and then to the left in order to generate a better solution.

Finally, more recent approaches often incorporate several techniques into one solution approach for the resource-constrained project scheduling problem. For example, Debels et al. (2006) introduce a meta-heuristic that combines elements of a scatter search algorithm with elements of a heuristic method based on electromagnetism theory. An algorithm combining a genetic algorithm, path linking, and tabu search is described by Kochetov and Stolyar (2003) while a genetic algorithm combined with forward-backward improvement for the multi-mode RCPSP is introduced by Lova et al. (2009).

Recent evaluations of various heuristic approaches for the classical resource-constrained have, for example, been performed by Hartmann and Kolisch (2000) as well as by Kolisch and Hartmann (2006). Here, Kolisch and Hartmann (2006) conclude that hybrid approaches perform currently best when solving the RCPSP.

3.2.4 Other Approaches

Apart from heuristic approaches for solving the RCPSP, several exact solution methods have been introduced. While the earliest exact solution approaches often focused on mixed-integer linear programming formulations (cf., for example, Wiest (1963) and Pritsker et al. (1969)), the majority of recent developments use branch-and-bound algorithms.

The general idea of a branch-and-bound algorithm is to recursively divide a given problem into smaller subproblems such that, for each of these subproblems, a lower bound on the objective function value of the set of feasible solutions represented by this subproblem can be calculated. This enumeration of solutions if referred to as branching and results in a branching tree. Furthermore, an upper bound for the original problem is used to prune branches of the branching tree in order to reduce the search space. Here, if the lower bound of the subproblem represented by a tree node is larger than or equal to the current upper bound, the corresponding branch of the branching tree does not have to be continued. This pruning of branches is referred to as bounding.

As a result of this, a branch-and-bound algorithm relies on several features: a branching rule, an algorithm for calculating the lower bounds of subproblems, as well as an algorithm for calculating an (initial) upper bound. Here, first of all, any feasible solution for a given problem instance can be regarded as an upper bound for this instance. For this reason, an initial upper bound can, for example, be calculated by a heuristic algorithm and then be updated as better solutions are generated by the branch-and-bound algorithm such that at any time the currently best feasible solution is used as an upper bound. Next, a short overview of different branch-and-bound algorithms based on the branching rules employed in these algorithms as well as various lower bounds for the resource-constrained project scheduling problem is given.

One of the earliest branch-and-bound algorithms for the RCPSP has been presented by Stinson et al. (1978) (branching based on extension alternatives) and is discussed in detail and compared to other exact approaches by Patterson (1984). More recent branch-and-bound algorithms for the classical RCPSP have, for example, been introduced by Christofides et al. (1987) as well as by Demeulemeester and Herroelen (1992, 1997) (branching based on delaying alternatives), by Patterson et al. (1989) (branching based on precedence trees), and by Brucker et al. (1998) (branching based on schedule schemes). Similarly, branch-and-bound algorithms for the multi-mode resource-constrained project scheduling problem have, for example, been described by Hartmann and Drexl (1998) as well as Sprecher et al. (1997) and Sprecher and Drexl (1998).

Various constructive as well as destructive lower bounds for the resource-constrained project scheduling problem have, for example, been described by Klein and Scholl (1999). Here, one of the most basic constructive lower bounds is given by the length of a critical path of the project as described in Section 3.1.2. An LP-based lower bound has been introduced by Mingozzi et al. (1998) while an extension of this approach by constraint propagation into an destructive lower bound is described by Brucker and Knust (2000), Baptiste and Demassey (2004), and Demassey et al. (2005).

Finally, more recent developments for solving the RCPSP are based on hybrid approaches incorporating techniques based on constraint propagation and satisfiability testing (cf. Schutt et al. (2009)). This approach is extended by linear relaxations based on integer programming formulations as described by Berthold et al. (2010).

4 A Solution Approach for the HEP based on Priority Rules

After literature relating to the resource-constrained project scheduling problem as well as its extensions has been introduced in the previous chapter, a first solution solution approach for the problem of hospital evacuations as it has been described in Chapter 2 is presented in this chapter.

Before this solution approach is introduced, however, the problem of hospital evacuations is first modeled as a multi-mode resource-constrained project scheduling problem with resource transfers and blockings. In the following, a formal description of this model is introduced in Section 4.1. Here, in particular, the incorporation of resource transfers as well as blockings into the model is described in detail. Additionally, this section contains a mixed-integer linear programming formulation as well as further considerations regarding extensions of the model.

Afterward, in Section 4.2, the solution approach itself is introduced. This solution approach is based on a tabu search algorithm that generates schedules for mode selections using either a parallel or a serial schedule generation scheme. Here, in particular, the focus of this section is on the adaption of the schedule generation schemes as they have been described in the previous chapter. Also, we discuss some shortcomings of this solution approach in this section.

4.1 Model

In this section, a model for the problem of hospital evacuations is introduced. As described in Section 2.2.2, the HEP consists of scheduling the evacuation of all patients from their initial locations inside the hospital to safety zones (or staging areas) inside or outside the hospital using the available resources (i.e. assistants, aids, and building sections) such as to minimize the time required to evacuate all patients. The model proposed here is based on the multi-mode resource-constrained project scheduling problem and incorporates resource transfers as well as blockings. In the following, the model for the problem of hospital evacuations is described in Section 4.1.1. Afterward, a mixed-integer linear programming formulation for this model is introduced in Section 4.1.2 while further considerations related to the problem are discussed in Section 4.1.3.

4.1.1 Problem Description

In order to evacuate a patient from his initial location to a safety zone, various resources (i.e. assistants, aids, and building sections) are required. Here, each type of assistant (e.g. nurses, firemen, etc.) with different skills as well as each type of aid (e.g. stretchers, wheelchairs, etc.) is modeled as a renewable resource k with a capacity R_k equal to the amount of available resource units of this type. Furthermore, each building section (e.g. each corridor, stair, elevator, etc.) is modeled as a separate renewable resource k (i.e. there might be multiple resources of the same type) with a capacity R_k equal to the capacity of the respective building section. In the following, the set $\mathcal{R}$ contains all resources $k = 1, \ldots, r$ while the subsets $\mathcal{R}^{\text{asst}}$, $\mathcal{R}^{\text{aid}}$, and $\mathcal{R}^{\text{sect}}$ contain all resources that model assistants, aids, and building sections, respectively. Additionally, the set $\mathcal{R}^{\text{trf}} = \mathcal{R}^{\text{asst}} \cup \mathcal{R}^{\text{aid}}$ contains all resources that incur a transfer time if they are transferred between two jobs.

Now, the evacuation of a patient from his initial location (e.g. his sickroom) to a safety zone or staging area is regarded as a job j such that an overall of N jobs $j = 1, \ldots, N$ have to be scheduled. These jobs as well as a dummy source job 0 and a dummy sink job $N+1$ are represented by the set $J = \{0, 1, \ldots, N, N+1\}$. Associated with each job $j \in J$ are m_{j1} different modes $m_1 = 1, \ldots, m_{j1}$ as well as m_{j2} different modes $m_2 = 1, \ldots, m_{j2}$ such that mode m_1 is used to select the required assistants and aids for the evacuation of the patient (e.g. two assistants and a stretcher, or one assistant and a wheelchair, etc.) while mode m_2 is used to select the evacuation route along which the patient will be evacuated. In the following, modes m_1 are referred to as equipment modes and are included in a set $\mathcal{M}_{j1} = \{1, \ldots, m_{j1}\}$ for each job $j \in J$ while modes m_2 are referred to as route modes and are included in a set $\mathcal{M}_{j2} = \{1, \ldots, m_{j2}\}$ for each job j. Based on these modes, each job $j \in J$ is assigned an additional set $\mathcal{M}_j$ with

$$\mathcal{M}_j = \{(m_1, m_2) \mid m_1 \in \mathcal{M}_{j1} \text{ and } m_2 \in \mathcal{M}_{j2}\}$$

containing all combinations of modes $m_1 \in \mathcal{M}_{j1}$ and $m_2 \in \mathcal{M}_{j2}$. For the sake of simplicity, these mode combinations are abbreviated by modes $m = (m_1, m_2)$ in the following such that each job $j \in J$ is assigned an overall of m_j different modes $m = 1, \dots, m_j$ (with $m_j = m_{j1} \cdot m_{j2}$). It should be noted that the two dummy jobs 0 and $N+1$ are assigned only $m_0 = m_{N+1} = 1$ mode combination $(m_{01}, m_{02}) = (m_{N+1,1}, m_{N+1,2}) = (1, 1)$.

Next, as described in Section 2.2.2, the infrastructure of the hospital is partitioned into building sections such that the evacuation of a patient from his initial location to a safety zone along an evacuation route generally requires multiple of these sections. In this model, each evacuation route for a patient is represented by a chain of operations such that each operation corresponds to the transport of the patient through a specific building section (cf. Example 4.1). Thus, a total of n_j operations O_{uj} ($u = 1, \dots, n_j$) are associated with each job $j \in J$ such that precedence constraints $O_{uj} \rightarrow O_{vj}$ represent the sequence of operations modeling the selected evacuation route (i.e. the order in which the patient is transported through the building sections). It should be noted that these chains of operations indirectly model the underlying network structure of the infrastructure of the hospital. In the following, the operations O_{uj} with $j = 1, \dots, N$ and $u = 1, \dots, n_j$ are identified by the numbers $1, \dots, n$, i.e. there are a total of n operations. Additionally, a dummy source operation 0 is associated with dummy job 0 and a dummy sink operation $n+1$ is associated with dummy job $N+1$.

It should be noted that the differentiation between jobs (representing the evacuation of a patient) as well as operations (representing one step in the evacuation of a patient) differs from the concept of activities generally used in resource-constrained project scheduling. Instead, this differentiation is based on the concept of jobs and operations used in shop scheduling problems where each job consists of a chain of operations that have to be processed on dedicated machines. Here, while each operation in a shop scheduling problem only requires one machine (i.e. one resource unit), multiple resources in various quantities (i.e. assistants, aids, as well as sufficient amount of space in a specific building section) are required in order to process one operation in this problem.

Example 4.1 We consider an example of a hospital building consisting of six rooms (e.g. sickrooms) as well as two exits corresponding to the safety zones to which the patients have to be evacuated. In this example, the corridor is partitioned into the eight sections C1 through C8. The floor plan of this hospital building is displayed in Figure 4.1.

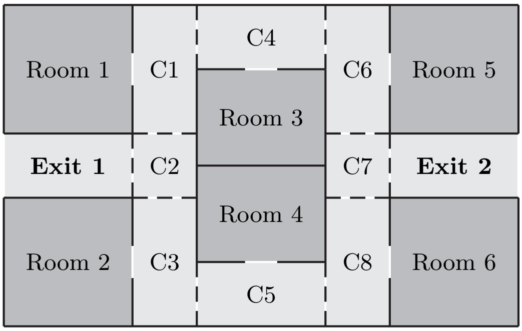


Figure 4.1: Floor plan of the hospital building considered in Example 4.1. The hospital consists of six room, two exits, as well as a corridor that is partitioned into the eight sections C1 through C8.

$$\begin{tikzcd}
\text{Room 5} \arrow[r] & C6 \arrow[r] & C4 \arrow[r] & C1 \arrow[r] & C2 \arrow[r] & \text{Exit 1}
\end{tikzcd}$$

$$\begin{tikzcd}
\text{Room 5} \arrow[r] & C6 \arrow[r] & C7 \arrow[r] & \text{Exit 2}
\end{tikzcd}$$

Figure 4.2: Operations representing the two possible shortest evacuation routes from room 5 (i.e. the initial location of the patient) to the two exits (i.e. the safety zones).

Now, if a patient has to be evacuated from his initial location in room 5 to either of the two exits, two possible evacuation routes based on shortest paths between the corresponding locations can be calculated. The operations representing these evacuation routes are displayed in Figure 4.2 and consist of one operation at the initial location (e.g. representing the preparation of the patient for the transport), the transport itself through the corridor sections, as well as a postprocessing of the patient at the exit (e.g. to prepare the patient for the further transport to a sheltering facility). □

In the following, the set $V = \{1, \ldots, n\}$ contains all real operations of jobs $j = 1, \ldots, N$ while the set $V_{\text{all}} = \{0, 1, \ldots, n, n+1\}$ contains all operations including the two dummy operations 0 and $n + 1$. Additionally, for each job $j \in J$ as well as for each route mode $m_2 \in \mathcal{M}_{j2}$ of this job, an additional set $V_j(m_2)$ is introduced that contains all operations O_{uj} of job j that represent the evacuation route corresponding to route mode m_2. Also, because resource consumptions are represented by resource flows, additional sets V_0, V_*, V_{in}, and V_{out} are introduced. Here, the sets $V_0 = \{0, 1, \ldots, n\}$ and $V_* = \{1, \ldots, n+1\}$ are defined as before and are used to calculate the transfer of building sections between operations of different jobs. Next, the set V_{in} contains all operations which require incoming resource transfers of assistants and aids (i.e. before the patient can be evacuated from his initial location) and includes the first operation of each evacuation route of each job $j = 1, \ldots, N$. Similarly, the set V_{out} contains all operations from which outgoing resource transfers of assistants and aids are allowed (i.e. after the patient has arrived in a safety zone) and includes the last operation of each evacuation route of each job $j = 1, \ldots, N$. Finally, the sets $V'_{\text{out}} = V_{\text{out}} \cup \{0\}$ and $V'_{\text{in}} = V_{\text{in}} \cup \{n+1\}$ additionally contain dummy source operation 0 or dummy sink operation $n+1$, respectively.

Below, $\sigma(u) \in J$ corresponds to the job j to which an operation $u \in V_{\text{all}}$ belongs. Now, each operation $u \in V_{\text{all}}$ is assigned processing times p_{um} as well as resource requirements r_{umk} for resources $k \in \mathcal{R}$ depending on the selected mode combination $m \in \mathcal{M}_{\sigma(u)}$ with $m = (m_1, m_2)$ in which the corresponding job $\sigma(u)$ is processed. Here, because only operations $u \in V_j(m_2)$ belonging to the evacuation route denoted by mode $m_2 \in \mathcal{M}_{j2}$ of job $j \in J$ have to be scheduled, all other operations $v \in V_j(\tilde{m}_2)$ of this job with $\tilde{m}_2 \in \mathcal{M}_{j2}$ and $\tilde{m}_2 \neq m_2$ are assigned processing times $p_{vm} = 0$ as well as resource requirements $r_{vmk} = 0$ for all resources $k \in \mathcal{R}$, i.e. these operations do not have to be scheduled if mode combination m is selected. For operations $u \in V_j(m_2)$ of the job corresponding to the evacuation route denoted by mode m_2, however, the processing time p_{um} depends on the speed with which the patient can be evacuated (i.e. it depends on the assistants and aids used for the evacuation of the patient as denoted by mode $m_1 \in \mathcal{M}_{j1}$) as well as the length of the building section through which the patient has to be evacuated. The processing time p_{um} is then calculated as

$$p_{um} = \frac{(\text{length of the building section})}{(\text{speed of the required resources})} + (\text{action time}). \tag{4.1}$$

Here, the term action time describes the time required to perform an additional action in specific building sections along the evacuation route. For example, it might describe the time required to prepare the patient at his initial location, the time required to use an elevator, or the time required to open and pass through a door. It should be noted that the processing time of an operation can only be regarded as a minimum time that an operation has to be processed for in any feasible schedule. In particular, if a blocking occurs, an operation remains active until its successor operation can be started, i.e. all resources required by this operation remain occupied.

Similarly, the resource requirements r_{umk} of operations $u \in V_j(m_2)$ corresponding to the selected evacuation route as denoted by mode $m_2 \in \mathcal{M}_{j2}$ of job $j \in J$ also depend on the required assistants and aids as denoted by mode $m_1 \in \mathcal{M}_{j1}$. It is important to note that all of these operations $u \in V_j(m_2)$ require the same assistants and aids, i.e. these resources can not be exchanged for the duration of the complete job. As already described above, all required assistants and aids have to be transferred to the first operation representing the selected evacuation route (this operation is contained in the set V_{in}) and only become available again after the last operation representing the selected evacuation route has been completed (this operation is contained in the set V_{out}). Apart from assistants and aids, each operation also requires a specific amount of space in the building section corresponding to the operation. This amount of space again depends on the required assistants and aids used for the evacuation of the patient. For example, evacuating a patient on a stretcher requires more space than evacuating a patient that can walk with the help of an assistant.

Finally, dummy source operation 0 as well as dummy sink operation $n + 1$ have processing times $p_{01} = p_{n+1,1} = 0$ as well as resource requirements $r_{01k} = r_{n+1,1k} = R_k$ for all resources $k \in \mathcal{R}$ in the only available mode combination $m = 1$ of the corresponding dummy job. Thus, similar to the model described by Artigues et al. (2003), all resource units are initially located at dummy source operation 0 and are collected at dummy sink operation $n + 1$ at the end of the evacuation. In the following, it is assumed that all assistants and aids are available at a starting location associated with dummy source operation 0. From this starting location, they first have to be transferred to the initial locations of the patients before these can be evacuated. On the other hand, at the end of the evacuation, it is assumed that the assistants and aids do not have to be transferred to a location associated with dummy source operation $n + 1$. Instead, the evacuation ends as soon as the last patient has been evacuated.

Next, as stated above, precedence constraints $u \rightarrow v$ are inserted between two operations $u \in V_j(m_2)$ and $v \in V_j(m_2)$ representing two successive building sections of an evacuation route of job $j \in J$ as it is denoted by route mode $m_2 \in \mathcal{M}_{j2}$. Additionally, precedence constraints $0 \to v$ are inserted between dummy source operation 0 and the first operation $v \in V_{\text{in}}$ of each evacuation route while precedence constraints $u \to n+1$ are inserted between the last operation $u \in V_{\text{out}}$ of each evacuation route and dummy sink operation $n+1$.

Example 4.2 We consider an evacuation problem in which $N = 2$ patients have to be evacuated. While the first patient can be evacuated along two different evacuation routes represented by operations $1 \to 2 \to 3$ and $4 \to 5$, respectively, the second patient can only be evacuated along one evacuation route represented by operations $6 \to 7 \to 8$. The two corresponding jobs $j = 1, 2$ are displayed in Figure 4.3.

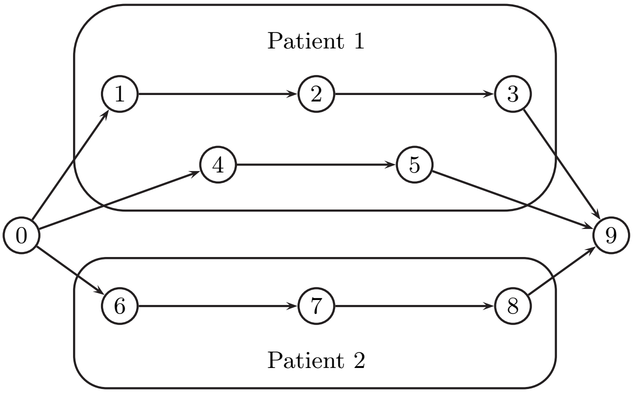


Figure 4.3: Activity-on-node network representing the operations of the two jobs considered in the problem instance from Example 4.2 as well as the two dummy operations 0 and 9.

In this example, dummy source operation 0 has to be processed before the first operation of each evacuation route (i.e. before operations 1, 4, and 6) while dummy sink operation 9 has to be processed after the last operation of each evacuation route (i.e. after operations 3, 5, and 8). □

In the following, the set $\mathcal{NB}$ contains all precedence constraints $u \to v$ between non-blocking operations (i.e. operation u ends immediately after it has been active for its processing time) while the set $\mathcal{B}$ contains all precedence constraints $u \to v$ between blocking operations (i.e. operation u can only end after operation v has started processing). Blocking constraints have been extensively studied in shop scheduling problems where blockings occur if no intermediate storage is available and a job remains on a machine until the next machine is available. For example, Hall and Sriskandarajah (1996) give an overview of blocking shop scheduling problems with a focus on flow-shop scheduling problems while Mascis and Pacciarelli (2002) consider the blocking job-shop scheduling problem. In the context of the resource-constrained project scheduling problem, blocking constraints have only been considered in a limited amount of works, e.g. by Pappert et al. (2010) in an assembly line scheduling problem as well as by Kröger (2013) in a rail-rail transshipment problem.

For the model presented here, all precedence constraints $0 \to v$ between dummy source operation 0 and the first operation of an evacuation route as well as all precedence constraints $u \to n+1$ between the last operation of an evacuation route and dummy sink operation $n+1$ are included in the set $\mathcal{NB}$ while all other precedence constraints between successive operations along an evacuation route are included in the set $\mathcal{B}$. Here, the latter precedence constraints have to be considered as blocking because a patient that is being evacuated physically stays in a building section until he arrives in the next building section.

Finally, an important part of the model is the transfer of assistants and aids between jobs. Whenever an assistant or an aid is transferred, the physical transfer requires a certain amount of time depending on the current location of the resource as well as the location of the patient within the hospital. This transfer time is denoted by $\varDelta_{uvk}$ for all operations $u \in V'_{\text{out}}$ and $v \in V'_{\text{in}}$ as well as for all resources $k \in \mathcal{R}^{\text{trf}}$ where u is the operation of the job from which the resource is being transferred and v is the operation of the job to which the resource is being transferred. Similar to the processing times described above, the transfer times $\varDelta_{uvk}$ depend on the physical distance between the building sections used by operations u and v as well as on the speed with which the resource can be transferred.

Here, because operations $u \in V_{\text{out}}$ represent the safety zones to which the patients have to be evacuated and operations $v \in V_{\text{in}}$ represent the initial locations of the patients, it can be assumed that $\varDelta_{uvk} > 0$ holds for all resources $k \in \mathcal{R}^{\text{trf}}$ between these operations. Furthermore, $\varDelta_{0vk} > 0$ can be assumed to hold for the transfer time of resource $k \in \mathcal{R}^{\text{trf}}$ from the starting location associated with dummy source operation 0 to the initial location of a patient associated with operation $v \in V_{\text{in}}$. Finally, $\Delta_{u,n+1,k} = 0$ can be assumed to hold for the transfer time of resource $k \in \mathcal{R}^{\text{trf}}$ from operation $u \in V'_{\text{out}}$ to dummy sink operation $n+1$. These latter transfer times denote that resources $k \in \mathcal{R}^{\text{trf}}$ do not have to be transferred to any specific location associated with dummy sink operation $n+1$ at the end of the evacuation.

It is important to note that aids have to be transported by assistants. In particular, an amount of μ_{kl} assistants $k \in \mathcal{R}^{\text{asst}}$ are required to transfer one unit of aid $l \in \mathcal{R}^{\text{aid}}$ to the patient. Here, if the assistants required to transport the aid are in a different location than the aid before the transfer, they first have to be transferred to the location of the aid before they can transport it to the location of the patient. For this reason, the actual transfer times can differ from the times Δ_{uvk} such that three scenarios have to be considered as described below.

First of all, if an assistant $k \in \mathcal{R}^{\text{asst}}$ becomes available at time C_u at operation $u \in V'_{\text{out}}$ and is directly transferred to operation $w \in V'_{\text{in}}$ (i.e. if the assistant does not have to transport an aid), inequality (4.2) has to hold for the starting time S_w of operation w.

$$S_w \geq C_u + \Delta_{uwk} \tag{4.2}$$

Otherwise, if an assistant $k \in \mathcal{R}^{\text{asst}}$ from operation $u \in V'_{\text{out}}$ is used to transport an aid $l \in \mathcal{R}^{\text{aid}}$ from operation $v \in V'_{\text{out}}$ to operation $w \in V_{\text{in}}$, two different scenarios have to be considered. In both scenarios, the assistant first has to be transferred to the location of the aid (unless they are in the same location already, i.e. if $\Delta_{uvk} = 0$ holds), which he can then transport to the location of the patient. Now, if $C_u + \Delta_{uvk} \geq C_v$ holds for operations u and v, i.e. if the assistant arrives at the location corresponding to operation v after this operation has been completed, inequality (4.3) has to hold for the starting time of operation w. In this case, the assistant can directly transport the aid to the location of the patient after he has arrived at the location of the aid.

$$S_w \geq C_u + \Delta_{uvk} + \Delta_{vwk} \tag{4.3}$$

On the other hand, if $C_u + \Delta_{uvk} < C_v$ holds for operations u and v, i.e. if the assistant arrives at the location of the aid before the corresponding operation v has been completed, inequality (4.4) has to hold for the starting time S_w of operation w. In this case, the assistant has to wait at the location of the aid until the aid becomes available before he can transport the aid to the location of the patient.

$$S_w \geq C_v + \Delta_{vwk} \tag{4.4}$$

In order to visualize blockings as well as resource transfers for the model introduced in this section, a small example based on the evacuation of two patients from a hospital is given in Example 4.3.

Example 4.3 We again consider the hospital from Example 4.1 displayed in Figure 4.1. Now, two patients have to be evacuated such that patient 1 has to be evacuated from room 5 to exit 2 with the help of one assistant as well as one wheelchair while patient 2 has to be evacuated from room 3 to exit 2 with the help of one assistant. The operations corresponding to the evacuation of these two patients as well as the processing times and required building sections of these operations are displayed in the activity-on-node network in Figure 4.4.

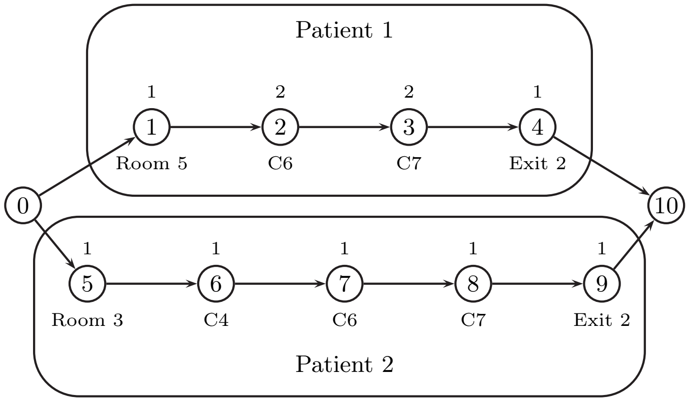


Figure 4.4: Activity-on-node network displaying the operations corresponding to the evacuation of the two patients as well as the two dummy operations 0 and 10 for the scenario from Example 4.3. The processing times of the operations are given above the nodes while the required building sections are given below.

In order to evacuate these patients, two assistants as well as one wheelchair are available at exit 1. Additionally, all building sections $k \in \mathcal{R}^{\text{sect}}$ (i.e. all rooms, exits, and corridor sections) have a capacity of $R_k = 1$ such that only one patient can be transported through a building section at any time t independent of the required assistants and aids. Next, the transfer times for all resources $k \in \mathcal{R}^{\text{trf}}$ are based on the shortest paths between the different locations used in this example such that one time unit is required in order to move through any building section (i.e. the transfer times are independent of the resources). The resulting transfer times $\Delta_{uvk} = \Delta_{uv}$ are given in Table 4.1. Finally, one assistant is required to transport the wheelchair between different locations.

	Room 3	Room 5	Exit 1	Exit 2
Room 3	0	3	4	4
Room 5	3	0	5	3
Exit 1	4	5	0	6
Exit 2	4	3	6	0

Table 4.1: Transfer times between the different locations of the hospital considered in Example 4.3.

Below, a feasible schedule for this example is displayed in Figure 4.5. In this schedule, two blockings occur between operations 6 and 7 (blocked by operation 2 on corridor section C6) as well as between operations 7 and 8 (blocked by operation 3 on corridor section C7). The makespan of this schedule is $C_{max} = 12$. Here, an assistant as well as the wheelchair can both be transferred from dummy source operation 0 to operation 1 for the evacuation of patient 1 while the second assistant can be transferred from dummy operation 0 to operation 5 for the evacuation of patient 2.

In order to visualize an example where assistant and wheelchair are not in the same location, we reduce the amount of available assistants to one assistant. An alternative schedule for this example is displayed in Figure 4.6. In this schedule, the assistant first evacuates patient 2 to exit 2 (represented by operation 9). Afterward, the assistant has to return to the location of the wheelchair at exit 1 (represented by operation 0) from where he can then transport the wheelchair to the location of patient 1 (represented by operation 1). Overall, this schedule has a makespan of $C_{max} = 26$. □

It should be noted that, in the model presented here, all assistants used for the transport of aids that are required for the evacuation of a patient are also themselves used for the evacuation of this patient. As generally only one unit of aid is required (e.g. one stretcher, one wheelchair, etc.), this is

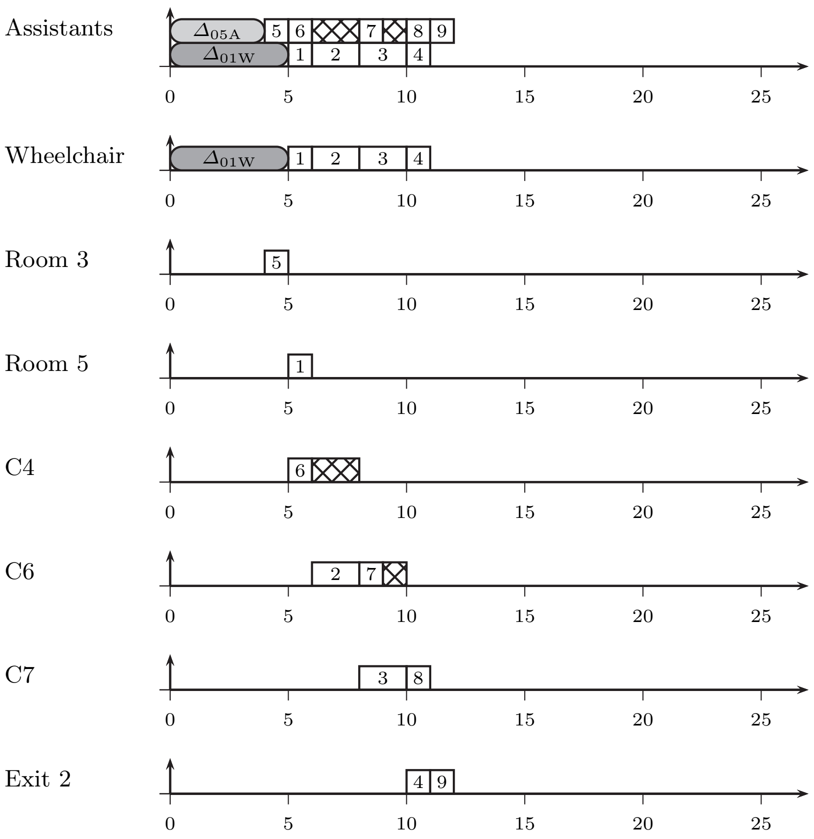


Figure 4.5: A feasible schedule for the problem of evacuating two patients from the small hospital displayed in Figure 4.1 with the help of two assistants as well as one wheelchair. Here, blockings occur between operations 6 and 7 as well as between operations 7 and 8 and are marked by crosshatched boxes. Resource transfers are identified by oval boxes such that direct transfers are marked in light gray while resource transports are marked in dark gray.

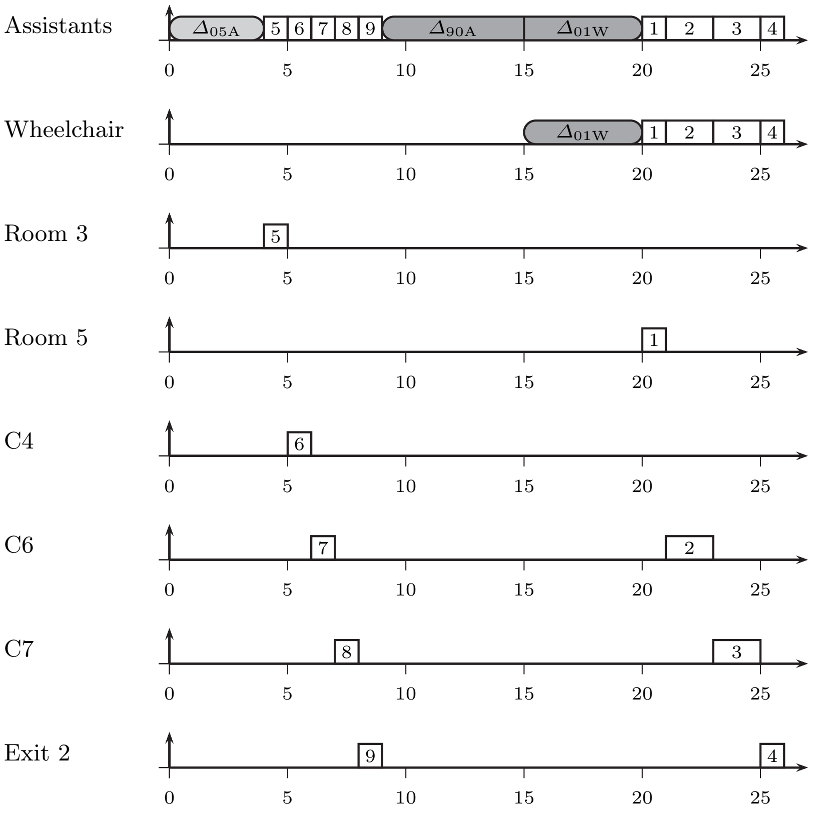


Figure 4.6: A schedule for the problem of evacuating two patients from the small hospital displayed in Figure 4.1 using one assistant as well as one wheelchair. Here, after patient 2 has been evacuated to exit 2, the assistant first has to be transferred to exit 1 where the wheelchair is located (represented by operation 0) and from there, the assistant transports the wheelchair to the initial location of patient 1 (represented by operation 1).

a feasible assumption for this problem. This assumption as well as other extensions of this model are further discussed in Section 4.1.3.

4.1.2 Mixed-Integer Linear Programming Formulation

In this section, a mixed-integer linear programming formulation for the MR-CPSP with blockings and resource transfers modeling the problem of hospital evacuations is described. For this formulation, the following integer and binary variables are required.

First of all, integer variables S_u and C_u denote the starting and finishing time of operation $u \in V_{\text{all}}$, respectively. In order to select the mode combination in which a job has to be processed, binary variables x_{jm} are introduced for all jobs $j \in J$ and mode combinations $m \in \mathcal{M}_j$ with

$$x_{jm} = \begin{cases} 1, \text{ if job } j \text{ is executed in mode combination } m \\ 0, \text{ otherwise.} \end{cases}$$

Next, resource transfers of resources $k \in \mathcal{R}^{\text{trf}}$ have to be calculated for the first and the last operation of each job while resource transfers of resources $k \in \mathcal{R}^{\text{sect}}$ have to be calculated between arbitrary operations of different jobs that require the same building section. Here, integer variables f^{trf}_{uvk} equal the amount of resource units of resource $k \in \mathcal{R}^{\text{trf}}$ transferred from operation $u \in V'_{\text{out}}$ to operation $v \in V'_{\text{in}}$ while integer variables f^{sect}_{uvk} equal the amount of resource units of resource $k \in \mathcal{R}^{\text{sect}}$ transferred from operation $u \in V_0$ to operation $v \in V_*$. Similarly, binary variables $\alpha^{\text{trf}}_{uvk}$ for all $u \in V'_{\text{out}}$, $v \in V'_{\text{in}}$, and $k \in \mathcal{R}^{\text{trf}}$ with

$$\alpha^{\text{trf}}_{uvk} = \begin{cases} 1, \text{ if resource } k \text{ is transferred from operation } u \text{ to operation } v \\ 0, \text{ otherwise} \end{cases}$$

as well as binary variables $\alpha^{\text{sect}}_{uvk}$ for all $u \in V_0$, $v \in V_*$, and $k \in \mathcal{R}^{\text{sect}}$ with

$$\alpha^{\text{sect}}_{uvk} = \begin{cases} 1, \text{ if resource } k \text{ is transferred from operation } u \text{ to operation } v \\ 0, \text{ otherwise} \end{cases}$$

are introduced. Additionally, integer variables z_{uvwkl} denote the amount of resource units of resource $k \in \mathcal{R}^{\text{asst}}$ from operation $u \in V'_{\text{out}}$ that are used to support the transfer of resource $l \in \mathcal{R}^{\text{aid}}$ from operation $v \in V'_{\text{out}}$ to operation $w \in V_{\text{in}}$. As before, binary variables β_{uvwkl} are introduced for all $u,v \in V'_{\text{out}}$, $w \in V_{\text{in}}$, $k \in \mathcal{R}^{\text{asst}}$, and $l \in \mathcal{R}^{\text{aid}}$ with

$$\beta_{uvwkl} = \begin{cases} 1, & \text{if resource } k \text{ from operation } u \text{ supports the transfer} \\ & \quad \text{of resource } l \text{ from operation } v \text{ to operation } w \\ 0, & \text{otherwise.} \end{cases}$$

For these latter variables, dummy sink operation $n+1$ does not have to be considered because $\Delta_{u,n+1,k} = 0$ is assumed to hold for the transfer times of all resources $k \in \mathcal{R}^{\text{sect}}$ from operations $u \in V'_{\text{out}}$ to dummy sink operation $n+1$. As a result of this, no actual transfers occur and no assistants are required to transport the aids to this dummy operation.

Now, the mixed-integer linear programming formulation itself is introduced. It should be noted that all parameters used in this formulation (i.e. processing times p_{um}, resource requirements r_{umk}, resource capacities R_k, transport requirements μ_{kl}, as well as transfer times Δ_{uvk}) are assumed to be integer. As before, the objective is to minimize the makespan (4.5) as denoted by the completion time C_{n+1} of dummy operation $n+1$, i.e. the hospital should be evacuated as quickly as possible.

$$\min \quad C_{n+1} \tag{4.5}$$

Next, the various constraints for the MRCPSP model are introduced. Here, first of all, equations (4.6) ensure that exactly one mode combination $m \in \mathcal{M}_j$ is chosen for each job $j \in J$ in which the job will be executed.

$$\sum_{m \in \mathcal{M}_j} x_{jm} = 1 \quad (j \in J) \tag{4.6}$$

Below, constraints (4.7) ensure that each operation $u \in V_{\text{all}}$ is processed for at least its processing time p_{um} in the selected mode combination $m \in \mathcal{M}_{\sigma(u)}$ of the corresponding job $\sigma(u)$. The actual completion time of the operation might be later if a blocking occurs.

$$S_u + \sum_{m \in \mathcal{M}_{\sigma(u)}} p_{um} \cdot x_{\sigma(u),m} \leq C_u \quad (u \in V_{\text{all}}) \tag{4.7}$$

In order to model the precedence constraints $u \to v$ between two successive operations $u, v \in V_{\text{all}}$, inequalities (4.8) as well as equalities (4.9) are introduced. Here, inequalities (4.8) model the precedence constraints $(u,v) \in \mathcal{NB}$ for which no blocking constraints have to be observed, i.e. operation u ends immediately after it has been processed for its processing time p_{um}. Similarly, equations (4.9) model the precedence constraints $(u,v) \in \mathcal{B}$ for which blocking constraints have to be observed, i.e. operation u can only end at the starting time S_v of operation v.

$$C_u - S_v \leq 0 \quad ((u,v) \in \mathcal{NB}) \tag{4.8}$$

$$C_u - S_v = 0 \quad ((u,v) \in \mathcal{B}) \tag{4.9}$$

Next, the constraints related to resource transfers are introduced. First of all, equations (4.10) model incoming resource transfers of resources $k \in \mathcal{R}^{\text{trf}}$ between jobs while equations (4.11) model outgoing resource transfers of resources $k \in \mathcal{R}^{\text{trf}}$ between jobs. Here, the amount of incoming resource units of resources $k \in \mathcal{R}^{\text{trf}}$ to the first operation $v \in V'_{\text{in}}$ of the selected evacuation route has to be equal to the resource requirements r_{vmk} of the operation if the corresponding job $\sigma(v)$ is executed in mode combination $m \in \mathcal{M}_{\sigma(v)}$. Similarly, the amount of outgoing resource units of resource $k \in \mathcal{R}^{\text{trf}}$ from the last operation $u \in V'_{\text{out}}$ of the selected evacuation route also has to be equal to the resource requirements r_{umk} of the operation if the corresponding job $\sigma(u)$ is executed in mode combination $m \in \mathcal{M}_{\sigma(u)}$.

$$\sum_{u \in V'_{\text{out}}} f^{\text{trf}}_{uvk} = \sum_{m \in \mathcal{M}_{\sigma(v)}} r_{vmk} \cdot x_{\sigma(v),m} \quad (v \in V'_{\text{in}};\ k \in \mathcal{R}^{\text{trf}}) \tag{4.10}$$

$$\sum_{v \in V'_{\text{in}}} f^{\text{trf}}_{uvk} = \sum_{m \in \mathcal{M}_{\sigma(u)}} r_{umk} \cdot x_{\sigma(u),m} \quad (u \in V'_{\text{out}};\ k \in \mathcal{R}^{\text{trf}}) \tag{4.11}$$

Below, the transfer of building sections is modeled by equations (4.12) and (4.13). Unlike assistants and aids, building sections are transferred between individual operations of different jobs instead of between the jobs themselves. As before, the amount of incoming and outgoing resource units to and from an operation has to be equal to the amount of resource units required by the operation.

$$\sum_{u \in V_0} f^{\text{sect}}_{uvk} = \sum_{m \in \mathcal{M}_{\sigma(v)}} r_{vmk} \cdot x_{\sigma(v),m} \quad (v \in V_*;\ k \in \mathcal{R}^{\text{sect}}) \tag{4.12}$$

$$\sum_{v \in V_*} f^{\text{sect}}_{uvk} = \sum_{m \in \mathcal{M}_{\sigma(u)}} r_{umk} \cdot x_{\sigma(u),m} \quad (u \in V_0;\ k \in \mathcal{R}^{\text{sect}}) \tag{4.13}$$

Now, constraints (4.14) and (4.15) are introduced in order to calculate which assistants are used to transport the required aids to the initial location from which the patient will be evacuated. Here, equations (4.14) ensure that an amount of μ_{kl} units of resource $k \in \mathcal{R}^{\text{asst}}$ from operations $u \in V'_{\text{out}}$ are assigned to transport one unit of resource $l \in \mathcal{R}^{\text{aid}}$ from operation $v \in V'_{\text{out}}$ to operation $w \in V_{\text{in}}$. Additionally, inequalities (4.15) ensure that the amount of resource units of resource $k \in \mathcal{R}^{\text{asst}}$ from operation $u \in V'_{\text{out}}$ used to transport resource units of resources $l \in \mathcal{R}^{\text{aid}}$ from operations $v \in V'_{\text{out}}$ to operation $w \in V_{\text{in}}$ may not exceed the amount of resource units of resource k that are actually transferred from operation u to operation w.

$$\sum_{u \in V'_{\text{out}}} z_{uvwkl} = \mu_{kl} \cdot f^{\text{trf}}_{vwl} \qquad \begin{pmatrix} v \in V'_{\text{out}};\ w \in V_{\text{in}}; \\ k \in \mathcal{R}^{\text{asst}};\ l \in \mathcal{R}^{\text{aid}} \end{pmatrix} \tag{4.14}$$

$$\sum_{v \in V'_{\text{out}}} \sum_{l \in \mathcal{R}^{\text{aid}}} z_{uvwkl} \leq f^{\text{trf}}_{uwk} \qquad \begin{pmatrix} u \in V'_{\text{out}};\ w \in V_{\text{in}}; \\ k \in \mathcal{R}^{\text{asst}} \end{pmatrix} \tag{4.15}$$

In the following, the binary variables $\alpha^{\text{trf}}_{uvk}$, $\alpha^{\text{sect}}_{uvk}$, and β_{uvwkl} are calculated based on the selected resource transfers. Here, inequalities (4.16) and (4.17) ensure that $\alpha^{\text{trf}}_{uvk} = 1$ holds if and only if resource $k \in \mathcal{R}^{\text{trf}}$ is transferred from operation $u \in V'_{\text{out}}$ to operation $v \in V'_{\text{in}}$.

$$f^{\text{trf}}_{uvk} \leq R_k \cdot \alpha^{\text{trf}}_{uvk} \quad (u \in V'_{\text{out}};\ v \in V'_{\text{in}};\ k \in \mathcal{R}^{\text{trf}}) \tag{4.16}$$

$$\alpha^{\text{trf}}_{uvk} \leq f^{\text{trf}}_{uvk} \quad (u \in V'_{\text{out}};\ v \in V'_{\text{in}};\ k \in \mathcal{R}^{\text{trf}}) \tag{4.17}$$

Next, inequalities (4.18) and (4.19) ensure that $\alpha^{\text{sect}}_{uvk} = 1$ holds if and only if resource $l \in \mathcal{R}^{\text{sect}}$ is transferred from operation $u \in V_0$ to operation $v \in V_*$.

$$f^{\text{sect}}_{uvk} \leq R_k \cdot \alpha^{\text{sect}}_{uvk} \quad (u \in V_0;\ v \in V_*;\ k \in \mathcal{R}^{\text{sect}}) \tag{4.18}$$

$$\alpha^{\text{sect}}_{uvk} \leq f^{\text{sect}}_{uvk} \quad (u \in V_0;\ v \in V_*;\ k \in \mathcal{R}^{\text{sect}}) \tag{4.19}$$

Finally, inequalities (4.20) and (4.21) ensure that $\beta_{uvwkl} = 1$ holds if and only if resource $k \in \mathcal{R}^{\text{asst}}$ from operation $u \in V'_{\text{out}}$ is used to transport resource $l \in \mathcal{R}^{\text{aid}}$ from operation $v \in V'_{\text{out}}$ to operation $w \in V_{\text{in}}$.

$$z_{uvwkl} \leq \max\{R_k, R_l\} \cdot \beta_{uvwkl} \qquad \begin{pmatrix} u,v \in V'_{\text{out}};\ w \in V_{\text{in}}; \\ k \in \mathcal{R}^{\text{asst}};\ l \in \mathcal{R}^{\text{aid}} \end{pmatrix} \tag{4.20}$$

$$\beta_{uvwkl} \leq z_{uvwkl} \qquad \begin{pmatrix} u,v \in V'_{\text{out}};\ w \in V_{\text{in}}; \\ k \in \mathcal{R}^{\text{asst}};\ l \in \mathcal{R}^{\text{aid}} \end{pmatrix} \tag{4.21}$$

After all constraints related to resource transfers have been introduced above, the starting times S_w of operations $w \in V'_{\text{in}}$ are now calculated by inequalities (4.22) and (4.23). Here, on the one hand, inequalities (4.22) calculate the earliest time at which resource $k \in \mathcal{R}^{\text{trf}}$ from operation $u \in V'_{\text{out}}$ can arrive at operation $w \in V'_{\text{in}}$ if it is transferred directly. On the other hand, inequalities (4.23), calculate the earliest time at which resource $k \in \mathcal{R}^{\text{asst}}$ from operation $u \in V'_{\text{out}}$ can arrive at operation $w \in V_{\text{in}}$ if resource k is used to transport resource $l \in \mathcal{R}^{\text{aid}}$ from operation $v \in V'_{\text{out}}$ to operation w.

$$S_w \geq C_u + \varDelta_{uwk} - M \cdot (1 - \alpha^{\text{trf}}_{uvk}) \qquad \begin{pmatrix} u \in V'_{\text{out}}; \\ w \in V'_{\text{in}}; \\ k \in \mathcal{R}^{\text{trf}} \end{pmatrix} \tag{4.22}$$

$$S_w \geq C_u + \varDelta_{uvk} + \varDelta_{vwl} - M \cdot (1 - \beta_{uvwkl}) \qquad \begin{pmatrix} u,v \in V'_{\text{out}}; \\ w \in V_{\text{in}}; \\ k \in \mathcal{R}^{\text{asst}}; \\ l \in \mathcal{R}^{\text{aid}} \end{pmatrix} \tag{4.23}$$

Here, M is a big integer value that should be chosen such that M is an upper bound on the time required to process all operations. Also, it should be noted that inequalities (4.22) represent the two scenarios (4.2) and (4.4) while inequalities (4.23) represent scenario (4.3). Finally, inequalities (4.24) link the starting times S_w of operations $w \in V_*$ with the resource transfers of resources $k \in \mathcal{R}^{\text{sect}}$.

$$S_w \geq C_u - M \cdot (1 - \alpha^{\text{sect}}_{uwk}) \quad (u \in V_0;\, w \in V_*;\, k \in \mathcal{R}^{\text{sect}}) \tag{4.24}$$

Lastly, constraints (4.25) through (4.33) given below define the domains of the variables used in this mixed-integer linear programming formulation.

$$S_u \in \mathbb{N} \qquad (u \in V_{\text{all}}) \tag{4.25}$$

$$C_u \in \mathbb{N} \qquad (u \in V_{\text{all}}) \tag{4.26}$$

$$x_{jm} \in \{0,1\} \qquad (j \in J;\, m \in \mathcal{M}_j) \tag{4.27}$$

$$f^{\text{sect}}_{uvk} \in \mathbb{N} \qquad (u \in V_0;\, v \in V_*;\, k \in \mathcal{R}^{\text{sect}}) \tag{4.28}$$

$$\alpha^{\text{sect}}_{uvk} \in \{0,1\} \qquad (u \in V_0;\, v \in V_*;\, k \in \mathcal{R}^{\text{sect}}) \tag{4.29}$$

$$f^{\text{trf}}_{uvk} \in \mathbb{N} \qquad (u \in V'_{\text{out}};\, v \in V'_{\text{in}};\, k \in \mathcal{R}^{\text{trf}}) \tag{4.30}$$

$$\alpha^{\text{trf}}_{uvk} \in \{0,1\} \qquad (u \in V'_{\text{out}};\, v \in V'_{\text{in}};\, k \in \mathcal{R}^{\text{trf}}) \tag{4.31}$$

$$z_{uvwkl} \in \mathbb{N} \qquad (u \in V'_{\text{out}};\, v \in V'_{\text{out}};\, w \in V_{\text{in}};\, k \in \mathcal{R}^{\text{asst}};\, l \in \mathcal{R}^{\text{aid}}) \tag{4.32}$$

$$\beta_{uvwkl} \in \{0,1\} \qquad (u \in V'_{\text{out}};\, v \in V'_{\text{out}};\, w \in V_{\text{in}};\, k \in \mathcal{R}^{\text{asst}};\, l \in \mathcal{R}^{\text{aid}}) \tag{4.33}$$

4.1.3 Further Considerations

Apart from the constraints introduced above, various extensions of the problem might also be of interest. Here, first of all, the number of assistants required to transport aids to the location of the patient might be larger than the number of assistants required to actually evacuate the patient. In this case, additional assistants have to be assigned to transport these aids. This more general situation is considered in Chapter 5 in which the resource-constrained project scheduling problem with first- and second-tier resource transfers is tackled. On the other hand, it is also possible to include aids as well as assistants as composite resources such that specific assistants are assigned to specific aids for the duration of the complete evacuation. This situation can be represented with the model introduced above and renders it unnecessary to assign assistants to the transport of aids during the scheduling at the cost of flexibility.

Next, resources with time-dependent resource profiles instead of renewable resources can be considered. These can be used, for example, if assistants arrive over time to help with the evacuation or if some assistants are not available for the complete planning horizon. Such time-dependent resource profiles can, for example, be realized by adding additional dummy operations that are scheduled at specific times corresponding to the jump points of the resource profiles. These additional operations then either release additional resources (source nodes) or collect resources (sink nodes) similar to the dummy operations 0 and $n + 1$.

Another possible extension is the introduction of release dates r_j for jobs $j \in J$ denoting the time at which a patient becomes available for evacuation. On the other hand, deadlines or due dates d_j for jobs $j \in J$ can be used if a patient has to arrive in a specific safety zone at a specific time, for example if the further transport to sheltering facilities has already been planned. Here, due dates can also be used in conjunction with alternative time-based objective functions, e.g. based on the earliness or tardiness of jobs.

Finally, another design decision concerns the evacuation routes for the patients. In the model presented above, evacuation routes have to be calculated in advance and are then selected by the scheduler by choosing the corresponding route mode for each patient. In this case, one possible approach is to calculate the shortest path from the location of each patient to each safety zone. The number of possible evacuation routes can then be reduced even further if each patient has to be evacuated to a specific safety zone. It should be noted that some patients can only be moved between different floors of the hospital by elevator (e.g. if they can only be evacuated in a hospital bed). This has to be taken into account when calculating evacuation routes in advance.

Alternatively, it might also be possible to enumerate all possible evacuation routes for each patient instead of only those based on shortest paths. Depending on the infrastructure of the hospital, however, this approach is likely to result in a large number of possible modes m_j for each job $j \in J$. An alternative might be to not select evacuation routes based on route modes but instead have the scheduler assemble the evacuation routes. For example, instead of chains of operations representing the evacuation routes, a tree of operations can be used representing all possible evacuation routes such that the root of the tree represents the initial location of the patient and each leaf corresponds to a safety zone at the end of one possible evacuation route. Then, starting with the root node, the scheduler has to decide which of the child nodes (corresponding to the alternative evacuation routes) is scheduled in each step.

4.2 A Tabu Search Algorithm

In this section, solution approaches for the problem of hospital evacuations are introduced. Here, because practical instances for the problem are likely to include a large number of jobs, these solution approaches are local-search algorithms that can compute feasible solutions for the problem in a short amount of time.

In the following, the MRCPSP model for the problem of hospital evacuations is decomposed into an assignment as well as a scheduling subproblem such that both subproblems can be solved separately. While the assignment subproblem consists of assigning an equipment mode $m_1 \in \mathcal{M}_{j1}$ as well as a route mode $m_2 \in \mathcal{M}_{j2}$ to each job $j \in J$, the scheduling subproblem consists of actually scheduling the operations $u \in V_j(m_2)$ corresponding to the selected route mode $m_2 \in \mathcal{M}_{j2}$ of each job $j \in J$. Now, the assignment subproblem is solved by a tabu search algorithm that selects and modifies the mode assignments for all jobs $j \in J$. By selecting a specific equipment mode $m_1 \in \mathcal{M}_{j1}$ as well as route mode $m_2 \in \mathcal{M}_{j2}$ for each job $j \in J$, the MRCPSP is reduced to a RCPSP (i.e. the scheduling subproblem), which is then solved by either a parallel or a serial schedule generation scheme.

Below, we first describe the tabu search algorithm that has been implemented in order to solve the assignment subproblem. An outline of this algorithm is presented in Algorithm 4.1. It should be noted that the general principle of tabu search has first been introduced by Glover (1989, 1990) and has since then been improved and applied to a variety of different optimization problems (cf. Glover and Laguna (1998)).

In line 1 of this algorithm, an initial solution $s \in S$ from the set of feasible solutions is generated. Here, a solution corresponds to a mode assignment of modes $m_1 \in \mathcal{M}_{j1}$ and $m_2 \in \mathcal{M}_{j2}$ to each job $j \in J$. In order to generate an initial solution, the tabu search algorithm selects an equipment mode m_1 as well as a route mode m_2 for each job j such that

$$\min_{m \in \mathcal{M}_j} \left\{ \sum_{u \in V_j(m_2)} p_{um} \right\}$$

holds for the mode combination $m = (m_1, m_2) \in \mathcal{M}_j$ of job j and the sum of the processing times of the operations corresponding to route mode m_2 is minimized. This is done by first selecting an equipment mode $m_1 \in \mathcal{M}_{j1}$ for each job $j \in J$ such that the speed with which the patient can be moved through the building sections based on the required assistants and aids is maximized. Then, in a second step, a route mode $m_2 \in \mathcal{M}_{j2}$ is selected for each job j such that the sum of the processing times as denoted by equation (4.1) is minimized for the operations corresponding to this route mode.

Based on this initial solution, a schedule is then generated by the selected schedule generation scheme and stored as the currently best solution s^* with the makespan *best* in line 2 of the algorithm. The actual search is then performed in lines 3 to 26 of the algorithm. Here, in each iteration, a schedule s' is chosen from the neighborhood $\mathcal{N}_{\text{mode}}$ in line 4 of the algorithm. This neighborhood $\mathcal{N}_{\text{mode}}$ contains all solutions for which either the equipment mode $m_1 \in \mathcal{M}_{j1}$ or the route mode $m_2 \in \mathcal{M}_{j2}$ of exactly one job $j \in J$ has been modified with respect to the current solution s, i.e. a modification in this neighborhood can either change the equipment mode m_1 or the route mode m_2 of a job j.

In order to select a solution in line 4 of the algorithm, possible neighbors $s' \in \mathcal{N}_{\text{mode}}(s)$ are evaluated by generating corresponding schedules using either the parallel or the serial schedule generation scheme until either a non-tabu solution s' with a better objective function value $c(s') < c(s)$ than the objective function value of the current solution s has been found (first-fit) or until no more neighbors are available. In the latter case, the best non-tabu solution s' that has been found is selected. If a solution $s' \in \mathcal{N}_{\text{mode}}(s)$ with $c(s') <$ *best* that is better than the currently best solution is generated, this solution is accepted immediately even if it is tabu (aspiration criterion).

Now, if a new solution s' has been selected, the tabu list TL is updated accordingly in line 12 of the algorithm and the current solution s is replaced by this solution s' in line 13. Additionally, if the solution is better than the currently best solution s^* (i.e. if $c(s') < best$ holds), the best solution s^* is replaced by solution s' in line 15.

Algorithm 4.1: Tabu Search

```
1  Generate an initial solution $s \in S$;
2  $best := c(s)$; $s^* := s$; $TL := \emptyset$; $S^* := \emptyset$;
3  repeat
4      Choose a solution $s'$ from the selected neighborhood that is not tabu or
       satisfies an aspiration criterion;
5      if no solution $s'$ could be generated then
6          if $S^* \neq \emptyset$ then
7              Restart with an elite solution $(s, TL) \in S^*$;
8          else
9              break;
10         end
11     else
12         Update the tabu list $TL$;
13         $s := s'$;
14         if $c(s') < best$ then
15             $best := c(s')$; $s^* := s'$; $TL := \emptyset$; $S^* := S^* \cup \{(s', TL)\}$;
16             if $|S^*| > l_{max}$ then
17                 Remove the oldest elite solution from $S^*$;
18             end
19         end
20     end
21     if a restart condition is satisfied then
22         if $S^* \neq \emptyset$ then
23             Restart with an elite solution $(s, TL) \in S^*$;
24         end
25     end
26 until a stopping condition is satisfied;
```

As stated above, tabu moves (i.e. modifications of the current solution s that might restore a previous solution and could result in cyclic behavior of the tabu search) are stored in a tabu list TL. For this algorithm, the tabu list stores both, the equipment mode $m_1 \in \mathcal{M}_{j1}$ as well as the route mode $m_2 \in \mathcal{M}_{j2}$ of the modified job $j \in J$ (i.e. the job for which either of the two modes has been changed) before the modification. Then, any move that would restore exactly this mode combination $m = (m_1, m_2)$ for this job j is tabu. The tabu list TL has a limited capacity η which is adapted dynamically as described by Brucker and Knust (2006) such that during an improving phase of the tabu search,the length of the tabu list is reduced according to

$$\eta := \max\{\eta - 1, TL_{min}\} \tag{4.34}$$

while, during a non-improving phase of the tabu search, the length of the tabu list is increased according to

$$\eta := \min\{\eta + 1, TL_{max}\}. \tag{4.35}$$

In these equations, TL_{min} and TL_{max} define the minimum and maximum capacity of the tabu list, respectively. Now, whenever the amount of elements within the tabu list TL exceeds the current capacity η, the oldest elements are removed from the tabu list. Additionally, if the currently best solution s^* could be improved during an iteration, all elements are removed from the tabu list in line 15 of the algorithm.

In order to intensify the search process, a total of l_{max} elite solutions can be stored during the search as described by Nowicki and Smutnicki (1996). An elite solution is a solution s' that improves the currently best solution s^* during an iteration. If such a solution is generated, both, the solution s' as well as the current tabu list TL are stored in the set S^* of elite solutions in line 15. It should be noted that the tabu list stored here is also updated to contain the move made in the next iteration in order to avoid making the same move if the tabu search is restarted from this elite solution. Now, if no solution $s' \in \mathcal{N}_{\text{mode}}(s)$ could be generated in line 4 or if the tabu search could not improve the currently best solution for a total of t_{max}, the tabu search restarts from the oldest elite solution $(s, TL) \in S^*$ in either line 7 or in line 23, respectively.

Finally, the tabu search terminates after a given stopping condition is satisfied. For example, this stopping condition can be a maximum number of iterations the tabu search has to perform or, alternatively, a maximum amount of time after which the tabu search is terminated. The best solution s^* found during the tabu search is then returned. It should be noted that the tabu search also terminates if no solution $s' \in \mathcal{N}_{\text{mode}}(s)$ could be generatedin line 4 and the set S^* of elite solutions is empty.

In the remainder of this section the two schedule generation schemes used to solve the scheduling subproblem are described in more detail. Here, the parallel schedule generation is described in Section 4.2.1 while the serial schedule generation is described in Section 4.2.2. Finally, Section 4.2.3 highlights some shortcomings of these solution approaches. In particular, it is shown that the solution space considered by these algorithms does not necessarily contain an optimal solution.

In order to simplify the description of the schedule generation schemes, the notation for the MRCPSP model introduced in the previous section is adapted. In particular, as both modes $m_1 \in \mathcal{M}_{j1}$ and $m_2 \in \mathcal{M}_{j2}$ are fixed for all jobs $j \in J$, any data related to other modes can be omitted. Thus, for each job j, only the set $V_j = V_j(m_2)$ corresponding to the selected evacuation route has to be considered. Similarly, the sets V_{all}, V_0, V_*, V_{in}, V_{out}, V'_{in}, and V'_{out} only contain operations that actually have to be scheduled while the sets $\mathcal{NB}$ and $\mathcal{B}$ only contain precedence constraints $u \to v$ between these operations. Finally, only the processing time $p_u = p_{um}$ as well as the resource requirements $r_{uk} = r_{umk}$ for all resources $k \in \mathcal{R}$ corresponding to the selected mode combination $m = (m_1, m_2) \in \mathcal{M}_{\sigma(u)}$ of job $\sigma(u)$ have to be considered for each operation $u \in V_{\text{all}}$.

4.2.1 Parallel Schedule Generation Scheme

In this section, the parallel schedule generation scheme (parallel SGS) is described that has been implemented to solve the scheduling subproblem of the HEP. This algorithm is an adaption of the parallel schedule generation scheme for the resource-constrained multi-project scheduling problem with transfer times as described by Krüger and Scholl (2009) where resource transfers are selected based on a priority rule. The parallel schedule generation scheme is outlined in Algorithm 4.2.

As described in Section 3.2.3, the general idea of the parallel schedule generation scheme is to schedule operations from a list of eligible operations E_λ in each iteration λ at gradually increasing time points t_λ until either the list E_λ is empty or no more operations from the list can be feasibly scheduled. For this algorithm, the list E_λ contains all operations that can be scheduled to start at time t_λ with respect to precedence constraints as well as transfer times while the set A_λ contains all operations $u \in V_{\text{all}}$ that are active at time t_λ. It should be noted that resource-feasibility is not guaranteed for the operations contained in the list E_λ. In particular, at the time when the list

E_λ is calculated, it is possible that not a sufficient amount of space is available in the required building section in order to start an operation $u \in E_\lambda$ at time t_λ. The reason for this is related to blockings and is discussed in more detail later in this section.

Algorithm 4.2: Parallel Schedule Generation Scheme

```
λ := 1;
Calculate an initial time point $t_1$;
Initialize the set $A_1 := \emptyset$ of active operations at time $t_1$;
Calculate the list $E_1$ of operations that can start at time $t_1$;
while not all operations have been scheduled do
    Sort the list $E_\lambda$ according to a priority rule;
    repeat
        foreach $w \in E_\lambda$ do
            if $w \in V'_{in}$ then
                CalculateResourceTransfers($w, t_\lambda$)
            end
            if $w$ can be scheduled resource-feasibly then
                Schedule operation $w$ and update resources;
                Set starting time $S_w := t_\lambda$;
                $A_\lambda := A_\lambda \cup \{w\}$; $E_\lambda := E_\lambda \setminus \{w\}$;
                if $w$ has a predecessor operation $u \in V_{\sigma(w)}$ then
                    Set completion time $C_u := t_\lambda$;
                    $A_\lambda := A_\lambda \setminus \{u\}$;
                end
            end
        end
    until no more operations could be scheduled;
    $\lambda := \lambda + 1$;
    Calculate the next time point $t_\lambda$;
    Calculate the new list $E_\lambda$ and the new set $A_\lambda$;
end
```

Now, in line 2 of the algorithm, an initial time point $t_1 \geq 0$ with

$$t_1 = \min_{u \in V'_{\text{in}}} \left\{ \max_{k \in \mathcal{R}^{\text{trf}} \text{ with } r_{uk} > 0} \Delta_{0uk} \right\}$$

is calculated as the earliest time at which an operation $u \in V'_{\text{in}}$ can start.

For this, the first operation $u \in V'_{\text{in}}$ of each job $j \in J$ is regarded and the transfer times Δ_{0uk} between dummy source operation 0 and the initial location of the patient for all resources $k \in \mathcal{R}^{\text{trf}}$ required for the evacuation of the patient are compared. It should be noted that resource transports of aids do not have to be considered separately because assistants and aids are assumed to start from the same initial location (as denoted by dummy source operation 0), i.e. inequality (4.4) applies to all resources $k \in \mathcal{R}^{\text{trf}}$ required by an operation $u \in V'_{\text{in}}$.

Afterward, the initial set of active operations A_1 is initialized in line 3 and the initial list of eligible operations E_1 is calculated in line 4 of the algorithm. For this, all operations $u \in V_{\text{in}}$ (i.e. all precedence-feasible operations) that can be scheduled to start at time t_1 (i.e. all transfer-feasible operations for which $\Delta_{0uk} \leq t_1$ holds for all required resources $k \in \mathcal{R}^{\text{trf}}$) are included in the list E_1.

Next, as well as in each subsequent iteration λ of the parallel schedule generation scheme, all operations in the list E_λ are sorted according to a chosen priority rule. For this algorithm, the priority rules listed in Table 4.2 have been implemented. Here, the priority rules SPT and LPT are based on the processing times of the operations and sort the operations according to increasing or decreasing processing times, respectively. Next, the priority rules LST and LFT require the latest start and finish times LS_u and LF_u of operations $u \in V_{\text{all}}$ that can be calculated based on an upper bound UB on the time available to process all operations as described in Section 3.1.2. These priority rules then sort the operations $u \in E_\lambda$ according to increasing latest start times LS_u or latest finish times LF_u, respectively. Finally, the priority rule RAND randomly selects operations from the list E_λ.

Priority Rule	Description
SPT	Shortest Processing Time
LPT	Longest Processing Time
LST	Latest Start Time
LFT	Latest Finish Time
RAND	Random

Table 4.2: Priority rules used to sort the list of eligible operations E_λ.

After the list of eligible operations E_λ has been sorted according to the selected priority rule, the parallel schedule generation scheme tries to schedule operations $w \in E_\lambda$ from the list of eligible operations until either all operations have been scheduled or no more operations can be feasibly scheduled. In order to schedule an operation $w \in E_\lambda$, a sufficient amount r_{wk} of space has to be available in the required building section $k \in \mathcal{R}^{\text{sect}}$ at time point t_λ. Now, if operation w is not the first operation of the corresponding job $\sigma(w)$ (i.e. if $w \notin V'_{\text{in}}$ holds), the operation is immediately scheduled in the required building section at time t_λ in line 12 of the algorithm. Otherwise, if w is the first operation of the corresponding job $\sigma(w)$ (i.e. if $w \in V'_{\text{in}}$ holds), resource transfers of assistants and aids to this operation are calculated in line 10 of the algorithm. In this case, a sufficient amount of assistants and aids has to be selected such that the operation can transfer-feasibly be scheduled to start at time point t_λ. These resource transfers are calculated by the procedure CalculateResourceTransfers outlined below.

First of all, in line 2 of this procedure, a set $\mathcal{F}_k^{\text{asst}}$ is calculated for each resource $k \in \mathcal{R}^{\text{asst}}$ required by operation $w \in V'_{\text{in}}$, i.e. for all resources k for which $r_{wk} > 0$ holds. These sets $\mathcal{F}_k^{\text{asst}}$ contain all possible feasible resource transfers of resource k from already completed operations $u \in V'_{\text{out}}$ (i.e. from operations with completion times $C_u \leq t_\lambda$) to operation w. Here, a resource transfer of resource k from operation u to operation w is feasible if inequality (4.2) holds between the completion time C_u of operation u and the current decision point t_λ, i.e. if $C_u + \Delta_{uwk} \leq t_\lambda$ holds.

Afterward, if and only if a sufficient amount of each required resource $k \in \mathcal{R}^{\text{asst}}$ can be transferred to operation w until time t_λ, all resources $l \in \mathcal{R}^{\text{aid}}$ required by operation w are considered in lines 5 to 18. For this, all possible feasible resource transfers of resource $l \in \mathcal{R}^{\text{aid}}$ from completed operations $v \in V'_{\text{out}}$ to operation w are calculated in line 6 and stored in sets $\mathcal{F}_l^{\text{aid}}$. As described above, in order to transfer one unit of resource l from operation v to operation w, an amount of μ_{kl} units of resources $k \in \mathcal{R}^{\text{asst}}$ are required to transport the resource. For this reason, a resource transfer of one unit of resource l from operation v to operation w is called feasible if a sufficient amount of supporting resource units of resources $k \in \mathcal{R}^{\text{asst}}$ from completed operations $u \in V'_{\text{out}}$ with

$$\max\{C_u + \Delta_{uvk} + \Delta_{vwl}, C_v + \Delta_{vwk}\} \leq t_\lambda \tag{4.36}$$

are available to satisfy all transport requirements μ_{kl}. Here, inequality (4.36) corresponds to inequalities (4.3) and (4.4) introduced above. When calculating all possible resource transfers of resources $l \in \mathcal{R}^{\text{aid}}$, it is sufficient to only regard resource transfers of resources $k \in \mathcal{R}^{\text{aid}}$ contained in the set $\mathcal{F}_k^{\text{asst}}$

Procedure CalculateResourceTransfers(w, t_λ)

```
for $k \in \mathcal{R}^{asst}$ with $r_{wk} > 0$ do
    Calculate the set $\mathcal{F}_k^{\text{asst}}$ of all possible feasible resource transfers of
    resource $k$ from completed operations $u \in V'_{\text{out}}$ that can arrive at
    operation $w$ until time $t_\lambda$;
end
if sufficient resources $k \in \mathcal{R}^{asst}$ can be transferred to operation $w$ then
    for $l \in \mathcal{R}^{aid}$ with $r_{wl} > 0$ do
        Calculate the set $\mathcal{F}_l^{\text{aid}}$ of all possible feasible resource transfers of
        resource $l$ from completed operations $v \in V'_{\text{out}}$ that can be
        transported by resources $k \in \mathcal{R}^{\text{asst}}$ from the set $\mathcal{F}_k^{\text{asst}}$ and arrive at
        operation $w$ until time $t_\lambda$;
        if sufficient resources $l$ can be transported to operation $w$ then
            while not sufficient resource transfers have been chosen do
                if no feasible resource transfer for resource $l$ exists then
                    Operation $w$ can not be scheduled transfer-feasibly;
                end
                Choose a feasible resource transfer of resource $l$ from the set
                $\mathcal{F}_l^{\text{aid}}$ as well as sufficient feasible transporting resources
                $k \in \mathcal{R}^{\text{asst}}$ from the sets $\mathcal{F}_k^{\text{asst}}$;
                Update the sets $\mathcal{F}_k^{\text{asst}}$ and $\mathcal{F}_l^{\text{aid}}$;
            end
        else
            Operation $w$ can not be scheduled transfer-feasibly;
        end
    end
    for $k \in \mathcal{R}^{asst}$ with $r_{wk} > 0$ do
        if not sufficient resources $k$ are transferred to operation $w$ then
            Choose sufficient feasible resource transfers of resource $k$ from
            the set $\mathcal{F}_k^{\text{asst}}$;
        end
    end
else
    Operation $w$ can not be scheduled transfer-feasibly;
end
```

because only the resource unit corresponding to these resource transfers can be transferred to operation w until time point t_λ.

Now, if and only if a sufficient amount of resource units of resource l can be transported to operation w, the actual resource transfers are selected in lines 8 to 14 of the procedure. For this, resource transfers from the set $\mathcal{F}_l^{\text{aid}}$ consisting of one unit of resource l as well as a sufficient amount of transporting resources $k \in \mathcal{R}^{\text{asst}}$ from the corresponding sets $\mathcal{F}_k^{\text{asst}}$ are chosen until a sufficient amount of resource units of resource l have been selected or until no more feasible resource transfers can be selected. In order to choose these resource transfers for both, resource l as well as the required transporting resources $k \in \mathcal{R}^{\text{asst}}$, a priority rule ES (earliest start) has been implemented that selects resource transfers according to non-decreasing arrival times of the resources at operation w. In the following, this priority rule is referred to as a transfer rule in order to distinguish it from the priority rules used to select operations. It should be noted that Krüger and Scholl (2009) introduce additional transfer rules that can be used to select resource transfers. These other transfer rules are not used in this algorithm due to the problem discussed in Section 4.2.3.

After both, a resource transfer of resource l as well as a sufficient amount of transporting resource units of resources $k \in \mathcal{R}^{\text{asst}}$ have been selected in line 12, the sets $\mathcal{F}_k^{\text{asst}}$ and $\mathcal{F}_l^{\text{aid}}$ are updated, i.e. the selected resource transfers are removed from these sets. It should be noted that removing resource transfers $k \in \mathcal{R}^{\text{asst}}$ from the sets $\mathcal{F}_k^{\text{asst}}$ might also influence the sets $\mathcal{F}_l^{\text{aid}}$, e.g. if a resource l can no longer be transported from a completed operation $v \in V'_{\text{out}}$ to operation w until time point t_λ by the remaining resource transfers from the sets $\mathcal{F}_k^{\text{asst}}$. If this is the case and not a sufficient amount of resource units of resource l can be transported to operation w, this operation can not be scheduled transfer-feasibly and the procedure terminates without a solution in line 10. Similarly, the operation can not be scheduled and the procedure terminates without a solution if not a sufficient amount of feasible resource transfers of resources $k \in \mathcal{R}^{\text{asst}}$ or $l \in \mathcal{R}^{\text{aid}}$ is available in lines 2 or 6, respectively.

Finally, after a sufficient amount of resource transfers of resources $l \in \mathcal{R}^{\text{aid}}$ as well as the corresponding transporting resources $k \in \mathcal{R}^{\text{asst}}$ have been selected, the remaining resources transfers of resources $k \in \mathcal{R}^{\text{asst}}$ are selected in lines 19 to 23. These resources transfers are again chosen from the sets $\mathcal{F}_k^{\text{asst}}$ based on the transfer rule ES until a sufficient amount of resource units of resources k from completed operations $u \in V'_{\text{in}}$ have been selected to be transferred to operation w. It should be noted that resource units of resources $l \in \mathcal{R}^{\text{aid}}$ from operation $u \in V'_{\text{out}}$ only have to be transported to operation w if $\Delta_{uwl} > 0$ holds (i.e. if an actual transfer takes place).

Otherwise, no supporting resources $k \in \mathcal{R}^{\text{asst}}$ have to be selected for the corresponding resource transfers. For the problem of hospital evacuations, this is the case for all resource transfers to dummy sink operation $n+1$.

Now, if a sufficient amount of resource transfers to operation w could be calculated by procedure CalculateResourceTransfers in line 10 of the parallel schedule generation scheme outlined in Algorithm 4.2 and if a sufficient amount of space is available in the required building section $k \in \mathcal{R}^{\text{sect}}$, the operation can actually be scheduled in line 13 of the algorithm. In this case, all selected assistants and aids are transferred to operation w and the resource profile of the building section is updated at time t_λ.

Regardless of whether operation w is the first operation of the corresponding evacuation route or not, the starting time of operation w is set to $S_w = t_\lambda$ after it has been scheduled and the sets A_λ as well as E_λ are updated in lines 14 and 15. Additionally, if operation w has a predecessor operation $u \in V_{\sigma(w)}$ on the selected evacuation route, the completion time of this operation u is set to $C_u = t_\lambda$ and operation u is removed from the set A_λ in lines 17 and 18. As a result of this, the building section used by operation u becomes available again at time t_λ and can be used by other operations.

These steps outlined in lines 8 to 21 of the algorithm are repeated for all operations $w \in E_\lambda$. As pointed out above, it is not guaranteed for the operations contained in the list E_λ that a sufficient amount of space is available in the required building sections at time point t_λ when the list is calculated by the parallel schedule generation scheme. This is due to the blocking constraint that prevents an operation $u \in V_{\text{all}}$ from completing until its successor operation $w \in V_{\sigma(u)}$ has been started (cf. lines 16 to 19). For this reason, building sections might only become available again during an iteration λ as operations from the set E_λ are being scheduled. Thus, multiple passes over the list E_λ are performed in each iteration of the parallel schedule generation scheme such that in each pass, as many operations as possible are scheduled from the list E_λ. This is repeated until either the list E_λ is empty, i.e. all eligible operations have been scheduled, or until no more operations could be scheduled in the previous pass.

Then, after no more operations could be scheduled in the last pass of an iteration λ, the next time point $t_{\lambda+1}$, the new list of eligible operations $E_{\lambda+1}$, as well as the new set of active operations $A_{\lambda+1}$ are calculated in lines 23 to 25 for the next iteration $\lambda+1$. For this, in a first step, the next time point $t_{\lambda+1}$ is calculated based on the completion times of currently active operations as well as on the transfer times to unscheduled operations from the set V'_{in} as follows.

First of all, in order to calculate the next time point $t_{\lambda+1}$, all active operations $w \in A_\lambda$ with $S_w + p_w > t_\lambda$ are considered (i.e. all operations w that have not yet been active for their processing time p_w). It should be noted that the set A_λ might also contain operations w with $S_w + p_w \leq t_\lambda$ that have already been active for at least their processing time p_w, i.e. if the respective successor operations could not yet be started and the operations are blocked. Based on these operations, a temporary time point

$$t_{\text{temp}} = \min \left\{ UB, \min_{w \in A_\lambda \text{ with } S_w + p_w > t_\lambda} \{S_w + p_w\} \right\}$$

is calculated. Here, if no active operations $w \in A_\lambda$ with $S_w + p_w > t_\lambda$ exist, the temporary time point t_{temp} is set to an upper bound UB.

Next, all unscheduled operations $w \in V'_{\text{in}}$ are considered. For each of these operations w, the earliest time $t_{\text{ES},w}$ until which all required assistants and aids can be transferred from already completed operations $u \in V'_{\text{out}}$ to operation w is calculated based on the procedure CalculateResourceTransfers with respect to the transfer rule ES. It should be noted that the procedure is called with the input parameter $t_\lambda = UB$, i.e. all possible resource transfers from already completed operations $u \in V'_{\text{out}}$ are considered. Then, if a sufficient amount of resource units of resources $k \in \mathcal{R}^{\text{asst}}$ and $l \in \mathcal{R}^{\text{aid}}$ can be transferred to operation w, the earliest time $t_{\text{ES},w}$ until which all of these resource units can arrive at operation w with respect to the transfer rule ES is returned. Otherwise, if not a sufficient amount of resource units can be transferred to operation w, the value $t_{\text{ES},w} = UB$ is returned. Now, the next time point $t_{\lambda+1}$ is calculated as

$$t_{\lambda+1} = \max \left\{ t_\lambda + 1, \min \left\{ t_{\text{temp}}, \min_{w \in V'_{\text{in}},\ w \text{ is unscheduled}} t_{\text{ES},w} \right\} \right\}.$$

It should be noted that $t_{\lambda+1} \geq t_\lambda + 1$ has to hold for the next time point $t_{\lambda+1}$. Also, it is possible that the next time point is calculated as $t_{\lambda+1} = UB$ although not all operations have been scheduled. In this case, either the upper bound has been chosen too small or a deadlock occurred (cf. page 89) and the parallel schedule generation scheme terminates without a solution.

After the next time point $t_{\lambda+1}$ has been determined, the new list $E_{\lambda+1}$ of eligible operations as well as the new set $A_{\lambda+1}$ of active operations can be computed. In order to calculate the new set $A_{\lambda+1}$, all currently active operations $w \in A_\lambda$ are considered. Here, if $S_w + p_w = t_{\lambda+1}$ holds for operation w (i.e. if the operation has been active for its processing time p_w) and if operation w is the last operation of the corresponding evacuation route (i.e. if $w \in V'_{\text{out}}$ holds), the operation ends at the new time point $t_{\lambda+1}$ with the completion time $C_w = t_{\lambda+1}$. In this case, all resources (i.e. assistants, aids, as well as the building section required by operation w) become available again at this time point. All other operations $w \in A_\lambda$ with $w \notin V'_{\text{out}}$ are always added to the new set $A_{\lambda+1}$. These operations are only completed after they have been active for at least their processing times p_w and their respective successor operation has started.

Finally, in a last step, the new list $E_{\lambda+1}$ of eligible operations is calculated. For this, the successor operations of all active operations $w \in A_{\lambda+1}$ with $S_w + p_w \leq t_{\lambda+1}$ that have been active for at least their processing time p_w are added to the list $E_{\lambda+1}$. Additionally, all unscheduled operations $w \in V'_{\text{in}}$ to which all required resources $k \in \mathcal{R}^{\text{trf}}$ can be transferred until time point $t_{\lambda+1}$ are added based on the results from procedure CalculateResourceTransfers.

Afterward, the parallel schedule generation scheme continues with the next iteration $\lambda + 1$. This is repeated until either all operations have been scheduled successfully or until the next time point $t_{\lambda+1}$ is equal to the upper bound UB. In the latter case, no feasible schedule could be generated and the algorithm terminates without a solution.

Example 4.4 We again consider the hospital from Example 4.1 displayed in Figure 4.1. Now, $N = 3$ patients have to be evacuated such that patient 1 has to be evacuated from room 5 to exit 1 by one assistant, patient 2 has to be evacuated from room 3 to exit 2 by two assistants as well as one stretcher, and patient 3 has to be evacuated from room 3 to exit 1 by one assistant. The operations corresponding to the three jobs as well as the building sections required by the operations are displayed in the activity-on-node network in Figure 4.7.

For this example, all real operations $u \in V$ are assumed to have unit processing times $p_u = 1$ as well as resource requirements $r_{uk} = 1$ in the required building section $k \in \mathcal{R}^{\text{sect}}$. These building sections $k \in \mathcal{R}^{\text{sect}}$ are assumed to have a capacity of $R_k = 1$ such that only one patient can be evacuated through a building section at any time. Finally, two assistants as well as one stretcher (which has to be transported by one assistant) are available for the evacuation of the patients and are initially located at exit 2. The transfer times between the different locations are given in Table 4.1 and are the same for both, the assistants as well as the stretcher.

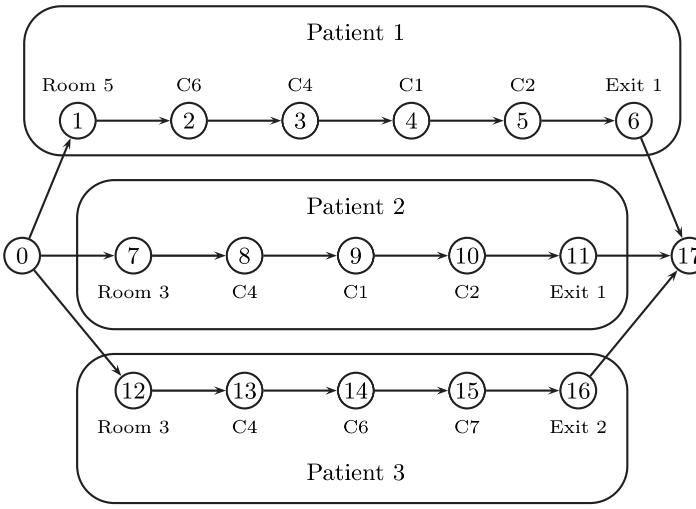


Figure 4.7: Activity-on-node network displaying the operations for the $N = 3$ jobs corresponding to patients 1, 2, and 3 as well as the two dummy operations 0 and 17. The building sections required by the operations are given next to the corresponding nodes.

In the following, this example is used to visualize some iterations of the parallel schedule generation scheme. For this, a schedule for this example generated by the parallel schedule generation scheme is displayed in Figure 4.8. Here, first of all, the initial time point t_1 is calculated based on the earliest time at which all required resources can be transferred to one of the three patients. In this case, one assistant can be transferred from exit 2 to room 5 (i.e. the initial location of patient 1) until time point $t_1 = 3$. The corresponding operation 1 is then scheduled to start at time $S_1 = t_1 = 3$.

Afterward, the next time point t_2 is calculated as the earliest time $t_2 > t_1$ at which an operation $u \in A_1$ has been active for its processing time p_u or the earliest time at which an operation $u \in V'_{\text{in}}$ can be scheduled transfer-feasibly. In this case, operation 1 has been active for its processing time p_1 at time $t = S_1 + p_1 = 4$. Also, operation 12 can be scheduled transfer-feasibly

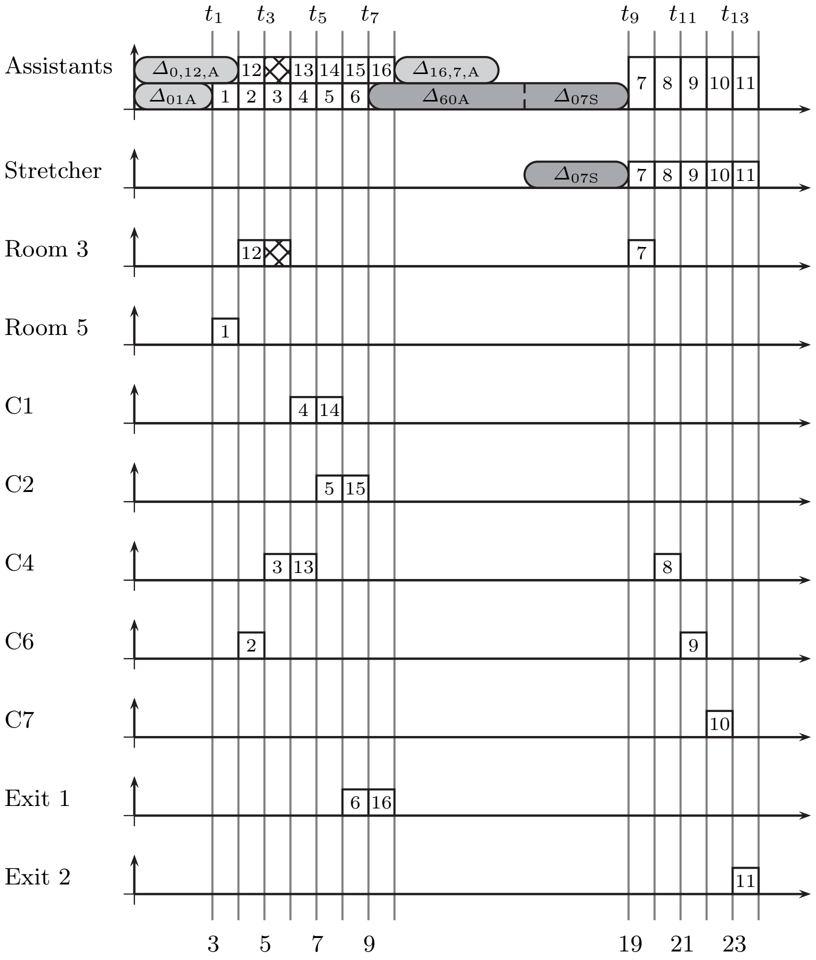


Figure 4.8: A schedule for Example 4.4 as it is generated by the parallel schedule generation scheme. The figure also displays the time points t_λ computed in iterations $\lambda = 1, \ldots, 13$ by the algorithm.

at this time, i.e. one assistant can be transferred from exit 2 to room 3 until time $t = 4$. Thus, the next time point $t_2 = 4$ is chosen and the set $E_2 = \{2, 12\}$ of eligible operations as well as the set $A_2 = \{1\}$ of active operations can be calculated for this time point. Now, in iteration $\lambda = 2$, operations 2 and 12 can both be scheduled to start at time $t_2 = 4$. Additionally, after operation 2 has been scheduled in this iteration, its predecessor operation 1 is assigned the completion time $C_1 = 4$ and is removed from the set of active operations.

Similarly, the next time point $t_3 = 5$ as well as the corresponding set $E_3 = \{3, 13\}$ of eligible operations can be calculated. Here, because both operations from this set require floor segment C4, only one of these operations can be scheduled to start at this time point. For example, we assume that operation 3 is scheduled to start at time point $t_3 = 5$ based on the selected priority rule (e.g. based on the priority rule RAND). In this case, operation 13 can not be scheduled. and its predecessor operation 12 remains in the set of active operations and blocks room 3. In the following iterations $\lambda = 4$ to $\lambda = 7$, the remaining operations corresponding to the evacuation of patients 1 and 2 are scheduled. It should be noted that in iterations 4, 5, and 6, up to two passes may be required per iteration in order to schedule both operations from the corresponding set of eligible operations. For example, we assume that the parallel schedule generation scheme first tries to schedule operation 13 in iteration 4. As building section C4 is still occupied by operation 3 at this time, the algorithm next tries to schedule operation 4 in building section C1. This succeeds and operation 4 is scheduled to start at time $t_4 = 6$ in building section C1. Additionally, the completion time of operation 3 is set to $C_6 = 6$ and building section C4 becomes available again at this time point. Now, in a second pass, the parallel schedule generation scheme again tries to schedule operation 13 in building section C4 at time point t_4 and succeeds as the building section is now available again.

Next, time point $t_8 = 10$ is calculated as the completion time of operation 16. Here, because operation 16 is the last operation of the evacuation route for patient 3, this operation can be completed immediately at time $C_{16} = 10$. At this time point t_8, however, the set of eligible operations is empty as operation 7 can not be scheduled transfer-feasibly. For this reason, the next time point t_9 is calculated as the earliest time at which all required resources can be transferred from already completed operations $\{0, 6, 16\} \in V'_{\text{out}}$ to operation 7. In this case, time point $t_9 = 19$ is calculated such that one assistant from operation 6 (corresponding to exit 2) is first transferred to operation 0 (corresponding to exit 1) and transports the stretcher from there to operation 7 (corresponding to room 3) while the second assistant is immediately transferred from operation 16 (corresponding to exit 2) to operation 7. The starting time of operation 7 is then set to $C_7 = t_9 = 19$. Finally, the remaining operations 8 to 11 can be scheduled in iterations 10 to 13 such that the makespan of this schedule is $C_{max} = 24$. $\square$

After the parallel schedule generation scheme has been described above, the remainder of this section is used to consider some properties of this algorithm. In particular, the time complexity of this algorithm is analyzed below, some properties of the schedules generated by the algorithm are considered on page 85, and the problem of deadlocks is discussed on page 89.

Time Complexity

The time complexity of the parallel schedule generation scheme is dominated by line 10 of Algorithm 4.2 in which the resource transfers to an operation $w \in V'_{\text{in}}$ until time point t_λ are calculated by Procedure CalculateResource-Transfers. In the following, we first analyze the time complexity of this procedure which is dominated by calculating the resource transfers of resources $l \in \mathcal{R}^{\text{aid}}$ from completed operations $v \in V'_{\text{out}}$ to operation w. Here, the time complexity of calculating all possible feasible resource transfers of a resource $l \in \mathcal{R}^{\text{aid}}$ in line 6 is bounded by $\mathcal{O}(N^2 \cdot |\mathcal{R}^{\text{asst}}|)$. This follows from the fact that each resource l can be transferred from at most N operations $v \in V'_{\text{out}}$ (including the dummy operation 0) to operation w and at most $|\mathcal{R}^{\text{asst}}|$ different resources $k \in \mathcal{R}^{\text{asst}}$ may be required to support the transfer from N different operations $u \in V'_{\text{out}}$.

Next, the actual resource transfers of resources l to operation w are chosen in lines 8 to 14. For this, a resource transfer of one unit of resource l from the set $\mathcal{F}_l^{\text{aid}}$ as well as a sufficient amount of transporting resources $k \in \mathcal{R}^{\text{asst}}$ from the sets $\mathcal{F}_k^{\text{asst}}$ are selected in line 12 of the procedure based on the transfer rule ES. Here, the earliest time at which one resource transfer from the set $\mathcal{F}_l^{\text{aid}}$ can arrive at operation w can be calculated in $\mathcal{O}(N^2 \cdot |\mathcal{R}^{\text{asst}}|^2)$ time. Thus, the time required to select one resource transfer in line 12 is bounded by $\mathcal{O}(N^3 \cdot |\mathcal{R}^{\text{asst}}|^2)$ because at most N resource transfers from the set $\mathcal{F}_l^{\text{aid}}$ have to be evaluated. As operation w requires at most R_l units of resource l, the time complexity of lines 8 to 14 is bounded by $\mathcal{O}(N^3 \cdot |\mathcal{R}^{\text{asst}}|^2 \cdot R_{max}^{\text{aid}})$ where R_{max}^{aid} is the maximal capacity of any resource $l \in \mathcal{R}^{\text{aid}}$.

As these steps have to be repeated for all required resources $l \in \mathcal{R}^{\text{aid}}$, the time complexity of procedure CalculateResourceTransfers is bounded by $\mathcal{O}(N^3 \cdot |\mathcal{R}^{\text{asst}}|^2 \cdot |\mathcal{R}^{\text{aid}}| \cdot R^{\text{aid}}_{max})$. Now, this procedure has to be applied for at most $|E_\lambda| \leq N$ operations in each pass of iteration λ. Furthermore, each iteration can consist of at most $|E_\lambda| \leq N$ passes if only one operation is scheduled in each pass. Finally, the parallel schedule generation scheme requires at most n iterations to schedule all operations (unless the algorithm terminates without a solution as described above). Thus, the time complexity of the parallel schedule generation scheme is

$$\mathcal{O}(n \cdot N^5 \cdot |\mathcal{R}^{\text{asst}}|^2 \cdot |\mathcal{R}^{\text{aid}}| \cdot R^{\text{aid}}_{max}) \tag{4.37}$$

It can be seen that the time complexity of this parallel schedule generation scheme is significantly worse than the time complexity of the parallel SGS for the classical RCPSP described in Section 3.2.3 due to the additional problem of calculating resource transfers to operations $w \in V'_{\text{in}}$. In particular, the algorithm runs in pseudo-polynomial time depending on the maximal capacity R^{aid}_{max} of any resource $l \in \mathcal{R}^{\text{aid}}$ because resource transfers of resources l are chosen independently for each required unit of resource l while the parallel schedule generation scheme for the classical RCPSP runs in polynomial time (cf. Section 3.2.3). The reason for this is discussed in more detail in Section 4.2.3. It can be assumed, however, that this does not pose a problem for most practical applications with scarce resources.

Finally, it should be noted that the time complexity for testing if sufficient space is available in a building section $k \in \mathcal{R}^{\text{sect}}$ in order to schedule an operation $w \in V_{\text{all}}$ at a time point t_λ can be neglected here because, for each building section $k \in \mathcal{R}^{\text{sect}}$, only the amount of available space $R_k(t_\lambda)$ at time point t_λ is stored. Thus, it can be tested in $\mathcal{O}(1)$ if a sufficient amount of space is available in order to schedule operation w.

Schedule Properties

Now, some properties of the schedules generated by the parallel schedule generation scheme described in this section are considered. As shown by Kolisch (1996a), the parallel schedule generation scheme for the classical resource-constrained project scheduling problem generates non-delay schedules (cf. Section 3.2). Here, although the general idea of the algorithm is maintained, this property does not hold for our algorithm as shown in Example 4.5. In particular, this is caused by the problem of selecting resource transfers of resources $l \in \mathcal{R}^{\text{aid}}$ as well as transporting resources $k \in \mathcal{R}^{\text{asst}}$ and is discussed in more detail in Section 4.2.3.

Example 4.5 We consider a problem consisting of $N = 4$ jobs in which space requirements in the building sections of the selected evacuation routes are neglected, i.e. each job can be modeled by only one operation $u = 1, \dots, 4$. Next, one type of assistants (i.e. $\mathcal{R}^{\text{asst}} = \{1\}$) as well as one type of aids (i.e. $\mathcal{R}^{\text{aid}} = \{2\}$) with capacities $R_1 = R_2 = 2$ are available for the evacuation of the patients such that $\mu_{12} = 1$ unit of resource 1 is required in order to transport one unit of resource 2. Now, the processing times p_u and resource requirements r_{uk} as well as the transfer times Δ_{uvk} with $\Delta_{uvk} = \Delta_{uv}$ for both resources $k = 1,2$ are displayed in Table 4.3.

	0	1	2	3	4	5
p_u	0	2	4	1	2	0
r_{u1}	2	1	1	1	2	2
r_{u2}	2	1	0	0	2	2

(a) Operation parameters.

	0	1	2	3	4	5
0	0	2	2	4	3	0
1	2	0	1	2	2	0
2	2	1	0	2	2	0
3	4	2	2	0	2	0
4	3	2	2	2	0	0
5	0	0	0	0	0	0

(b) Transfer times.

Table 4.3: The processing times p_u as well as the resource requirements r_{uk} of the operations $u = 0, \dots, 5$ are displayed in (a) while the transfer times $\Delta_{uvk} = \Delta_{uv}$ for both resources $k = 1,2$ between all pairs of operations $u,v = 0, \dots, 5$ are given in (b).

Below, a schedule as it is generated by the parallel schedule generation scheme based on an arbitrary priority rule is displayed in Figure 4.9. Similar to Example 4.4, time point $t_1 = 2$ is again calculated as the earliest time at which an operation can start based on the required resource transfers. In this case, both, operations 1 and 2 can be scheduled to start at time $t_1 = 2$. Then, after operation 1 has been completed, one assistant can be transferred from operation 1 to operation 3 until time point $t_3 = 6$, i.e. operation 3 can start at this time point.

Then, after operation 3 has been completed at time point $t_4 = 7$, the next time point t_5 has to be calculated at which operation 4 can be scheduled to start transfer-feasibly. For this, the earliest arrival times of both units of resource 2 at operation 4 are calculated. At time point t_4, one unit of resource 2 is located at operation 0 while the other unit of resource 2 is located at operation 1. Now, in order to transport one unit of resource 2 to operation 4, one unit of resource 1 is required, which can be transferred from either operation 2 or from operation 3. Thus, the following earliest arrival times can be calculated for the resulting combinations:

- $2 \rightarrow 0 \rightarrow 4: \quad C_2 + \Delta_{20} + \Delta_{04} = 11$
- $2 \rightarrow 1 \rightarrow 4: \quad C_2 + \Delta_{21} + \Delta_{14} = 10$
- $3 \rightarrow 0 \rightarrow 4: \quad C_3 + \Delta_{30} + \Delta_{04} = 14$
- $3 \rightarrow 1 \rightarrow 4: \quad C_3 + \Delta_{31} + \Delta_{14} = 11$

Now, procedure CalculateResourceTransfers first selects the transfer of resource 1 from operation 2 to operation 1 from where the resource then transports resource 2 to operation 4 based on transfer rule ES because this resource transfer has the smallest arrival time at operation 4. Afterward, only the transfer of resource 1 from operation 3 to operation 0 can be chosen, from where the resource then transports resource 2 to operation 4. These two resource transfers result in the time point $t_5 = 14$, at which operation 4 is then scheduled to start. The makespan of the resulting schedule is $C_{max} = 16$.

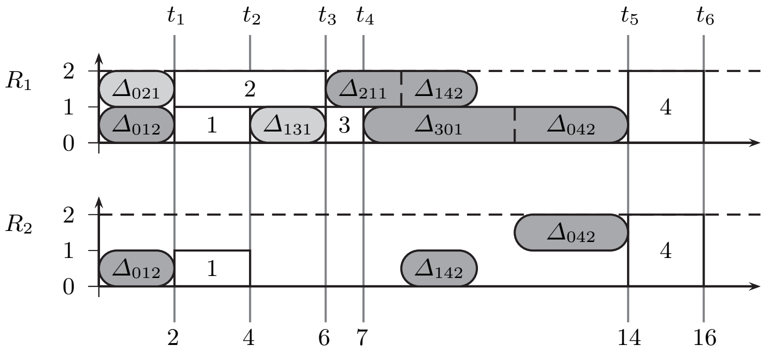


Figure 4.9: A schedule as it is generated by the parallel schedule generation scheme for the problem instance consisting of $n = 4$ real operations from Example 4.5.

This schedule, however, is not a non-delay schedule. Indeed, it is not even a semi-active schedule. Instead, it is possible to shift operation 4 to the left by exchanging the two resource transfers chosen by the transfer rule with the two remaining resource transfers calculated above. In this case, the resulting schedule displayed in Figure 4.10 is a non-delay schedule (i.e. no operation could be locally or globally shifted to the left even if preemption was allowed) with a makespan of $C_{max} = 13$.

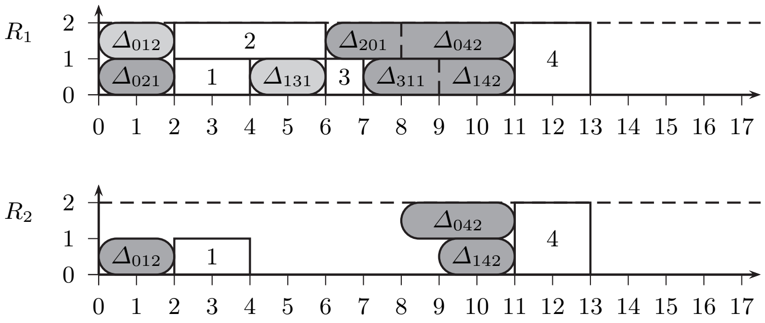


Figure 4.10: Non-delay schedule for the problem instance from Example 4.5.

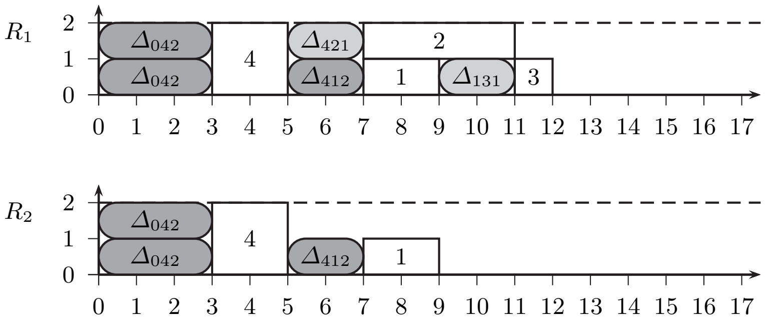


Figure 4.11: Optimal schedule for the problem instance from Example 4.5.

Finally, the unique optimal schedule for this example with a makespan of $C_{max} = 12$ is displayed in Figure 4.11. It should be noted that the parallel schedule generation scheme is not able to generate this schedule because it will always schedule operations 1 and 2 at the earliest time point $t_1 = 2$ regardless of the selected priority rule used to select operations from the set E_λ of eligible operations. $\square$

As can be seen in this example, the schedules generated by the parallel schedule generation scheme implemented for the problem of hospital evacuations can not be classified as belonging to either of the sets of semi-active, active, or non-delay schedules (cf. Section 3.2.2). This is caused by the transfer rule ES, which is not necessarily able to select resource transfers such that all required resources arrive at an operation $w \in V'_{\text{in}}$ as early as possible. Instead, it may be possible to locally shift some operations to the left by selecting different resource transfers as shown in Example 4.5. Also, this example shows that the solution space considered by the parallel schedule generation scheme may not contain an optimal solution.

Deadlocks

Finally, the problem of deadlocks that can occur as a result of the blocking constraint between two or more operations is discussed. If a deadlock occurs, not all operations can be scheduled by the parallel schedule generation scheme and the algorithm terminates without a solution for the given mode assignment. Below, the occurrence of a deadlock is visualized in Example 4.6 based on the evacuation of two patients from a hospital.

Example 4.6 We again consider the hospital from Example 4.1 displayed in Figure 4.1. Now, $N = 2$ patients have to be evacuated such that patient 1 has to be evacuated by one assistant from room 5 to exit 1 while patient 2 has to be evacuated by one assistant from room 3 to exit 2. The operations corresponding to these two jobs as well as the building sections required by the operations are displayed in the activity-on-node network displayed in Figure 4.12. For this example, all real operations $u \in V$ are assigned unit processing times $p_u = 1$ and require $r_{uk} = 1$ unit of space in the corresponding building section $k \in \mathcal{R}^{\text{sect}}$. Also, as before, all building sections $k \in \mathcal{R}^{\text{sect}}$ have a capacity of $R_k = 1$. Finally, in order to evacuate these patients, two assistants are available and are initially located at exit 1. As before, the transfer times for the assistants between the different locations are given in Table 4.1.

Now, a (partial) schedule as it is generated by the parallel schedule generation scheme is displayed in Figure 4.13. In this schedule, operations 1,

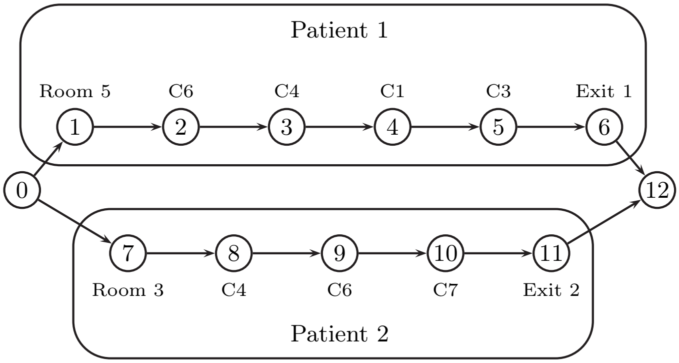


Figure 4.12: Activity-on-node network displaying the operations as well as the building sections required by the operations corresponding to the evacuation routes of the two jobs from Example 4.6.

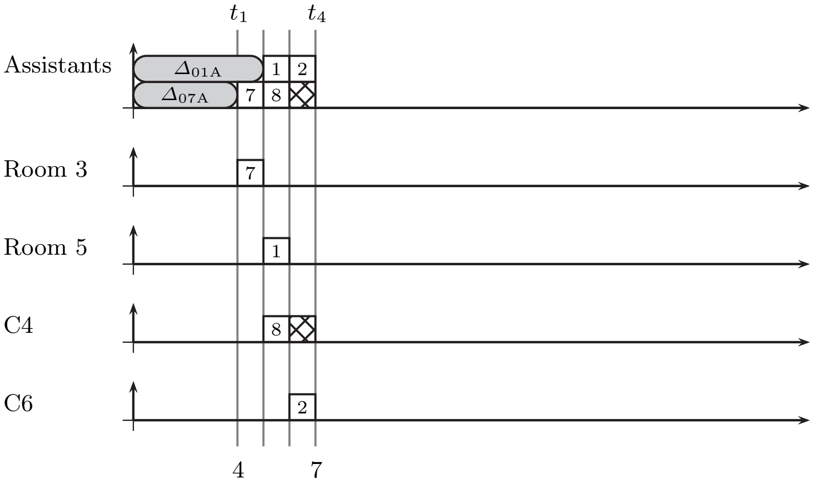


Figure 4.13: Partial schedule as it is generated by the parallel schedule generation scheme for the problem instance from Example 4.6. In iteration $\lambda = 4$, the algorithm terminates due to a deadlock between operations 2 and 8.

7, and 8 could be scheduled to start in the required building sections at time points $t_1 = 4$ and $t_2 = 5$, respectively. Afterward, for iteration $\lambda = 3$, the list $E_3 = \{2, 9\}$ of eligible operations is calculated. Here, both of these operations require building section C3. We assume that the algorithm now schedules operation 2 to start at time $t_3 = 6$ in building section 6 (e.g. based on the priority rule RAND). In this case, operation 8 remains active and blocks building section C4.

Then, in iteration $\lambda = 4$, the algorithm calculates the set $E_4 = \{3, 9\}$ of eligible operations that can be scheduled at time point $t_4 = 7$. From this set, operation 3 requires building section C4 (which is currently occupied by operation 8) while operation 9 requires building section C6 (which is currently occupied by operation 2). Now, because the required building section is occupied for both operations, neither operation can be scheduled in this iteration, i.e the two operations 2 and 8 remain active and building sections C4 and C6 continue to be blocked. This situation is referred to as a deadlock because the parallel schedule generation scheme is unable to schedule operations 3 and 9, i.e. operations 2 and 8 remain active indefinitely. It should be noted that the parallel schedule generation scheme is unable to calculate a new time point t_5 at the end of iteration $\lambda = 4$ because both operations 2 and 8 have already been active for their respective processing times and no further operations $w \in V'_{\text{in}}$ exist. Thus, the next time point is set to $t_5 = UB$ and the algorithm terminates without a solution. □

4.2.2 Serial Schedule Generation Scheme

In this section, the serial schedule generation scheme that has been implemented to solve the scheduling subproblem of the problem of hospital evacuations is introduced. Below, this scheme is outlined in Algorithm 4.3.

Similar to the schedule generation scheme for the classical RCPSP introduced in Section 3.2.3, the general idea of this algorithm is to schedule one operation in each iteration λ such that the operation starts as early as possible with respect to the given precedence constraints, resource requirements, as well as transfer times. For this, a set E_λ of eligible operations is calculated in each iteration λ that contains all operations that can be scheduled precedence- as well as resource-feasibly (i.e. unlike for the parallel schedule generation scheme, all operations contained in the set E_λ can actually be scheduled feasibly at the time that the set E_λ is calculated). Here, the initial set E_1 of eligible operations is calculated in line 2 of the algorithm and contains all operations $w \in V_{\text{in}}$ (i.e. it contains the first operation of the selected evacuation route of the corresponding job $\sigma(w)$).

Algorithm 4.3: Serial Schedule Generation Scheme

```
λ := 1;
Calculate an initial set $E_1$ of eligible operations;
**while** *not all operations have been scheduled* **and** $E_\lambda \neq \emptyset$ **do**
    Choose an eligible operation $w$ from the set $E_\lambda$;
    Calculate the earliest time $t$ at which operation $w$ can be scheduled;
    **if** $w \in V'_{in}$ **then**
        CalculateResourceTransfers($w, t$);
    **end**
    Schedule operation $w$ with starting time $S_w = t$ and update resources;
    **if** $w \in V'_{out}$ **then**
        Set completion time $C_w := S_w + p_w$;
    **end**
    **if** *$w$ has a predecessor operation $u \in V_{\sigma(w)}$* **then**
        Set completion time $C_u := t$;
        Update building section $k \in \mathcal{R}^{\text{sect}}$ required by operation $u$;
    **end**
    $\lambda := \lambda + 1$;
    Calculate the new set $E_\lambda$;
**end**
```

Now, in each iteration λ, one operation $w \in E_\lambda$ from the current set of eligible operations is selected based on a priority rule in line 4 of the algorithm. Similar to the parallel schedule generation scheme, the priority rules listed in Table 4.2 can be used in order to select operations. After an operation $w \in E_\lambda$ has been selected, the earliest time t at which this operation can be scheduled to start is calculated in line 5. This earliest start time t is calculated based on precedence constraints, resource requirements, as well as transfer times as described below.

First of all, a time $t_{\text{pred}} = S_u + p_u$ is calculated as the earliest time at which operation w can be scheduled precedence-feasibly based on the earliest completion times $S_u + p_u$ of its predecessor operation $u \in V_{\sigma(w)}$. If operation w does not have a predecessor operation (i.e. if $w \in V'_{\text{in}}$ holds), the earliest time t_{pred} at which operation w can be scheduled precedence-feasibly is set to $t_{\text{pred}} = 0$. Next, the earliest time t_{sect} is calculated at which a sufficient amount of space is available in the required building section $k \in \mathcal{R}^{\text{sect}}$ in order to schedule operation w. Here, because the completion time of operation w is generally not known at the time when operation w is scheduled due to possible blockings, the operation is not scheduled between already scheduled operations. Instead, it is ensured that a sufficient amount of space is available in the building section from time t_{sect} until the upper bound UB.

Finally, if $w \in V'_{\text{in}}$ holds (i.e. if operation w is the first operation of the selected evacuation route of job $\sigma(w)$), the earliest time at which all required assistants and aids can arrive at operation w has to be calculated. Similar to the parallel schedule generation scheme, this earliest arrival time is calculated by procedure CalculateResourceTransfers based on the transfer rule ES. The earliest time at which operation w can be scheduled transfer-feasibly is then denoted by t_{trf}. Now, the earliest time at which operation w can be scheduled to start is calculated as

$$t = \max\{t_{\text{pred}}, t_{\text{sect}}, t_{\text{trf}}\}.$$

It should be noted that $t_{\text{trf}} = 0$ holds for all operations $w \notin V'_{\text{in}}$ for which no resource transfers of assistants and aids have to be calculated.

After the earliest starting time t for an operation w has been calculated, the operation can be scheduled. Here, if $w \in V'_{\text{in}}$ holds, the actual resource transfers from completed operations $u \in V'_{\text{out}}$ to operation w are chosen in line 7 of the algorithm. These resource transfers are again selected by procedure CalculateResourceTransfers based on the transfer rule ES as described in Section 4.2.1.

Then, in line 9 of the algorithm, the starting time of operation w is set to $S_w = t$ and all required resources are updated. For this, a sufficient amount of space is assigned to operation w in the required building section $k \in \mathcal{R}^{\text{sect}}$ from time t until the upper bound UB. Additionally, if $w \in V'_{\text{in}}$ holds, the selected assistants and aids are transferred from completed operations $u \in V'_{\text{out}}$ to operation w. Instead, if operation w is the last operation of the selected evacuation route of job $\sigma(w)$ (i.e. if $w \in V'_{\text{out}}$ holds), the completion time C_w of this operation is immediately set to $C_w = t + p_w$ in line 11. In this case, the corresponding patient has been evacuated and all resources $k \in \mathcal{R}$ used by this operation become available again at this time. Finally, if operation w has a predecessor operation $u \in V_{\sigma(w)}$, the completion time C_u of this operation is set to $C_u = t$ in line 14 of the algorithm and the building section $k \in \mathcal{R}^{\text{sect}}$ required by operation u becomes available again at this time.

After operation w has been scheduled in iteration λ, the new set $E_{\lambda+1}$ of eligible operations for the next iteration $\lambda + 1$ is calculated in line 18. For this, all unscheduled operations $w \in V_{\text{all}}$ are added to the new set $E_{\lambda+1}$ that can be feasibly scheduled. Here, an operation $w \in V_{\text{all}}$ can be scheduled feasibly if its predecessor operation $u \in V_{\sigma(w)}$ has been scheduled already and a sufficient amount of space is available in the required building section $k \in \mathcal{R}^{\text{sect}}$. Additionally, if $w \in V'_{\text{in}}$ holds, it has to be ensured that a sufficient amount of resources $k \in \mathcal{R}^{\text{trf}}$ can be transferred from completed operations $u \in V'_{\text{out}}$ to operation w.

Finally, after the new set $E_{\lambda+1}$ has been calculated, the next iteration $\lambda+1$ can be started. This continues until either all operations have been scheduled or until the set E_λ is empty in an iteration λ although not all operations have been scheduled. The latter can again occur due to a deadlock if two or more operations block each other.

Example 4.7 We again consider the problem instance from Example 4.4 in which $N = 3$ jobs with a total of $n = 16$ operations have to be scheduled. A schedule for this problem instance as it is generated by the serial schedule generation scheme based on a priority rule that selects operations according to increasing numbers is displayed in Figure 4.14.

At the beginning, the initial set E_1 of eligible operations contains the first operation of each job, i.e. $E_1 = \{1, 7, 12\}$. Now, operation 1 is selected from this set based on the selected priority rule. The earliest starting time t of operation 1 is then calculated as the earliest time at which an assistant from exit 2 (the initial location of both assistants as well as the stretcher) can arrive at room 5 (the initial location of patient 1). In this case, the earliest starting time $t = 3$ is computed and operation 1 is scheduled to start at time $S_1 = 3$.

After operation 1 has been scheduled, the new set E_2 of eligible operations is calculated for the next iteration $\lambda = 2$. Here, this set $E_2 = \{2, 12\}$ contains operation 2, which can now be scheduled precedence- and resource-feasibly in building section C4, as well as operation 12. Operation 7 is not contained in this new set because only one assistant is currently available. Then, as before, the operation with the smallest number (i.e. operation 2) is chosen from the set E_2 and scheduled to start at its earliest precedence- and resource-feasible starting time $S_2 = 4$. After operation 2 has been scheduled, the completion time of operation 1 is set to $C_1 = 4$ and the building section required by operation 1 becomes available again.

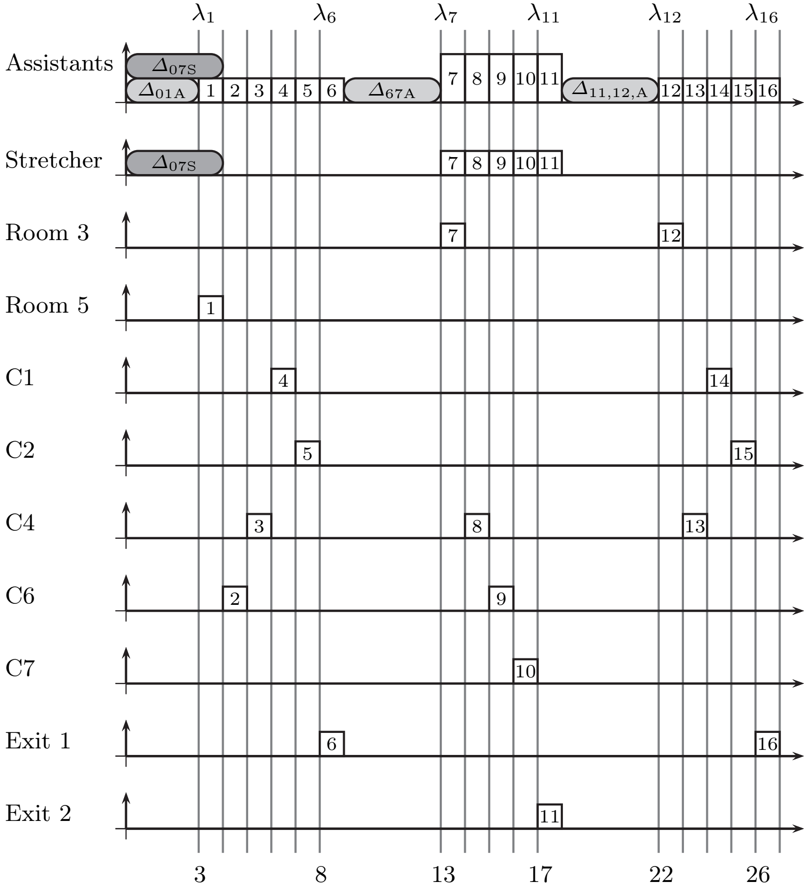


Figure 4.14: A schedule for the problem instance from Example 4.7 as it is generated by the serial schedule generation scheme.

These steps are repeated in iterations $\lambda = 3$ to $\lambda = 6$ until all operations of job 1 have been scheduled. Then, the new set E_7 of eligible operations is calculated as $E_7 = \{7, 12\}$, i.e. it again contains the first operation of the remaining two jobs. From this set, operation 7 is chosen and the earliest starting time $t = 13$ is calculated as the earliest time at which both assistants as well as the stretcher can be transferred to this operation. Here, the assistant located at exit 2 (corresponding to dummy operation 0) is used to transport the stretcher from exit 2 (also corresponding to operation 0) to room 3 (the initial location of patient 2) while the second assistant is directly transferred from exit 1 (corresponding to operation 6) to room 3. After operation 7 has been scheduled to start at time $S_7 = 13$, the remaining operations of job 2 are scheduled in iterations $\lambda = 8$ to $\lambda = 11$. Finally, in iteration $\lambda = 12$, the set $E_{12} = \{12\}$ only contains operation 12. This operation can be scheduled to start at time $S_{12} = 22$ (i.e. at the earliest time at which an assistant from operation 11 can arrive at operation 12) while the remaining operations of job 3 are then scheduled until iteration $\lambda = 16$. The makespan of the resulting schedule is $C_{max} = 27$. □

Time Complexity

Similar to the parallel schedule generation scheme, the time complexity of the serial scheme is again dominated by procedure CalculateResourceTransfers which is used to calculate resource transfers to operations $w \in V'_{\text{in}}$. As computed above, the time complexity of this procedure is bounded by $\mathcal{O}(N^3 \cdot |\mathcal{R}^{\text{asst}}|^2 \cdot |\mathcal{R}^{\text{aid}}| \cdot R^{\text{aid}}_{max})$.

Now, because n iterations have to be performed (i.e. one for each operation $w \in V_{\text{all}}$) and in each of these iterations, resource transfers to at most N operations have to be selected (i.e. once for each operation $w \in V'_{\text{in}}$ when calculating the new set $E_{\lambda+1}$), the time complexity of the serial schedule generation scheme is bounded by

$$\mathcal{O}(n \cdot N^4 \cdot |\mathcal{R}^{\text{asst}}|^2 \cdot |\mathcal{R}^{\text{aid}}| \cdot R^{\text{aid}}_{max}) \tag{4.38}$$

It can be seen that the time complexity of the serial schedule generation scheme is by a factor of N smaller than the time complexity of the parallel schedule generation scheme. This is due to the fact that resource transfers have to be calculated at most once for each operation $w \in V'_{\text{in}}$ during an iteration of the serial schedule generation scheme. In contrast to this, resource transfers may have to be evaluated multiple times for the same operations during one iteration of the parallel schedule generation scheme (i.e. if multiple passes are required).

Finally, it should be noted that testing whether a sufficient amount of space is available in the building section $k \in \mathcal{R}^{\text{sect}}$ required by an operation $w \in V_{\text{all}}$ can be done in $\mathcal{O}(N)$ time (i.e. each building section is used at most once by each job $j \in J$) by storing only the jump points at which the corresponding resource profile changes.

Further Results

Next, we again consider some properties of the schedules generated by the serial schedule generation scheme. Here, unlike the serial schedule generation scheme for the classical RCPSP described in Section 3.2.3, the serial schedule generation scheme implemented to solve the scheduling subproblem of the hospital evacuation problem does not even necessarily generate semi-active schedules. This is again caused by the transfer rule ES used to select resource transfers as visualized in Example 4.5. Due to this restriction, no optimal solution may exist in the set of schedules generated by the serial schedule generation scheme presented in this section.

Finally, similar to the parallel schedule generation scheme, deadlocks can occur if two or more operations block each other. This can again be visualized by Example 4.6 if, in each iteration λ, the same operation is selected from the set of eligible operations E_λ as in this example using the specified priority rule (e.g. the priority rule RAND).

4.2.3 Shortcomings of this Solution Approach

In this section, we discuss some shortcomings of the schedule generation schemes introduced in the previous sections. As already described above, neither the parallel nor the serial schedule generation scheme is necessarily able to generate an optimal solution for a given problem instance due to the transfer rule ES used to select resource transfers. In particular, this problem occurs if an operation $w \in V'_{\text{in}}$ has to be scheduled that requires both assistants as well as aids. Here, in order to schedule operation w, an amount of r_{wk} units of resource $k \in \mathcal{R}^{\text{asst}}$ as well as an amount of r_{wl} units of resource $l \in \mathcal{R}^{\text{aid}}$ have to be transferred from completed operations $u \in V'_{\text{out}}$ to operation w. Furthermore, the transfer of each unit of resource $l \in \mathcal{R}^{\text{aid}}$ has to be supported by an amount of μ_{kl} units of resource $k \in \mathcal{R}^{\text{asst}}$.

In the following, this problem is modeled as an extension of the transportation problem with second-tier resources as it has been introduced by Waldherr et al. (2013). Here, for the transportation problem with second-tier resources, a set $\mathcal{R}$ consisting of first-tier resources as well as a set $\mathcal{Q}$ consisting of second-tier resources are given such that an amount of μ_{kl} units of second-tier resource $k \in \mathcal{Q}$ is required in order to support the transfer of one unit of first-tier resource $l \in \mathcal{R}$. Next, a set $\mathcal{S}$ of supply nodes is given such that an amount of a_{uk} units of resource $k \in \mathcal{Q}$ as well as an amount of a_{ul} units of resource $l \in \mathcal{R}$ are available at node $u \in \mathcal{S}$. Furthermore, a designated demand node D with a demand of b_l units of resource $l \in \mathcal{R}$ is given. Finally, transfer times Δ_{ul} denote the time required to transfer resource $l \in \mathcal{R}$ from node $u \in \mathcal{S}$ to node D while transfer times Δ_{uvk} denote the time required to transfer resource $k \in \mathcal{Q}$ from node $u \in \mathcal{S}$ to node $v \in \mathcal{S}$.

The transportation problem with second-tier resources then consists of the following. A sufficient amount of b_l units of resource $l \in \mathcal{R}$ from nodes $v \in \mathcal{S}$ has to be selected to be transferred to node D. Additionally, if an amount of $0 \leq q \leq a_{vl}$ units of resource $l \in \mathcal{R}$ has been selected to be transferred from node $v \in \mathcal{S}$ to node D and if $\Delta_{vl} > 0$ holds, an amount of $q \cdot \mu_{kl}$ units of resource $k \in \mathcal{Q}$ from nodes $u \in \mathcal{S}$ has to be assigned to support this transfer. Finally, no more than the available amount of a_{uk} units of resource $k \in \mathcal{Q}$ as well as a_{ul} units of resource $l \in \mathcal{R}$ from node $u \in \mathcal{S}$ can be used. The objective is to minimize the latest arrival time of all required resource units of resources $l \in \mathcal{R}$ at node D. Here, the arrival time of a resource $l \in \mathcal{R}$ from supply node $v \in \mathcal{S}$ at the demand node D can not be earlier than time $\Delta_{uvk} + \Delta_{vl}$ if the transfer of resource l is supported by resource $k \in \mathcal{Q}$ from supply node $u \in \mathcal{S}$. The corresponding decision version for this problem is to determine if a solution exists such that all required resource units arrive at node D until a given time T.

Now, the problem of selecting resource transfers of assistants and aids to an operation w as well as assigning assistants to support the transfer of the selected aids can be modeled as an extended transportation problem with second-tier resources as follows. First of all, the set $\mathcal{R}^{\text{aid}}$ corresponds to the set $\mathcal{R}$ of first-tier resources while the set $\mathcal{R}^{\text{asst}}$ corresponds to the set $\mathcal{Q}$ of second-tier resources. Then, a set $\mathcal{S} \subseteq V'_{\text{out}}$ of supply nodes is introduced that contains all completed operations $u \in V'_{\text{out}}$ with completion times C_u. At each of these nodes $u \in \mathcal{S}$, an amount of $a_{uk} \leq r_{uk}$ units of resource $k \in \mathcal{R}^{\text{trf}}$ is available. Next, the operation w that has to be scheduled corresponds to the demand node D. Associated with this demand node are a demand of $b_k = r_{wk}$ units of resource $k \in \mathcal{R}^{\text{trf}}$. Finally, transfer times Δ_{uvk} for all resources $k \in \mathcal{R}^{\text{trf}}$ are given between all pairs of nodes $u, v \in \mathcal{S}$ and transfer times $\Delta_{uk} = \Delta_{uwk}$ for all resources $k \in \mathcal{R}^{\text{trf}}$ are given from all nodes $u \in \mathcal{S}$ to the demand node D.

Thus, the extended transportation problem with second-tier resources differs from the transportation problem with second-tier resources introduced by Waldherr et al. (2013) in the following details. First of all, each supply node $u \in \mathcal{S}$ is assigned a completion time C_u at which the resource units located at this node become available. This completion time has to be taken into account when calculating the arrival times of resources $k \in \mathcal{R}^{\text{trf}}$ at the demand node (i.e. the arrival times have to be calculated according to inequalities (4.2) to (4.4)). Next, instead of only resource units of resources $l \in \mathcal{R}^{\text{aid}}$, the demand node also requires resource units of resources $k \in \mathcal{R}^{\text{asst}}$. Just as for the problem of hospital evacuations considered above, the same resource units of resource k can be used by the demand node that have already been used to support the transfer of resources $l \in \mathcal{R}^{\text{aid}}$ to the demand node. Associated with this difference are additional transfer times Δ_{uk} between supply nodes $u \in \mathcal{S}$ and the demand node for all resources $k \in \mathcal{R}^{\text{asst}}$ such that $\Delta_{uk} \leq \Delta_{ul}$ is assumed to hold for all resources $k \in \mathcal{R}^{\text{asst}}$ and $l \in \mathcal{R}^{\text{aid}}$ with $\mu_{kl} > 0$ (i.e. the transfer time for the supported resource l is always at least as large as the transfer time for the supporting resource k). As a result of this, a problem instance for the transportation problem with second-tier resources as it has been introduced by Waldherr et al. (2013) can be transformed into a problem instance for the extended transportation problem with second-tier resources by setting $C_u = 0$ for all nodes $u \in \mathcal{S}$ and $b_k = 0$ for all resources $k \in \mathcal{R}^{\text{asst}}$.

Now, we show that this problem is NP-hard even if only one type of assistants as well as one type of aids are available (cf. Waldherr et al. (2013)).

Theorem 4.1 *The decision version of the extended transportation problem with second-tier resources described above with $|\mathcal{R}^{asst}| = 1$ and $|\mathcal{R}^{aid}| = 1$ is NP-complete.* □

Proof In order to prove that the decision version of the extended transportation problem with second-tier resources and $|\mathcal{R}^{\text{asst}}| = |\mathcal{R}^{\text{aid}}| = 1$ is NP-complete, we show that the problem belongs to the set $\mathcal{NP}$ of nondeterministically polynomial solvable problems and then show that it is NP-complete by a polynomial-time reduction of the parallel machine scheduling problem $P2|pmtn; r_i|\sum U_i$.

First of all, the extended transportation problem with second-tier resources belongs to the set $\mathcal{NP}$ because it is possible to verify a solution for the problem in $\mathcal{O}(|\mathcal{S}|^2 \cdot |\mathcal{R}^{\text{asst}}| \cdot |\mathcal{R}^{\text{aid}}|)$ time. Here, the time complexity for verifying a solution is dominated by the time required to ensure that all selected resource units of resource $l \in \mathcal{R}$ from supply nodes $v \in \mathcal{S}$ can be transferred to the demand node until time T by resource units of resource $k \in \mathcal{Q}$ from supply nodes $u \in \mathcal{S}$ that have been assigned to support the respective transfers.

Next, we show that the decision version of the extended transportation problem with second-tier resources is NP-complete by a reduction of the machine scheduling problem $P2|pmtn; r_i|\sum U_i$.

Input: The machine scheduling problem $P2|pmtn; r_i|\sum U_i$ consists of a set J that contains n jobs $i = 1, \ldots, n$ with processing times p_i, release dates r_i, as well as due dates d_i. These jobs have to be scheduled on two parallel identical machines such as to minimize the number of late jobs (i.e. jobs i with completion times $C_i > d_i$). Furthermore, the jobs can be preempted, i.e. jobs that are being processed can be suspended and resumed on the same or another machine at a later time. Finally, for the decision version of this problem, an integer value U_{max} is given such that $\sum U_i \leq U_{max}$ has to hold, i.e. it has to be decided if a solution for a given problem instance exists such that no more than U_{max} jobs are late. This problem has been shown to be ordinary NP-complete by Du et al. (1992).

Transformation: First, we calculate a list $t_1 < t_2 < \ldots < t_m$ of strictly increasing time points based on the release dates r_i as well as the due dates d_i of the jobs $i \in J$. Based on these time points, a total of $m-1$ time intervals $I_h = [t_h, t_{h+1}[$ for $h = 1, \ldots, m-1$ can be defined. Below, the set $\mathcal{I} = \{I_1, \ldots, I_{m-1}\}$ contains all time intervals while the set $\mathcal{I}_i \subseteq \mathcal{I}$ contains all time intervals $I_h = [t_h, t_{h+1}[$ with $t_h \geq r_i$ and $t_{h+1} \leq d_i$ during which job $i \in J$ can be processed without being late.

Now, each job $i \in J$ is represented by a job node J_i while the two parallel machines are represented by machines nodes $U^1(I_h)$ and $U^2(I_h)$ for each time interval $I_h \in \mathcal{I}$. Additionally, one interval node $D_{I_h i}$ is introduced for each time interval $I_h \in \mathcal{I}_i$ during which job $i \in J$ can be processed. Finally, one job dummy node $X(J_i)$ for each job node J_i as well as one interval dummy node $Y(D_{I_h i})$ for each interval node $D_{I_h i}$ are introduced. In the following, all of these nodes are contained in the set $\mathcal{S}$ of supply nodes. Furthermore, all of these nodes $u \in \mathcal{S}$ are assigned completion times $C_u = 0$.

Next, an integer value M with

$$M = \max\left\{\max_{i=1}^{n}\{p_i\}, \max_{h=1}^{m-1}\{t_{h+1} - t_h\}\right\} + 1$$

is calculated based on the processing times p_i of the jobs $i \in J$ as well as the lengths of the time intervals I_h. Then, a resource k (with $k \in \mathcal{R}^{\text{asst}}$) is introduced such that an amount of $a_{X(J_i),k} = M - p_i$ units of this resource is available at each job dummy node $X(J_i)$, an amount of $a_{U^1(I_h),k} = a_{U^2(I_h),k} = t_{h+1} - t_h$ units of this resource is available at each machine node $U^1(I_h)$ and $U^2(I_h)$, an amount of $a_{D_{I_h}i,k} = t_{h+1} - t_h$ units of this resource is available at each interval node $D_{I_h}i$, and an amount of $a_{Y(D_{I_h}i),k} = M - (t_{h+1} - t_h)$ units of this resource is available at each interval dummy node $Y(D_{I_h}i)$.

Then, a resource l (with $l \in \mathcal{R}^{\text{aid}}$) is introduced such that an amount of $a_{J_i,l} = 1$ unit of this resource is available at each job node J_i and an amount of $a_{D_{I_h}i,l} = 1$ unit of this resource is available at each interval node $D_{I_h}i$. In the following, an amount of $\mu_{kl} = M$ units of resource k is required in order to support the transfer of one unit of resource l.

Now, the demand node requires an amount of

$$b_l = (n - U_{max}) + \sum_{i \in J} |\mathcal{I}_i|$$

units of resource l (i.e. all resource units of resource l from interval nodes $D_{I_h}i$ as well as $n - U_{max}$ resource units of resource l from the job nodes) and an amount of $b_k = 0$ units of resource k. The upper bound T is set to $T = 2$ (i.e. all required resource units have to arrive at the demand node until this time).

Finally, we introduce transfer times Δ_{uvk} for all resources $k \in \mathcal{R}^{\text{trf}}$ between all pairs of nodes $u,v \in \mathcal{S}$ as well as transfer times Δ_{uk} for all resources $k \in \mathcal{R}^{\text{trf}}$ from nodes $u \in \mathcal{S}$ to the demand node. Here, the transfer time from each job node J_i as well as from each interval node $D_{I_h}i$ to the demand node is set to $\Delta_{J_i,k} = \Delta_{J_i,l} = \Delta_{D_{I_h}i,k} = \Delta_{D_{I_h}i,l} = 1$ for both resources k and l. Next, the transfer time from each job dummy node $X(J_i)$ to the corresponding job node J_i is set to $\Delta_{X(J_i),J_i,k} = 1$ for resource k. Similarly, the transfer time from each interval dummy node $Y(D_{I_h}i)$ to the corresponding dummy node $D_{I_h}i$ is set to $\Delta_{Y(D_{I_h}i),D_{I_h}i,k} = 1$ for resource k. The transfer time from each interval node $D_{I_h}i$ to the corresponding job node J_i is set to $\Delta_{D_{I_h}i,J_i,k} = 1$ for resource k while the transfer times from the machine nodes $U^1(I_h)$ and $U^2(I_h)$ to the corresponding interval nodes $D_{I_h}i$ are set to $\Delta_{U^1(I_h),D_{I_h}i,k} = \Delta_{U^2(I_h),D_{I_h}i,k} = 1$ for resource k. Finally, the transfer time from each node $u \in \mathcal{S}$ to itself is set to $\Delta_{uuk} = \Delta_{uul} = 0$

for both resources k and l while all remaining transfer times between the nodes are set to values strictly larger than one.

This transformation is a polynomial transformation because at most n job nodes J_i as well as n corresponding job dummy nodes $X(J_i)$ have to be considered for n jobs. Similarly, for $m \leq 2n$ time intervals I_h, at most $4n$ machine nodes $U^1(I_h)$ and $U^2(I_h)$, $4n^2$ interval nodes $D_{I_h i}$, and $4n^2$ corresponding interval dummy nodes $Y(D_{I_h i})$ have to be considered. The remaining parameters associated with these nodes can also be computed in polynomial time.

Now, we show that a feasible solution for the machine scheduling problem exists if and only if a feasible solution for the extended transportation problem with second-tier resources exists.

"$\Rightarrow$" First, we show that any feasible solution for the machine scheduling problem $P2|pmtn; r_i|\sum U_i$ corresponds to a feasible solution for the extended transportation problem with second-tier resources. For this, we assume that job $i \in J$ is a punctual job (i.e. a job i with $C_i \leq d_i$ that is completed not later than at time d_i). Then, all $M - p_i$ units of resource k from the corresponding job dummy node $X(J_i)$ are assigned to support the transfer of resource l from job node J_i to the demand node. Now, p_i units of resource k are missing in order to support the transfer of resource l. Here, because job i is a punctual job, it is scheduled to be processed for a total of p_i time units in ν time intervals $I_{i_1}, \ldots, I_{i_\nu} \in \mathcal{I}_i$ (with $\nu \leq m$). These time intervals are considered separately as follows.

We assume that job i is scheduled to be processed for $q_{I_h i} \leq t_{h+1} - t_h$ time units in time interval $I_h \in \{I_{i_1}, \ldots, I_{i_\nu}\}$. Then, $q_{I_h i}$ units of resource k from the corresponding interval node $D_{I_h i}$ are assigned to support the transfer of resource l from job node J_i to the demand node. This is repeated for all time intervals $I_h \in \{I_{i_1}, \ldots, I_{i_\nu}\}$ such that a total of p_i units of resource k are assigned to support the transfer of resource l from job node J_i to the demand node. Thus, all support requirements for the transfer of resource l are satisfied and one unit of resource l can be transferred to the demand node until time $T = 2$. By repeating this for all punctual jobs $i \in J$, a total of $n - U_{max}$ units of resource l are transferred from the corresponding job nodes J_i to the demand node until time $T = 2$.

Next, we consider all interval nodes $D_{I_h i}$. Here, if $q_{I_h i} > 0$ holds and the corresponding job i is being processed for $q^1_{I_h i}$ time units on machine 1 as well as for $q^2_{I_h i}$ time units on machine 2 (with $q^1_{I_h i} + q^2_{I_h i} = q_{I_h i}$) during this time interval I_h, an amount of $q^1_{I_h i}$ units of resource k from machine node $U^1(I_h)$ as well as an amount of $q^2_{I_h i}$ units of resource k from machine node $U^2(I_h)$ are assigned to support the transfer of resource l from interval node $D_{I_h i}$ to the demand node (i.e. all $q_{I_h i}$ units of resource k that are used to support the transfer of resource l from job node J_i to the demand node can be replaced). Additionally, for all interval nodes $D_{I_h i}$, all $t_{h+1} - t_h - q_{I_h i}$ units of resource k available at interval node $D_{I_h i}$ itself as well as $M - (t_{h+1} - t_h)$ units of resource k from the corresponding dummy interval node $Y(D_{I_h i})$ are assigned to support the transfer of resource l from interval node $D_{I_h i}$ to the demand node such that one unit of resource l can arrive at the demand node until time $T = 2$. Thus, all units of resource l from the interval nodes $D_{I_h i}$ can be transferred to the demand node until time $T = 2$ such that all resource demands of the demand node are satisfied.

"$\Leftarrow$" Conversely, we show how a feasible solution for the machine scheduling problem with at most U_{max} late jobs can be constructed from a feasible solution of the extended transportation problem with second-tier resources. For this, we consider a feasible solution for the extended transportation problem with second-tier resources in which

$$b_l = n - U_{max} + \sum_{i \in J} |\mathcal{I}_i|$$

units of resource l arrive at the demand node on time. As there are only $\sum_{i \in J} |\mathcal{I}_i|$ units of resource l that can be transferred from interval nodes $D_{I_h i}$ to the demand node, at least $n' \geq (n - U_{max})$ units of resource l have to be transferred from job nodes J_i to this node.

Now, we assume without loss of generality that resource units of resource l are transferred from job nodes $J_1, \ldots, J_{n'}$ to the demand node in the feasible solution for the extended transportation problem with second-tier resources. In the following, we consider these job nodes J_i with $1 \leq i \leq n'$. Here, in order to transfer one unit of resource l from job node J_i to the demand node, at least p_i units of resource k from interval nodes $D_{I_h i}$ with $I_h \in \mathcal{I}_i$ have to be assigned to support the transfer. Now, we assume that $q_{I_h i} \leq t_{h+1} - t_h$ units of resource k

from interval node $D_{I_h i}$ are assigned to support the transfer. Then, two cases can be differentiated: either one unit of resource l is transferred from this interval node $D_{I_h i}$ to the demand node in the feasible solution for the extended transportation problem with second-tier resources, or it is not. In the latter case, we say that this interval node invalidates the transfer of resource l from job node J_i to the demand node and no longer consider this job. In the former case an amount of $q_{I_h i}$ units of resource k is missing at interval node $D_{I_h i}$ in order to support the transfer of resource l from interval node $D_{I_h i}$ to the demand node. For this reason, an amount of $q^1_{I_h i}$ units of resource k from machine node $U^1(I_h)$ as well as an amount of $q^2_{I_h i}$ units of resource k from machine node $U^2(I_h)$ (with $q^1_{I_h i} + q^2_{I_h i} = q_{I_h i}$) have to be assigned to support the transfer of resource l from interval node $D_{I_h i}$ to the demand node.

Then, the corresponding job i can be scheduled on time in the machine scheduling problem such that it is being processed for a total of $q^1_{I_h i}$ time units on machine 1 as well as for a total of $q^2_{I_h i}$ time units on machine 2 in the corresponding time interval $I_h = [t_h, t_{h+1}[$. Here, because $q_{I_h i} \leq t_{h+1} - t_h$ holds, job i can be scheduled such that it is not being processed by both machines simultaneously. This can be performed for p_i units of resource k from interval nodes $D_{I_h i}$ that are assigned to support the transfer of resource l from job node J_i to the demand node until job i is scheduled for p_i time units on the two available machines between its release date r_i and its due date d_i.

We repeat this for each job node J_i with $i \in \{1, \ldots, n'\}$ for which the transfer of resource l is not invalidated by some interval node. Then, because each interval node $D_{I_h i}$ from which no resource unit of resource l is transferred to the demand node can invalidate at most one job node J_i and at most $n' - (n - U_{max})$ interval nodes $D_{I_h i}$ can invalidate the transfer of their corresponding job node J_i, at least $n - U_{max}$ job nodes are not invalidated and the corresponding jobs can be scheduled as described above.

Thus, because the extended transportation problem with second-tier resources belongs to the set $\mathcal{NP}$ and the ordinary NP-complete machine scheduling $P2|pmtn; r_i|\sum U_i$ could be reduced to the extended transportation problem with second-tier resources by a polynomial-time reduction, the extended transportation problem with second-tier resources and $|\mathcal{R}^{\text{asst}}| = 1$ as well as $|\mathcal{R}^{\text{aid}}| = 1$ is also at least ordinary NP-complete. ■

This theorem shows that calculating resource transfers from completed operations $u \in V'_{\text{out}}$ to an operation $w \in V'_{\text{in}}$ such that all required resource units arrive at operation w as early as possible is an NP-hard problem. For this reason, while the approach used in procedure CalculateResourceTransfers to select resource transfers based on the transfer rule ES may not yield optimal solutions, it is also unlikely that a polynomial time algorithm exists that could solve this problem optimally (i.e. such that the selected resource transfers arrive at operation $w \in V'_{\text{in}}$ as early as possible). It should be noted, however, that the same does not hold for some special cases of the problem. In particular, if only assistants are required to evacuate the patients or if teams of assistants and aids are formed in advance as described in Section 4.1.3, the transfer rule ES is able to select resource transfers such that operation w can be scheduled to start as early as possible (i.e. in this case, no second-tier resources are required).

Finally, it should be noted that the same problem also has to be solved in order to shift operations $u \in V'_{\text{in}}$ to the left (i.e. it has to be ensured that a sufficient amount of resource units can be transferred to operation u until time $S'_u < S_u$). For this reason, it is also not a trivial problem to transform a schedule S into an active (or semi-active) schedule S' by left-shifts as described in Section 3.2.2. Due to these problems encountered in this solution approach based on priority rules, an alternative solution approach based on resource flows for the problem of hospital evacuations is presented in Chapter 7. Before this approach is introduced, however, we first consider the RCPSP with general resource transfers in Chapter 5 and describe a solution approach based on resource flows for this problem in Chapter 6.

5 Resource-Constrained Project Scheduling with Resource Transfers

Resource transfers and, in particular, the transport of resources by other resources are an integral part of the problem of hospital evacuations considered in the previous chapter. For this reason, this chapter deals with the subproblem of resource-constrained project scheduling with resource transfers. The additional problem of blockings as they occur in the problem of hospital evacuations is neglected here. Instead, the focus of this as well as the following chapter is on gaining a better understanding of the RCPSP with resource transfers and developing an heuristic algorithm that is better suited to work on resource flows used to represent the transfer of resources between activities.

This chapter is divided as follows: first of all, in Section 5.1, previous research by Krüger (2009) and Krüger and Scholl (2010) on project scheduling with generalized resource transfers is discussed where resources can be used for both, processing activities as well as supporting the transfer of other resources between activities. Apart from introducing an extension of the classification of setup times by Mika et al. (2006), this section also presents a model by Krüger (2009) that incorporates these generalized resource transfers. Then, in Section 5.2, an alternative model for the resource-constrained project scheduling problem with generalized resource transfers (this problem is referred to as the RCPSP with first- and second-tier resource transfers in the remainder of this thesis) is introduced. Finally, the model by Krüger (2009) as well as the model used in this thesis are compared in Section 5.3.

5.1 Project Scheduling with Resource Transfers in Literature

In Section 3.1.5, a classification of setup times in the context of the resource-constrained project scheduling problem as it has been introduced by Mika et al. (2006) has been described. Now, this section gives a further overview of resource transfers in project scheduling. Here, while the classification by

Mika et al. (2006) is primarily focused on direct resource transfers between activities, this section deals with the problem of resource-constrained project (or multi-project) scheduling with generalized resource transfers as it has been introduced by Krüger (2009) in her PhD thesis as well as in a paper by Krüger and Scholl (2010).

While Krüger (2009) primarily deals with the problem of project scheduling with stand-alone resource transfers (i.e. transfers of resources that do not require the support of other resources for the transfer) in her PhD thesis, she also introduces a broader classification of resource transfers including resource-using as well as resource-consuming transfers (i.e. transfers of resources that require the support of other resources for the transfer). This classification of resource transfers is described in more detail in Section 5.1.1. Based on this classification, Krüger (2009) then describes the resource-constrained (multi-)project scheduling problem with generalized resource transfers and presents a mixed-integer linear programming formulation for this problem. In the following, this problem is described in Section 5.1.2 while a mixed-integer linear programming formulation for this problem as it has been presented by Krüger (2009) is given in Section 5.1.3.

It should be noted that setup (or transfer) times in the context of the resource-constrained project scheduling problem have received only limited attention (cf. Section 3.1.5). In particular, to the best of our knowledge, the resource-constrained project scheduling problem with generalized resource transfers has only been considered by Krüger (2009) as well as Krüger and Scholl (2010) until now.

5.1.1 Classification of Resource Transfers

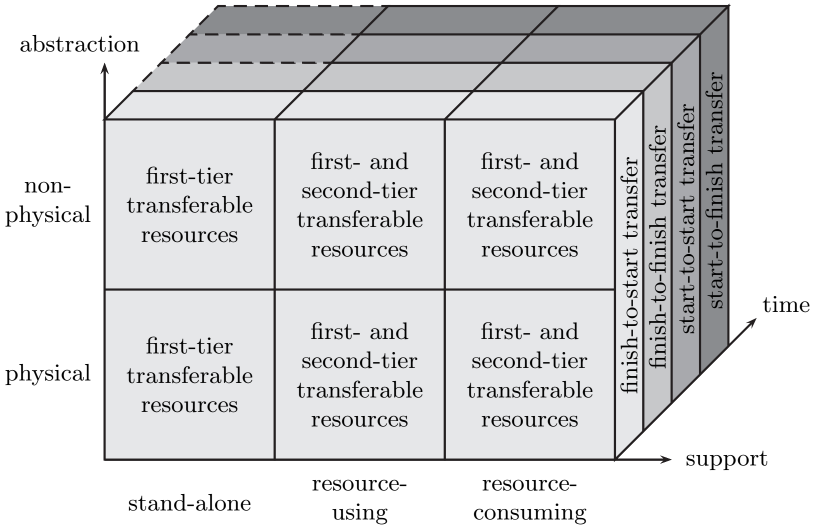
In the following, the classification of resource transfers in the three dimensions of time, abstraction, and support is described as it has been introduced by Krüger (2009). First of all, in the dimension of time, a transfer can originate either at the beginning or at the end of an activity and be directed either to the beginning or to the end of another activity. Thus, a total of four different transfer types can be distinguished, i.e. finish-to-start, start-to-start, finish-to-finish, and start-to-finish transfers. Of these, finish-to-start transfers are the most common type and occur if a resource is transferred from one activity that required this resource in order to be processed to another activity that requires it.

Next, in terms of abstraction, Krüger (2009) distinguishes between physical and non-physical transfers. While physical transfers represent the transfer of a resource from one location to another location, non-physical transfers take place without a change of location. For example, the setup of a machine can be regarded as a non-physical transfer where the machine remains in the same location but a transfer (or setup) time is still required to prepare the machine for the next activity.

Finally, in the third dimension, resource transfers can be differentiated between stand-alone transfers, resource-using transfers, as well as resource-consuming transfers. Here, a stand-alone transfer refers to the transfer of a resource that does not require supporting resources. For example, in the problem of hospital evacuations, the transfer of an assistant between two activities can be regarded as a stand-alone transfer. Next, resource-using transfers refer to the transfer of a resource that requires a supporting renewable resource for the transfer. Again, in the problem of hospital evacuations, the transfer of a wheelchair by an assistant corresponds to a resource-using transfer where the assistant is the supporting (renewable) resource. Finally, resource-consuming transfers occur if non-renewable resources instead of renewable resources are required in order to support the transfer of another resource. For example, the physical transfer of a resource might require money. In this case, the non-renewable resource money is consumed by the transfer.

In the classical resource-constrained project scheduling problem described in Chapter 3, non-renewable resources can be neglected because there either is a sufficient amount of resource units of non-renewable resources or there is not. In contrast to this, non-renewable resources are interesting to consider for the multi-mode RCPSP where the amount of resource units required by an activity depends on the mode in which the activity is executed. Indeed, for the MRCPSP with non-renewable resources, Kolisch and Drexl (1997) have shown that even deciding whether a feasible solution exists is NP-complete (i.e. the problem of assigning a mode to each activity such that the amount of available resource units of non-renewable resources is not smaller than the sum of non-renewable resource units required by the activities).

Here, a similar problem occurs if resource-consuming transfers are considered. In this case, even though the resource requirements are fixed for all activities (i.e. if each activity can only be executed in one mode), the actual amount of non-renewable resource units required during the project depends on the selected resource-consuming resource transfers. Thus, even though a sufficient amount of non-renewable resource units might be available to process all activities, no feasible schedule might exist for a given problem due to resource-consuming resource transfers.



abstraction
non-physical
physical
first-tier transferable resources
first- and second-tier transferable resources
first- and second-tier transferable resources
first-tier transferable resources
first- and second-tier transferable resources
first- and second-tier transferable resources
finish-to-start transfer
finish-to-finish transfer
start-to-start transfer
start-to-finish transfer
time
support
stand-alone
resource-using
resource-consuming

Figure 5.1: Classification of resource transfers along the three dimensions time, abstraction, and support as introduced by Krüger (2009). It should be noted that finish-to-finish, start-to-start, and start-to-finish stand-alone transfers can only occur in combination with resource-using or resource-consuming resource transfers.

Now, the resource units that are being transferred between two activities can be distinguished as either first- or second-tier resource units. Here, a resource unit that is transferred from one activity to another is referred to as a first-tier resource if it is required in order to process the activity that it is being transferred to. On the other hand, a resource unit is referred to as a second-tier resource if it is used to support the transfer of another (first-tier) resource to an activity. Thus, second-tier resource transfers only occur during resource-using or resource-consuming transfers. It should be noted that a renewable resource unit can be used to both, support the transfer of another resource to an activity as well as to process this activity itself. In this case, the resource unit can be regarded as a first- and second-tier resource simultaneously. On the other hand, we refer to a resource unit that is only used to support the transfer of another resource to an activity but not to process the activity itself as a pure second-tier resource.

Finally, Krüger (2009) remarks that higher-tier resource transfers are possible. For example, the transfer of a heavy crane from one construction site to another requires the support of workers to disassemble and then reassemble the crane, as well as lorries to transport it. Additionally, the lorries themselves need supporting resources in the form of drivers, i.e. these drivers support the transfer of the crane indirectly as third-tier resources. In this case, however, higher-tier resources can directly be assigned as second-tier resources to the supported first-tier resource, e.g. the heavy crane in this example (cf. Krüger (2009)).

5.1.2 RCPSP with Generalized Resource Transfers

In this section, the resource-constrained project scheduling problem with generalized resource transfers as it has been introduced by Krüger (2009) is described. In contrast to the resource-constrained project scheduling problem with sequence- and resource-dependent setup (or transfer) times discussed in Section 3.1.5 that only models stand-alone resource transfers between activities, this generalized problem contains all types of resource transfers described above (i.e. stand-alone transfers, resource-using transfers, as well as resource-consuming transfers between the activities).

It should be noted that the original problem formulation by Krüger (2009) deals with the resource-constrained multi-project scheduling problem, i.e. instead of one project, multiple projects sharing at least one resource have to be scheduled. The same problem description as well as the same mixed-integer linear programming formulation for this problem can also be used in order to model the resource-constrained (single) project scheduling problem as it has been introduced in Chapter 3 with only minor adaptions. In particular, instead of multiple projects consisting of multiple activities, only one project consisting of multiple activities has to be scheduled. For this reason, we describe the single-project problem with generalized resource transfers in this section.

As for the classical RCPSP, a total of n real activities $i = 1, \dots, n$ as well as a dummy source activity 0 and a dummy sink activity $n + 1$ have to be scheduled under precedence- and resource-constraints. In the following, the set $V = \{1, \ldots, n\}$ contains all real activities, the set $V_{\text{all}} = \{0, 1, \ldots, n, n+1\}$ contains all real activities as well as the two dummy activities 0 and $n+1$, and the set $A = \{(i,j) \,|\, i,j \in V_{\text{all}}, i \rightarrow j\}$ contains all precedence constraints $i \rightarrow j$ between pairs of activities $i,j \in V_{\text{all}}$ with $i \neq j$. Furthermore, the set $V_0 = V \cup \{0\}$ again contains all real activities as well as the dummy source activity 0 while the set $V_* = V \cup \{n+1\}$ contains all real activities as well as the dummy sink activity $n+1$. Finally, for the mixed-integer linear programming formulation described in Section 5.1.3, additional sets V_{r_i} and V_{s_i} are introduced for each activity $i \in V_{\text{all}}$ such that the set V_{r_i} contains all activities to which resource units can be transferred from activity i (i.e. all activities $j \in V_*$ that are no direct or indirect predecessors of activity i) while the set V_{s_i} contains all activities from which resource units can be transferred to activity i (i.e. all activities $h \in V_0$ that are no direct or indirect successors of activity i).

Next, the set $\mathcal{R}$ contains r renewable resources $k = 1, \ldots, r$ with resource capacities R_k while the set $\mathcal{NR}$ contains ρ non-renewable resources $k = r+1, \ldots, r+\rho$ with capacities R_k. Additionally, the set $\mathcal{R}^* = \mathcal{R} \cup \mathcal{NR}$ contains all renewable and non-renewable resources. Now, a processing time p_i as well as resource requirements r_{ik} for all resources $k \in \mathcal{R}^*$ are associated with all activities $i \in V_{\text{all}}$. It should be noted that dummy source activity 0 as well as dummy sink activity $n+1$ again have processing times $p_0 = p_{n+1} = 0$ and resource requirements $r_{0k} = r_{n+1,k} = R_k$ for all renewable resources $k \in \mathcal{R}$. Similarly, all non-renewable resources $k \in \mathcal{NR}$ are initially located at dummy source activity 0 (i.e. $r_{0k} = R_k$ holds for all $k \in \mathcal{NR}$). As non-renewable resources are consumed during the project, however, the amount of non-renewable resource units that is not consumed during the project is not known in advance. Thus, only those non-renewable resource units that are not consumed during the project have to be collected at dummy sink activity $n+1$.

As stated above, all three support types of stand-alone transfers, resource-using transfers, as well as resource-consuming transfers are integrated into the RCPSP with generalized resource transfers. Additionally, either finish-to-start, finish-to-finish, start-to-start, or start-to-finish transfers are possible. Here, finish-to-start transfers are the most common type of transfers that always occur if a resource that has been used to process one activity is transferred from the end of this activity to the start of the next activity by which it is required. On the other hand, finish-to-finish, start-to-start, and start-to-finish transfers only occur in connection with second-tier resource transfers. For example, start-to-start or start-to-finish transfers only occur if a second-tier resource that has been used to support the transfer of a first-tier resource to an activity $j \in V_*$ is not required to process activity j and can directly be transferred to the next activity from the start of activity j. Similarly, finish-to-finish or start-to-finish transfers to the end of an activity $i \in V_0$ only occur if a second-tier resource is required to support the transfer of a first-tier resource from the end of activity i to another activity $j \in V_*$.

Now, a support matrix μ is introduced such that each element μ_{kl} for k,$l = 1, \ldots, r + \rho$ (with $\mu_{kl} = 0$ for $k = l$) of this matrix denotes the amount of second-tier resource units of resource $k \in \mathcal{R}^*$ required to transfer one unit of first-tier resource $l \in \mathcal{R}^*$. Depending on whether the required second-tier resources $k \in \mathcal{R}^*$ are renewable or non-renewable resources, the transfer of first-tier resource $l \in \mathcal{R}^*$ is either resource-using (i.e. the second-tier resource units of resource $k \in \mathcal{R}$ used for the transfer will be available again after the transfer) or resource-consuming (i.e. the second-tier resource units of resource $k \in \mathcal{NR}$ will be consumed during the transfer). Also, it is possible that both types of resources are required for the transfer of a first-tier resource $l \in \mathcal{R}^*$. It is important to note that the actual amount of second-tier resource units of a non-renewable resource $k \in \mathcal{NR}$ consumed during a transfer depends on the transfer time between the activities such that an amount of μ_{kl} units of non-renewable resource $k \in \mathcal{NR}$ are consumed per time period of the transfer of a first-tier resource $l \in \mathcal{R}^*$.

Next, transfer times Δ_{ijk} are introduced between all pairs of activities $i \in V_0$ and $j \in V_*$ as well as for all resources $k \in \mathcal{R}^*$ such that these transfer times fulfill the triangle inequality $\Delta_{hik} + \Delta_{ijk} \geq \Delta_{hjk}$. This property can often be assumed to hold in practical applications, e.g. if the transfer times depend on the distance between two locations. Furthermore, Krüger (2009) assumes that $\Delta_{ijl} \geq \Delta_{ijk}$ holds for each pair of first-tier resource $l \in \mathcal{R}^*$ and second-tier resource $k \in \mathcal{R}^*$ with $\mu_{kl} > 0$, i.e. the time of the resource transfer is always determined by the supported first-tier resource $l \in \mathcal{R}^*$.

An important property of the model discussed here is the integration of the four transfer types finish-to-start (denoted by FS), finish-to-finish (denoted by FF), start-to-start (denoted by SS), and start-to-finish (denoted by SF) as they have been introduced above. In the following, these transfer types are contained in the set $\mathcal{S} = \{FS, FF, SS, SF\}$. It should be noted that an activity $j \in V_*$ to which resource units of resources $k \in \mathcal{R}^*$ are transferred from activities $i \in V_0$ by transfer type FS or SS (i.e. the resource units are transferred to the start of activity j) can only start after the resource units have arrived. Similarly, activity j can only end after all resource units of resources $k \in \mathcal{R}^*$ that are transferred from activities $i \in V_0$ to activity j by transfer type FF or SF (i.e. the resource units are transferred to the end of activity j) have arrived. This, in turn, delays outgoing resource transfers from either the start or the end of activity j.

Example 5.1 We consider a small project consisting of $n = 5$ activities, $r = 2$ renewable resources with capacities $R_1 = 3$ and $R_2 = 1$, as well as $\rho = 1$ non-renewable resource with a capacity of $R_3 = 6$. Additionally, activities 0 and 6 are the dummy source and dummy sink activity, respectively. The processing times as well as the resource requirements of the activities are given in Table 5.1 while the activity-on-node network displaying the precedence constraints between the activities is shown in Figure 5.2.

i	0	1	2	3	4	5	6
p_i	0	1	1	2	1	2	0
r_{i1}	3	1	2	1	1	2	3
r_{i2}	1	1	0	0	1	0	1
r_{i3}	6	0	2	0	1	0	-

Table 5.1: Processing times p_i and resource requirements r_{ik} for the $n = 5$ real activities $i = 1, \ldots, 5$ as well as the two dummy activities 0 and 6 from the project considered in Example 5.1. The amount of resource units of the non-renewable resource 3 required by dummy sink activity 6 is not known in advance. Instead, all resource units of resource 3 that are not consumed during the execution of the project have to be transferred to activity 6.

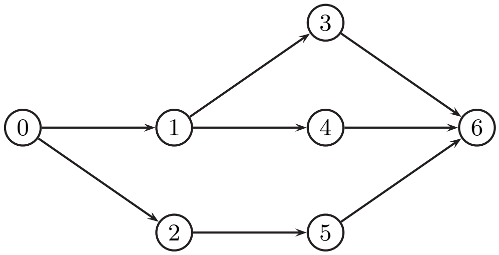


Figure 5.2: Activity-on-node network displaying the precedence constraints between the activities for the project from Example 5.1.

Next, the transfer times Δ_{ijk} for the renewable resources $k = 1,2$ between all pairs of activities $i,j \in V_{\text{all}}$ are given in Table 5.2. The transfer times for the non-renewable resource $k = 3$ are assumed to be $\Delta_{ij3} = 0$ for all $i,j \in V_{\text{all}}$. Finally, in order to transfer one unit of resource 2, an amount of $\mu_{12} = 2$ units of resource 1 as well as an amount of $\mu_{32} = 1$ units of resource 3 per time period are required. Thus, the transfer of resource 2 is both, resource-using as well as resource-consuming. The remaining resources 1 and 3 can be transferred by stand-alone transfer, i.e. they do not require supporting resources. A feasible resource flow for this project is displayed in Figure 5.3.

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	1	2	1	1	0
2	0	1	0	2	1	1	0
3	0	2	2	0	1	1	0
4	0	1	1	1	0	1	0
5	0	1	1	1	1	0	0
6	0	0	0	0	0	0	0

(a) Resource $k = 1$.

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	2	1	2	2	0
2	0	2	0	2	2	1	0
3	0	1	2	0	1	2	0
4	0	2	2	1	0	2	0
5	0	2	1	2	2	0	0
6	0	0	0	0	0	0	0

(b) Resource $k = 2$.

Table 5.2: Transfer times Δ_{ijk} for resources $k = 1$ (cf. (a)) and $k = 2$ (cf. (b)) between all pairs of activities $i,j = 0, \ldots, 6$ for the project considered in Example 5.1.

In this resource flow, several transfers are of particular interest. First of all, the transfer of resource 2 from activity 0 to activity 1 does not require any supporting resources because the transfer time for this resource between the two activities is $\Delta_{012} = 0$. Then, in order to support the transfer of resource 2 from activity 1 to activity 4, two units of the renewable resource 1 as well as two units of the non-renewable resource 3 (i.e. $\mu_{32} = 1$ unit per time period of the transfer time $\Delta_{142} = 2$) are required. Here, one unit of resource 1 is already available at activity 1 while another unit of resource 1 is transferred to activity 1 from the end of activity 2 by finish-to-finish transfer. Similarly, two units of resource 3 are transferred from activity 0 to activity 1 by finish-to-finish transfer.

Next, because the two units of resource 3 transferred from activity 1 to activity 4 are consumed during the transfer, one additional unit of resource

3 has to be transferred to activity 4 before the activity can be executed. For this reason, one additional unit of resource 3 is transferred from activity 0 to activity 4 by finish-to-start transfer. Finally, a start-to-start transfer occurs between activity 4 and activity 5 when one unit of resource 1 that is not required to process activity 4 is transferred from the start of activity 4 to the start of activity 5. It should also be noted that only one unit of resource 3 is not consumed during the project. This resource unit is directly transferred from dummy source activity 0 to dummy sink activity 6.

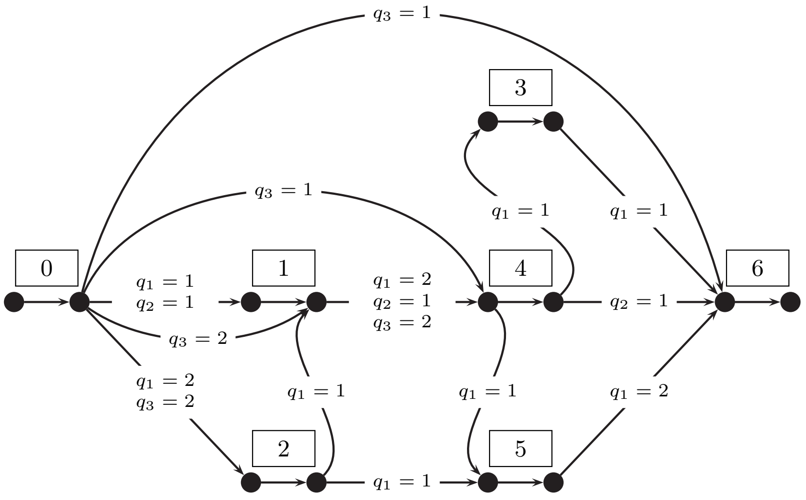


Figure 5.3: Feasible resource flow for the project considered in Example 5.1. Here, each activity $i = 0, \ldots, 6$ is represented by two black circles corresponding to the start and the end of the activity. Now, resource transfers originate at either the start or the end of an activity and terminate at either the start or end of another activity. Each resource transfer is assigned a quantity q_k denoting the amount of resource $k \in \mathcal{R}^*$ transferred between the activities.

Now, a schedule with the makespan $C_{max} = 8$ corresponding to this resource flow is displayed in Figure 5.4. Here, in particular, it can be seen that the start of activity 1 is delayed until one unit of resource 1 from activity 2 can arrive at activity 1 such that $C_2 + \Delta_{211} \leq C_1$ holds. Also, it should be noted that the transfer time of second-tier resource 1 used to support the transfer of first-tier resource 2 from activity 1 to activity 4 requires two time units even though, as a stand-alone transfer, resource 1 could be transferred in only one time unit. This is due to the longer transfer time $\Delta_{142} = 2$ of first-tier resource 2 between the two activities because resource 1 has to support the transfer of resource 2 for the complete duration of the transfer. Similarly, one unit of non-renewable resource 3 is required per time unit of the transfer time to support the transfer of first-tier resource 2, i.e. a total of two units of resource 3 are consumed during the transfer.

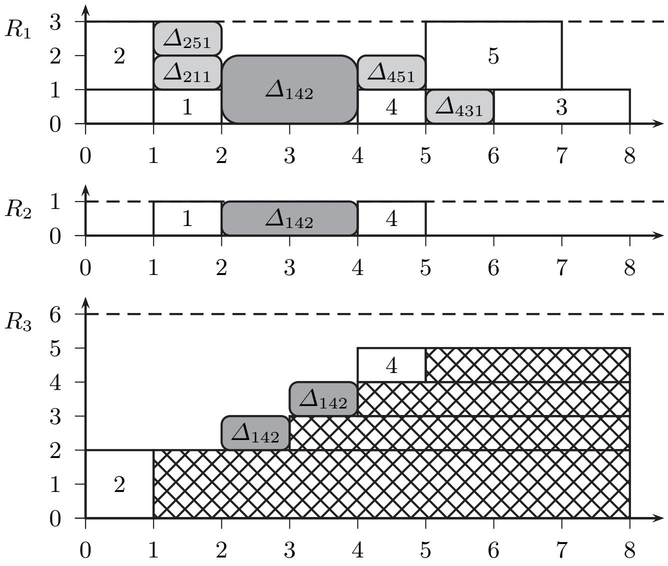


Figure 5.4: A schedule for the project considered in Example 5.1 based on the feasible resource flow displayed in Figure 5.3. Here, stand-alone transfers are marked in light-gray while resource-using or resource-consuming transfers are marked in dark gray. Furthermore, if a non-renewable resource has been consumed either during a transfer or by processing an activity, the resource is not available any more. This is indicated by crosshatched boxes in the corresponding resource profile.

Finally, if only $R_3 = 3$ units of non-renewable resource 3 were available, it should be noted that no feasible schedule exists for the project even though sufficient resource units of resource 3 are available to execute all real activities. This is due to the additional units of non-renewable resources required by resource-consuming transfers. □

As can be seen in Example 5.1, second-tier resources are not directly assigned to transfer specific first-tier resources in the model by Krüger (2009). Instead, it has to be ensured that a sufficient amount of resource units of each required second-tier resource $k \in \mathcal{R}^*$ is available at either the start or the end of activity $i \in V_0$ from which a first-tier resource $l \in \mathcal{R}^*$ is transferred to activity $j \in V_*$. For this, it might be necessary to use additional resource transfers in order to transfer all required second-tier resource units to the activity from which the first-tier resource will be transferred. Only after a sufficient amount of resource units of each required second-tier resource $k \in \mathcal{R}^*$ is available at the activity can the first-tier resource as well as the supporting resource units be transferred to the receiving activity.

5.1.3 Mixed-Integer Linear Programming Formulation

In this section, the mixed-integer linear programming formulation by Krüger (2009) for the problem of resource-constrained project scheduling with generalized resource transfers is described. For this formulation, the following integer and binary variables are required.

First of all, integer variables C_i are introduced for all activities $i \in V_{\text{all}}$ denoting the completion times of the activities. In order to model resource flows, integer variables f^s_{ijk} are introduced for all $i \in V_0$, $j \in V_{r_i}$, $k \in \mathcal{R}^*$, and $s \in \mathcal{S}$ denoting the amount of resource units of resource k transferred from activity i to activity j by transfer type s. Similarly, integer variables $\bar{f}^s_{ijk}$ denote the amount of resource units of second-tier resource $k \in \mathcal{NR}$ consumed during the transfer of any first-tier resource from activity $i \in V_0$ to activity $j \in V_{r_i}$ by transfer type $s \in \mathcal{S}$. Next, binary variables x^s_{ijk} are introduced for all $i \in V_0$, $j \in V_{r_i}$, $k \in \mathcal{R}^*$, and $s \in \mathcal{S}$ with

$$x^s_{ijk} = \begin{cases} 1, & \text{if resource } k \text{ is transferred from activity } i \text{ to activity } j \\ & \text{by transfer type } s \\ 0, & \text{otherwise.} \end{cases}$$

Finally, integer variables α_{ik} and β_{ik} denote the surplus amount of resource units of resource $k \in \mathcal{R}^*$ available at the start of activity $i \in V_0$ (i.e. resource units that are not required to process the activity) that are transferred to the next activity from either the start (α_{ik}) or the end (β_{ik}) of activity i. As before, it should be noted that all parameters used here (i.e. processing times p_i, resource requirements r_{ik}, resource capacities R_k, transfer times Δ_{ijk}, as well as support requirements μ_{kl}) are assumed to be integer.

Now, the mixed-integer linear programming formulation itself is introduced. Again, the objective function (5.1) is to minimize the makespan C_{max} of the project where the makespan is equal to the completion time C_{n+1} of dummy sink activity $n + 1$.

$$\min \quad C_{n+1} \tag{5.1}$$

In the following, the various constraints used to model the RCPSP with generalized resource transfers are introduced. Here, first of all, inequalities (5.2) ensure that the precedence constraints $(i, j) \in A$ between activities $i,j \in V_{\text{all}}$ are satisfied.

$$C_j - C_i \geq p_j \qquad ((i,j) \in A) \tag{5.2}$$

Next, inequalities (5.3) to (5.6) ensure that the finishing times of two activities $i \in V_0$ and $j \in V_{r_i}$ between which resource units of resources $k \in \mathcal{R}^*$ are transferred as either first- or second-tier resources observe the required transfer times Δ_{ijk}. Here, inequalities (5.3) represent finish-to-start transfers, inequalities (5.4) represent finish-to-finish transfers, inequalities (5.5) represent start-to-start transfers, and inequalities (5.6) represent start-to-finish transfers.

$$C_i + \Delta_{ijk} \leq C_j - p_j + T \cdot (1 - x^{FS}_{ijk}) \quad (i \in V_0;\ j \in V_{r_i};\ k \in \mathcal{R}^*) \tag{5.3}$$

$$C_i + \Delta_{ijk} \leq C_j + T \cdot (1 - x^{FF}_{ijk}) \quad (i \in V_0;\ j \in V_{r_i};\ k \in \mathcal{R}^*) \tag{5.4}$$

$$C_i - p_i + \Delta_{ijk} \leq C_j - p_j + T \cdot (1 - x^{SS}_{ijk}) \quad (i \in V_0;\ j \in V_{r_i};\ k \in \mathcal{R}^*) \tag{5.5}$$

$$C_i - p_i + \Delta_{ijk} \leq C_j + T \cdot (1 - x^{SF}_{ijk}) \quad (i \in V_0;\ j \in V_{r_i};\ k \in \mathcal{R}^*) \tag{5.6}$$

Now, inequalities (5.7) and (5.8) ensure that $x^s_{ijk} = 1$ holds for binary variables x^s_{ijk} if and only if resource $k \in \mathcal{R}^*$ is transferred from activity $i \in V_0$ to activity $j \in V_{r_i}$ by transfer type $s \in \mathcal{S}$.

$$f^s_{ijk} \leq R_k \cdot x^s_{ijk} \quad (i \in V_0;\ j \in V_{r_i};\ k \in \mathcal{R}^*;\ s \in \mathcal{S}) \tag{5.7}$$

$$x^s_{ijk} \leq f^s_{ijk} \quad (i \in V_0;\ j \in V_{r_i};\ k \in \mathcal{R}^*;\ s \in \mathcal{S}) \tag{5.8}$$

Below, equations (5.9) and (5.10) model the outgoing resource transfers from dummy source activity 0 where all units of resources $k \in \mathcal{R}^*$ are initially located. These resource units can either be transferred to real activities $j \in$

V or directly to dummy sink activity $n+1$. It should be noted that outgoing resource transfers from the dummy source activity can only originate from the end of the activity (cf. equations (5.10)), i.e. all resource transfers from the start of dummy source activity 0 are set to zero in equations (5.9).

$$\sum_{j \in V_*} \left(f_{0jk}^{SS} + f_{0jk}^{SF} \right) = 0 \quad (k \in \mathcal{R}^*) \tag{5.9}$$

$$\sum_{j \in V_*} \left(f_{0jk}^{FS} + f_{0jk}^{FF} \right) = R_k \ (k \in \mathcal{R}^*) \tag{5.10}$$

Similarly, equations (5.11) to (5.13) model the incoming resource transfers to dummy sink activity $n+1$. Here, all resource units of renewable resources $k \in \mathcal{R}$ are collected at the start of dummy sink activity $n+1$ (cf. equations (5.12)) while only those resource units of non-renewable resources $k \in \mathcal{NR}$ are collected at the start of dummy sink activity $n+1$ that have not been consumed during either processing a real activity $i \in V$ or by supporting a resource transfer as second-tier resources (cf. equations (5.13)). Equations 5.11 ensure that no resource transfers of resources $k \in \mathcal{R}^*$ to the end of dummy source activity $n+1$ can occur.

$$\sum_{i \in V_0} \left(f_{i,n+1,k}^{SF} + f_{i,n+1,k}^{FF} \right) = 0 \qquad (k \in \mathcal{R}^*) \tag{5.11}$$

$$\sum_{i \in V_0} \left(f_{i,n+1,k}^{FS} + f_{i,n+1,k}^{SS} \right) = R_k \qquad (k \in \mathcal{R}) \tag{5.12}$$

$$\sum_{i \in V_0} \left(f_{i,n+1,k}^{FS} + f_{i,n+1,k}^{SS} + \sum_{j \in V_*} \sum_{s \in \mathcal{S}} \bar{f}_{ijk}^{s} \right) = R_k - \sum_{i \in V} r_{ik} \ (k \in \mathcal{NR}) \tag{5.13}$$

The amount of resource units of non-renewable resources $k \in \mathcal{NR}$ consumed during the support of the transfer of a first-tier resource $l \in \mathcal{R}^*$ from activity $i \in V_0$ to activity $j \in V_{r_i}*$ is calculated in equations (5.14) such that μ_{kl} units of resource $k \in \mathcal{NR}$ are required to support the transfer of one unit of first-tier resource $l \in \mathcal{R}^*$ per period of the transfer time Δ_{ijl}.

$$\sum_{l \in \mathcal{R}^*} \mu_{kl} \cdot \Delta_{ijl} \cdot f_{ijl}^{s} = \bar{f}_{ijk}^{s} \quad (i \in V_0;\ j \in V_{r_i};\ k \in \mathcal{NR};\ s \in \mathcal{S}) \tag{5.14}$$

In order to link these variables $\bar{f}_{ijk}^{s}$ with the variables f_{ijk}^{s}, inequalities (5.15) are introduced. These inequalities ensure that the overall amount of f_{ijk}^{s} units of non-renewable resources $k \in \mathcal{NR}$ transferred from activity $i \in V_0$ to activity $j \in V_{r_i}$ has to be at least as large as the amount $\bar{f}_{ijk}^{s}$ of second-tier resource units that is transferred between these activities.

$$\bar{f}^s_{ijk} \leq f^s_{ijk} \quad (i \in V_0;\, j \in V_{r_i};\, k \in \mathcal{NR};\, s \in \mathcal{S}) \tag{5.15}$$

Next, inequalities (5.16) ensure that a sufficient amount of resource units of renewable second-tier resources $k \in \mathcal{R}$ are available to support the transfer of a first-tier resource $l \in \mathcal{R}^*$ from activity $i \in V_0$ to activity $j \in V_{r_i}$. It should be noted that these supporting resources are only required if the transfer time between the two activities is larger than 0 (i.e. if $\Delta_{ijl} > 0$ holds). This is indicated by the binary constant $\lceil \Delta_{ijl}/(\Delta_{ijl}+\varepsilon) \rceil$ where ε is a sufficiently small number.

$$\sum_{l \in \mathcal{R}^*} \mu_{kl} \cdot \left\lceil \frac{\Delta_{ijl}}{\Delta_{ijl} + \varepsilon} \right\rceil \cdot f^s_{ijl} \leq f^s_{ijk} \quad (i \in V_0;\, j \in V_{r_i};\, k \in \mathcal{R};\, s \in \mathcal{S}) \tag{5.16}$$

Finally, the following constraints keep track of the incoming and outgoing resource transfers for each activity. Here, first of all, equations (5.17) ensure that a sufficient amount of resource units of renewable resources $k \in \mathcal{R}$ is available at the start of activity $i \in V$ to satisfy the resource requirements r_{ik} of this activity. Additional resources units that have, for example, been used to support the transfer of other resources to this activity might not be required to process the activity. In this case, these surplus resources can either be transferred to the next activity from the start of activity i (variables α_{ik}) or from the end of activity i (variables β_{ik}).

$$\sum_{h \in V_{s_i}} \left(f^{FS}_{hik} + f^{SS}_{hik} \right) = r_{ik} + \alpha_{ik} + \beta_{ik} \quad (i \in V;\, k \in \mathcal{R}) \tag{5.17}$$

Next, for resource units of non-renewable resources $k \in \mathcal{NR}$ transferred to the start of activity $i \in V$, it has to be taken into account that some of these resource units are consumed either during the transfer if they have been required to support the transfer of first-tier resource units (represented by variables $\bar{f}^{FS}_{hik}$ and $\bar{f}^{SS}_{hik}$) or by the activity itself (represented by the resource requirements r_{ik} of the activity). In this case, the incoming resource transfers are represented by equations (5.18).

$$\sum_{h \in V_{s_i}} \left(f^{FS}_{hik} - \bar{f}^{FS}_{hik} + f^{SS}_{hik} - \bar{f}^{SS}_{hik} \right) = r_{ik} + \alpha_{ik} + \beta_{ik} \quad \begin{pmatrix} i \in V; \\ k \in \mathcal{NR} \end{pmatrix} \tag{5.18}$$

Below, equations (5.19) ensure that an amount of α_{ik} resource units of resource $k \in \mathcal{R}^*$ from the start of activity $i \in V$ that is not required to process activity i (and, in the case of non-renewable resources $k \in \mathcal{NR}$, that has not been consumed during the transfer to activity i) are transferred to activities $j \in V_{r_i}$.

$$\sum_{j \in V_{r_i}} \left(f_{ijk}^{SS} + f_{ijk}^{SF} \right) = \alpha_{ik} \quad (i \in V;\, k \in \mathcal{R}^*) \tag{5.19}$$

Next, equations (5.20) ensure that all resource units of renewable resource $k \in \mathcal{R}$ available at the end of activity i (i.e. r_{ik} resource units required to process activity i, β_{ik} resource units transferred from the start of activity i to the end of activity i, as well as all resource units transferred to the end of activity i from other activities $h \in V_{s_i}$) are transferred to activities $j \in V_{r_i}$.

$$\sum_{j \in V_{r_i}} \left(f_{ijk}^{FS} + f_{ijk}^{FF} \right) = r_{ik} + \beta_{ik} + \sum_{h \in V_{s_i}} \left(f_{hik}^{FF} + f_{hik}^{SF} \right) \quad \begin{pmatrix} i \in V; \\ k \in \mathcal{R} \end{pmatrix} \tag{5.20}$$

Similarly, equations (5.21) ensure that all resource units of non-renewable resources $k \in \mathcal{NR}$ available at the end of activity $i \in V$ are transferred to activities $j \in V_{r_i}$. Here, however, only those resource units can be transferred to other activities that have not been consumed by either activity i or by a resource-consuming transfer to the end of activity i.

$$\sum_{j \in V_{r_i}} \left(f_{ijk}^{FS} + f_{ijk}^{FF} \right) = \beta_{ik} + \sum_{h \in V_{s_i}} \left(f_{hik}^{FF} - \bar{f}_{hik}^{FF} + f_{hik}^{SF} - \bar{f}_{hik}^{SF} \right) \begin{pmatrix} i \in V; \\ k \in \mathcal{NR} \end{pmatrix} \tag{5.21}$$

Finally, the domains of the various variables used in this mixed-integer linear programming formulation are defined by (5.22) through (5.27).

$$C_i \in \mathbb{N} \qquad (i \in V_{\text{all}}) \tag{5.22}$$
$$x_{ijk}^s \in \{0,1\} \qquad (i \in V_0;\, j \in V_{r_i};\, k \in \mathcal{R}^*;\, s \in \mathcal{S}) \tag{5.23}$$
$$f_{ijk}^s \in \mathbb{N} \qquad (i \in V_0;\, j \in V_{r_i};\, k \in \mathcal{R}^*;\, s \in \mathcal{S}) \tag{5.24}$$
$$\bar{f}_{ijk}^s \in \mathbb{N} \qquad (i \in V_0;\, j \in V_{r_i};\, k \in \mathcal{NR};\, s \in \mathcal{S}) \tag{5.25}$$
$$\alpha_{ik} \in \mathbb{N} \qquad (i \in V;\, k \in \mathcal{R}^*) \tag{5.26}$$
$$\beta_{ik} \in \mathbb{N} \qquad (i \in V;\, k \in \mathcal{R}^*) \tag{5.27}$$

5.2 RCPSP with First- and Second-Tier Resource Transfers

As motivated above, resource transfers are an integral part of the MRCPSP model for the problem of hospital evacuations as it has been introduced in Chapter 4. For this reason, we consider the subproblem of resource-constrained project scheduling with first- and second-tier resource transfers in this section. While the model introduced in this section is similar to the resource-constrained (multi-)project scheduling problem with generalized resource transfers as it has been introduced by Krüger (2009), there are several differences between the two models, in particular regarding the management of second-tier (or higher-tier) resource transfers. These differences are discussed in more detail in Section 5.3 at the end of this chapter.

First, however, a formal description of the resource-constrained project scheduling problem with first- and second-tier resource transfers is given in Section 5.2.1 while a mixed-integer linear programming formulation for the problem is introduced in Section 5.2.2.

5.2.1 Problem Description

As stated above, the RCPSP with first- and second-tier resource transfers is similar to the RCPSP with generalized resource transfers outlined in Section 5.1.2. For this reason, only the differences between the two models are highlighted in this section.

First of all, no non-renewable resources are incorporated into the model presented here, i.e. only the set $\mathcal{R}$ of renewable resources $k = 1, \dots, r$ is considered. Additionally, no higher-tier resource transfers are possible in this problem such that a resource $k \in \mathcal{R}$ that requires supporting resources in order to be transferred between two activities may not itself be required to support the transfer of other resources. For this reason, the set $\mathcal{R}$ of renewable resources can be split into two disjunctive subsets $\mathcal{R}^{\text{sa}}$ and $\mathcal{R}^{\text{ru}}$ such that $\mathcal{R} = \mathcal{R}^{\text{sa}} \cup \mathcal{R}^{\text{ru}}$ holds. Then, the set $\mathcal{R}^{\text{sa}}$ contains all resources that can be transferred by stand-alone transfers while the set $\mathcal{R}^{\text{ru}}$ contains all resources that can only be transferred by resource-using transfers such that an amount of μ_{kl} units of second-tier resource $k \in \mathcal{R}^{\text{sa}}$ are required in order to support the transfer of one unit of resource $l \in \mathcal{R}^{\text{ru}}$. It should be noted that these supporting resources $k \in \mathcal{R}^{\text{sa}}$ are only required to support the transfer of a resource $l \in \mathcal{R}^{\text{ru}}$ from activity $i \in V_0$ to activity $j \in V_*$ if the corresponding transfer time between the two activities is larger than zero, i.e. if resource l actually has to be transferred.

Next, transfer times Δ_{ijk} again denote the amount of time required to transfer resource $k \in \mathcal{R}$ from activity $i \in V_0$ to activity $j \in V_*$. As before, the triangle inequality $\Delta_{hik} + \Delta_{ijk} \geq \Delta_{hjk}$ is assumed to hold between activities h,$i \in V_0$ and $j \in V_*$ for resource $k \in \mathcal{R}$. This property is later required in order to prove Theorem 5.1 and can be assumed to hold in many practical applications. Finally, it is again assumed that $\Delta_{ijl} \geq \Delta_{ijk}$ holds for the transfer time between activities $i \in V_0$ and $j \in V_*$ for all resources $k \in \mathcal{R}^{\mathrm{sa}}$ and $l \in \mathcal{R}^{\mathrm{ru}}$ with $\mu_{kl} > 0$.

In the following, the different types of resource transfers that can occur in this model are discussed. Unlike in the model by Krüger (2009), only start-to-start as well as finish-to start transfers are possible in this model (cf. Section 5.3). Moreover, because resources $l \in \mathcal{R}^{\mathrm{ru}}$ are not required to support the transfer of other resources, these resources are always transferred by finish-to-start transfers. Now, five different scenarios can be differentiated regarding the transfer of resources $k \in \mathcal{R}^{\mathrm{sa}}$ from an activity $h \in V_0$ to another activity $j \in V_*$ depending on whether the resource transfers originate from the start or the end of activity h as well as whether or not they are assigned to support the transfer of resources $l \in \mathcal{R}^{\mathrm{ru}}$.

First of all, resource $k \in \mathcal{R}^{\mathrm{sa}}$ can be transferred as pure first-tier resource from activity $h \in V_0$ to activity $j \in V_*$, i.e. the resource units are not required to support the transfer of a resource $l \in \mathcal{R}^{\mathrm{ru}}$ to activity j. In this case, two different scenarios can be differentiated such that the transfer of resource $k \in \mathcal{R}^{\mathrm{sa}}$ can either originate from the start or the end of activity h (i.e. depending on whether the resource units are required to process activity h or not). These two scenarios are visualized in Figure 5.5.

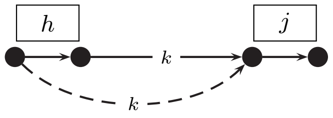


Figure 5.5: Transfer of pure first-tier resource $k \in \mathcal{R}^{\mathrm{sa}}$ from activity $h \in V_0$ to activity $j \in V_*$. Here, the solid line represents a finish-to-start transfer while the dashed line represents a start-to-start transfer.

In these scenarios, inequality (5.28) has to hold for the starting time S_j of activity $j \in V_*$ if resource $k \in \mathcal{R}^{\mathrm{sa}}$ is transferred from activity $h \in V_0$ to activity j by start-to-start transfer. Otherwise, if resource k is transferred by finish-to-start transfer, inequality (5.29) has to hold.

$$S_j \geq S_h + \Delta_{hjk} \tag{5.28}$$

$$S_j \geq S_h + p_h + \Delta_{hjk} \tag{5.29}$$

In the remaining three scenarios, resource $k \in \mathcal{R}^{\text{sa}}$ is transferred as a second-tier resources from activity $h \in V_0$ to support the transfer of first-tier resource $l \in \mathcal{R}^{\text{ru}}$ from activity $i \in V_0$ to activity $j \in V_*$. It should be noted that activity h and activity i do not necessarily have to be different activities (i.e. $h = i$ may hold). As before, the transfer of resource k can originate at either the start or the end of activity h. From there, the resource units of resource k support the transfer of first-tier resource l from activity i to the start of activity j, i.e. resource k is transferred from activity h via activity i to activity j as displayed in Figure 5.6.

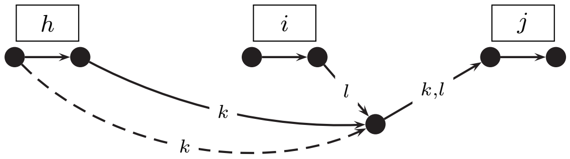


Figure 5.6: Transfer of second-tier resource $k \in \mathcal{R}^{\text{sa}}$ from activity $h \in V_0$ via activity $i \in V_0$ to activity $j \in V_*$ such that resource k supports the transfer of first-tier resource $l \in \mathcal{R}^{\text{ru}}$ from activity i to activity j. As before, the transfer of resource k can either originate at the end (solid line) or the start (dashed line) of activity h.

Unlike before, it is not sufficient to regard the time at which resource $k \in \mathcal{R}^{\text{sa}}$ from activity $h \in V_0$ can be transferred to activity $j \in V_*$ (i.e. either from the start or from the end of activity h) in order to calculate the time at which resource k arrives at activity j. Instead, the arrival of resource k at activity $i \in V_0$ in relation to the completion time C_i of activity i is important to consider. On the one hand, if resource k arrives at activity i before activity i has been completed, the further transfer is delayed until the completion time C_i of activity i. Only then can resource k support the further transfer of first-tier resource $l \in \mathcal{R}^{\text{ru}}$ from activity i to activity j. In this case, inequality (5.30) has to hold for the starting time S_j of activity j.

$$S_j \geq S_i + p_i + \Delta_{ijl} \tag{5.30}$$

On the other hand, if resource k arrives at activity i not earlier than the completion time C_i of activity i, the transfer can immediately continue such that resource k supports the transfer of resource l from activity i to activity

j. In this case, two different scenarios can be distinguished again based on whether resource k is transferred by start-to-start or finish-to-start transfer. Here, inequality (5.31) has to hold for the starting time S_j of activity j if resource k is transferred by start-to-start transfer while inequality (5.32) has to hold if resource k is transferred by finish-to-start transfer.

$$S_j \geq S_h + \mathit{\Delta}_{hik} + \mathit{\Delta}_{ijl} \tag{5.31}$$

$$S_j \geq S_h + p_h + \mathit{\Delta}_{hik} + \mathit{\Delta}_{ijl} \tag{5.32}$$

It should be noted that inequality (5.30) also has to hold if $\mathit{\Delta}_{ijl} = 0$ holds for the transfer time of resource l from activity i to activity j and no supporting resources $k \in \mathcal{R}^{\mathrm{sa}}$ are required. Also in this case, activity j can only start after activity i has been completed (i.e. at time $S_j \geq S_i + p_i$).

A limitation of the model presented here is that second-tier resources $k \in \mathcal{R}^{\mathrm{sa}}$ that support the transfer of first-tier resource $l \in \mathcal{R}^{\mathrm{ru}}$ to activity $j \in V_*$ but are not themselves required to process activity j have to remain at activity j until at least its starting time S_j. Due to this limitation, however, it is possible to reduce the solution space such that first all units of resource $k \in \mathcal{R}^{\mathrm{sa}}$ required to process an activity $j \in V_*$ are used to support the transfer of first-tier resources $l \in \mathcal{R}^{\mathrm{sa}}$ to activity j. Only if this amount of resource units of resource k is not sufficient (i.e. if more than r_{jk} units of second-tier resource k are required), additional units of this resource are transferred to activity j. Thus, the amount of pure second-tier resource units of resource k transferred to activity j is bounded by

$$\max\left\{0, \sum_{l \in \mathcal{R}^{\mathrm{ru}}} \mu_{kl} \cdot r_{jl} - r_{jk}\right\}$$

and depends on the actual amount of resource units of resource k required to support the transfer of first-tier resources $l \in \mathcal{R}^{\mathrm{ru}}$ to activity j. As described above, the actual amount might vary depending on the selected resource transfers of resources $l \in \mathcal{R}^{\mathrm{ru}}$, i.e. if $\mathit{\Delta}_{ijl} = 0$ holds for the transfer time of some resource l from activity $i \in V_0$ to activity j.

Theorem 5.1 *For the resource-constrained project scheduling problem with first- and second-tier resource transfers and a regular objective function, an optimal solution always exists in which pure second-tier resource transfers of resources $k \in \mathcal{R}^{sa}$ to an activity $j \in V_*$ are only used if a larger amount of resource units of resource k is required to support the transfer of first-tier resources $l \in \mathcal{R}^{ru}$ to activity j than is required to process the activity.* $\square$

Proof Let S be an optimal schedule for which the following assumptions can be made without loss of generality. First of all, two activities $j_1 \in V$ (i.e. a real activity) as well as $j_2 \in V_*$ (i.e. either a real activity or the dummy sink activity $n+1$) are given such that both activities require at least $r_{j_1k} = r_{j_2k} = 1$ unit of a resource $k \in \mathcal{R}^{\text{sa}}$. Additionally, activity j_1 requires at least $r_{j_1l} = 1$ unit of a resource $l \in \mathcal{R}^{\text{ru}}$ such that the transfer of this resource requires at least $\mu_{kl} = 1$ unit of second-tier resource k. In the following, these two resources are referred to as k and l. Furthermore, let activities $h_1, h_2 \in V_0$ (i.e. either real activities or the dummy source activity 0) be two activities where one unit of resource k is available at completion times C_{h_1} and C_{h_2}, respectively. Below, the unit of resource k located at activity h_1 is referred to as k_1 while the unit of resource k located at activity h_2 is referred to as k_2. Finally, let activity $i \in V_0$ be an activity where one unit of resource l is available at completion time C_i. Without loss of generality, it is assumed that $C_i \leq C_{h_1} + \Delta_{h_1ik}$ as well as $C_i \leq C_{h_2} + \Delta_{h_2ik}$ hold for the completion times of activities h_1, h_2, and i.

Now, in this schedule S, let resource k_1 from activity h_1 be used to support the transfer of first-tier resource l from activity i to activity j_1 such that this resource is not used to process activity j_1. Instead, resource k_2 from activity h_2 is transferred to activity j_1 in order to process this activity. Finally, resource k_1 is transferred from activity j_1 to activity j_2 by start-to-start transfer. These resource transfers are displayed in Figure 5.7.

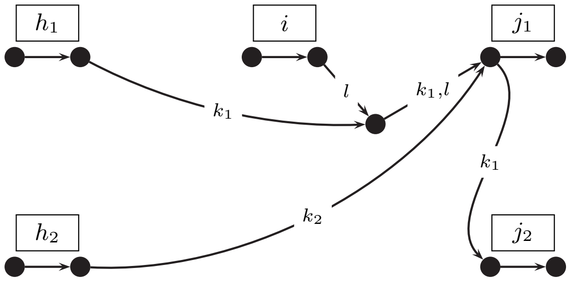


Figure 5.7: Resource transfers in the optimal schedule S.

For this schedule S, we now calculate the starting times S_{j_1} and S_{j_2} of activities j_1 as well as j_2. In order to do this, we consider the arrival of resource units l, k_1, and k_2 at activity j_1 as well as the arrival of resource unit

k_1 at activity j_2. If additional resources are required by the activities, the arrival times of these resources also have to be considered. Here, we assume that a_{j_1} is the latest time at which a resource unit other than resource units k_1, k_2, or l arrives at activity j_1 while a_{j_2} is the latest time at which a resource unit other than resource unit k_1 arrives at activity j_2. Now, assuming that the activities start as early as possible (i.e. as soon as all required resources are available) the starting time S_{j_1} of activity j_1 is given by equation (5.33) while the starting time S_{j_2} of activity j_2 is given by equation (5.34).

$$S_{j_1} = \max\{a_{j_1}, C_{h_1} + \Delta_{h_1 ik} + \Delta_{ij_1 l}, C_{h_2} + \Delta_{h_2 j_1 k}\} \tag{5.33}$$

$$S_{j_2} = \max\{a_{j_2}, \underbrace{\max\{a_{j_1}, C_{h_1} + \Delta_{h_1 ik} + \Delta_{ij_1 l}, C_{h_2} + \Delta_{h_2 j_1 k}\}}_{S_{j_1}} + \Delta_{j_1 j_2 k}\} \tag{5.34}$$

As can be seen here, resource unit k_1 can only be transferred from activity j_1 to activity j_2 after activity j_1 has been started. Thus, the starting time S_{j_2} of activity j_2 depends on the starting time S_{j_1} of activity j_1.

Now, schedule S is transformed into a schedule S' by changing the resource transfers such that resource unit k_1 from activity h_1 is used to support the transfer of resource l from activity i to activity j_1 as well as to process activity j_1. Additionally, resource unit k_2 is directly transferred from activity h_2 to activity j_2. These resource transfers are displayed in Figure 5.8

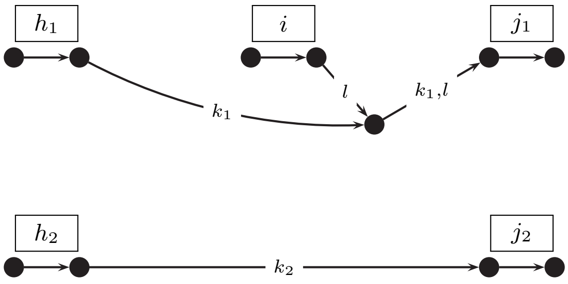


Figure 5.8: Resource transfers in the transformed schedule S'.

In this transformed schedule S', the starting time S'_{j_1} of activity j_1 is given by equation (5.35) while the starting time S'_{j_2} of activity j_2 is given by equation (5.36). It should be noted that the latest arrival times a_{j_1} and a_{j_2} of other resource units to activities j_1 and j_2 are not affected by the modified resource transfers in schedule S'.

$$S'_{j_1} = \max\{a_{j_1}, C_{h_1} + \Delta_{h_1ik} + \Delta_{ij_1l}\} \tag{5.35}$$

$$S'_{j_2} = \max\{a_{j_2}, C_{h_2} + \Delta_{h_2j_2k}\} \tag{5.36}$$

In the following, we compare the starting times of activities j_1 and j_2 in the two schedules S and S'. Here. for the starting time S_{j_1} of activity j_1 in the optimal schedule S as well as for the the starting time S'_{j_1} of activity j_1 in the transformed schedule S', we can see that

$$\begin{aligned} &\underbrace{\max\{\underline{a_{j_1}}, \underline{C_{h_1} + \Delta_{h_1ik} + \Delta_{ij_1l}}, C_{h_2} + \Delta_{h_2j_1k}\}}_{S_{j_1}} \\ \geq\ &\underbrace{\max\{\underline{a_{j_1}}, \underline{C_{h_1} + \Delta_{h_1ik} + \Delta_{ij_1l}}\}}_{S'_{j_1}}. \end{aligned}$$

Thus, the starting time S'_{j_1} of activity j_1 in schedule S' can not be later than the starting time S_{j_1} of activity j_1 in schedule S due to the same arrival time of resource unit k_1 from activity h_1 at activity j_1 as well as the same latest arrival time a_{j_1} of the remaining resource units required by activity j_1. Similarly, by comparing the starting time S_{j_2} of activity j_2 in the optimal schedule S with the starting time S'_{j_2} of activity j_2 in the transformed schedule S', we can see that

$$\begin{aligned} &\underbrace{\max\{\underline{a_{j_2}}, \max\{a_{j_1}, C_{h_1} + \Delta_{h_1ik} + \Delta_{ij_1l}, \underline{C_{h_2} + \Delta_{h_2j_1k}}\}\underline{+\Delta_{j_1j_2k}}\}}_{S_{j_2}} \\ \geq\ &\underbrace{\max\{\underline{a_{j_2}}, \underline{C_{h_2} + \Delta_{h_2j_2k}}\}}_{S'_{j_2}}. \end{aligned}$$

Here, again, the starting time S'_{j_2} of activity j_2 in schedule S' can not be later than the starting time S_{j_2} of activity j_2 in schedule S due to the triangle inequality $C_{h_2} + \Delta_{h_2j_1k} + \Delta_{j_1j_2k} \geq C_{h_2} + \Delta_{h_2j_2k}$ as well as the same latest arrival time a_{j_2} of the remaining resource units required by activity j_2. Thus, as neither activity j_1 nor activity j_2 starts later in the transformed schedule S' than in the optimal schedule S and the starting times of no other activities have been affected, the transformed schedule S' also has to be optimal.

Now, by iteratively changing resource transfers as shown above, it is possible to generate a schedule after a finite number of iterations in which pure second-tier resource transfers of resources $k \in \mathcal{R}^{\text{sa}}$ to an activity $j \in V$ are only used if the amount of r_{jk} units of resource k required to process activity j is not sufficient to support the transfer of all resources $l \in \mathcal{R}^{\text{ru}}$ required by activity j. As this schedule can not be worse than schedule S, it is also an optimal schedule. ■

It should be noted that this property only holds if resource units of resource $k \in \mathcal{R}^{\text{sa}}$ that have been transferred to an activity $j \in V$ as pure second-tier resources units (i.e the resource units are not required to process activity j) can only be transferred from this activity to the next activity after activity j has started at its starting time S_j (as denoted by equation (5.34)). If the resource units can immediately be transferred to the next activity, however, this property does not necessarily hold as visualized in Example 5.2.

Example 5.2 We consider a small project consisting of $n = 3$ real activities as well as $r = 2$ renewable resources with capacities $R_1 = 2$ and $R_2 = 1$. Now, an amount of $\mu_{12} = 1$ unit of resource 1 is required to support the transfer of one unit of resource 2 (i.e. resource 1 belongs to the set $\mathcal{R}^{\text{sa}}$ while resource 2 belongs to the set $\mathcal{R}^{\text{ru}}$). Below, the processing times p_i, resource requirements r_{ik}, as well as transfer times Δ_{ijk} with $\Delta_{ijk} = \Delta_{ij}$ for both resources $k = 1,2$ are given in Table 5.3. It should be noted that the triangle inequality $\Delta_{hi} + \Delta_{ij} \geq \Delta_{hj}$ holds for these transfer times Δ_{ij}. Finally, only the precedence constraint $1 \rightarrow 2$ is given between the real activities.

i	0	1	2	3	4
p_i	0	1	1	1	0
r_{i1}	2	1	1	1	2
r_{i2}	1	0	1	0	1

(a) Activity parameters.

	0	1	2	3	4
0	0	1	2	1	0
1	1	0	1	2	0
2	2	1	0	1	0
3	1	2	1	0	0
4	0	0	0	0	0

(b) Transfer times.

Table 5.3: The processing times p_i as well as the resource requirements r_{ik} of all activities $i = 0, \ldots, 4$ are displayed in (a) while the transfer times $\Delta_{ijk} = \Delta_{ij}$ for both resources $k = 1,2$ between all pairs of activities $i,j = 0, \ldots, 4$ are given in (b).

Now, if pure second-tier resources $k \in \mathcal{R}^{\text{sa}}$ are not required to wait until the starting time S_j of an activity $j \in V$ to which they have supported the transfer of a first-tier resource $l \in \mathcal{R}^{\mathrm{ru}}$, an optimal resource flow for this project is displayed in Figure 5.9. The earliest start schedule corresponding to this resource flow is shown in Figure 5.10.

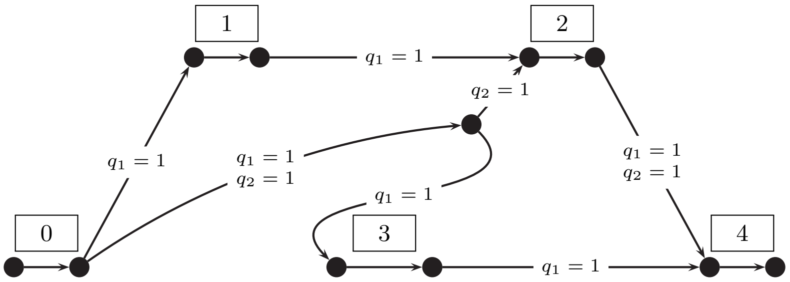


Figure 5.9: Optimal resource flow for the project from Example 5.2. Here, one unit of resource 1 from activity 0 supports the transfer of one unit of resource 2 to activity 2 and is immediately transferred from there to activity 3 (i.e. the further transfer is not delayed until the starting time S_2 of activity 2).

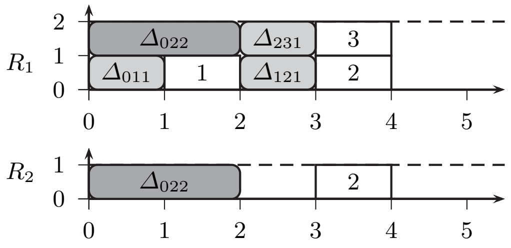


Figure 5.10: Optimal schedule for the project considered in Example 5.2 based on the feasible resource flow displayed in Figure 5.9. Here, the resource unit of resource 1 does not have to remain at activity 2 until the starting time S_2 of this activity but can instead be transferred to activity 3 immediately.

It can be seen that the makespan of the resulting schedule is $C_{max} = 4$. In comparison, if the resource unit of resource 1 has to remain at activity 2 until its starting time S_2 before it can be transferred to activity 3, the best makespan that can be achieved is $C_{max} = 5$. In this case, based on Theorem 5.1, an optimal schedule exists in which one unit of resource 1 that is used to support the transfer of resource 2 to activity 2 can also be used to process activity 2. A schedule visualizing this situation is shown in Figure 5.11.

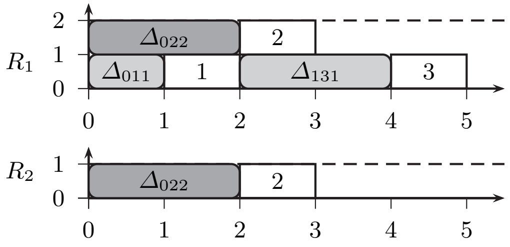


Figure 5.11: Optimal schedule for the project considered in Example 5.2 if pure second-tier resources are delayed until the starting time of the activity to which they have been transferred.

Thus, this example shows that Theorem 5.1 does not hold if resource units of pure second-tier resources $k \in \mathcal{R}^{sa}$ that have been transferred to an activity $j \in V$ do not have to remain at this activity until its starting time S_j but can instead be transferred to the next activity immediately. In this case, it might improve the solution to send additional second-tier resource units even if first-tier resource units required to process the activity could be used as simultaneous first- and second-tier resource units instead. $\square$

5.2.2 Mixed-Integer Linear Programming Formulation

In this section, a mixed-integer linear programming formulation for the RCPSP with first- and second-tier resource transfers is introduced. In this formulation, the following binary and integer variables are used.

First of all, the starting times of all activities $i \in V_{\text{all}}$ are denoted by integer variables S_i. Next, integer variables x_{ijk}^{1st} are introduced for all activities $i \in V_0$ and $j \in V_*$ as well as for all resources $k \in \mathcal{R}$ denoting the amount of first-tier (or simultaneous first- and second-tier) resource $k \in \mathcal{R}$ transferred from activity i to activity j. Similarly, integer variables x_{ijk}^{2nd} are introduced for all $i \in V_0$, $j \in V_*$, and $k \in \mathcal{R}^{\text{sa}}$ denoting the amount of pure second-tier resource k transferred from activity i to activity j. As resources $k \in \mathcal{R}^{\text{sa}}$ can be transferred by either finish-to-start or start-to-start transfer, additional integer variables y_{ijk} are introduced for all $i \in V_0$, $j \in V_*$, and $k \in \mathcal{R}^{\text{sa}}$ denoting the amount of resource k that is transferred by start-to-start transfer from the start of activity i to the start of activity j. Now, binary variables α^{sa}_{ijk} are introduced for all $i \in V_0$, $j \in V_*$, and $k \in \mathcal{R}^{\text{sa}}$ with

$$\alpha^{\text{sa}}_{ijk} = \begin{cases} 1, \text{ if resource } k \text{ is transferred from activity } i \text{ to activity } j \\ 0, \text{ otherwise.} \end{cases}$$

Similarly, binary variables α^{ru}_{ijl} are introduced for all $i \in V_0$, $j \in V_*$, and $l \in \mathcal{R}^{\text{ru}}$ such that

$$\alpha^{\text{ru}}_{ijl} = \begin{cases} 1, \text{ if resource } l \text{ is transferred from activity } i \text{ to activity } j \\ 0, \text{ otherwise.} \end{cases}$$

Additionally, binary variables β_{ijk} are introduced for all $i \in V_0$, $j \in V_*$, as well as for all resources $k \in \mathcal{R}^{\text{sa}}$ with

$$\beta_{ijk} = \begin{cases} 1, \text{ if resource } k \text{ is transferred from the end} \\ \quad \text{of activity } i \text{ to activity } j \\ 0, \text{ otherwise} \end{cases}$$

Here, while resources $l \in \mathcal{R}^{\text{ru}}$ are always transferred by finish-to-start transfer, resources $k \in \mathcal{R}^{\text{sa}}$ can be transferred by either start-to-start or finish-to-start transfer. For this reason, $\alpha_{ijk} = 1$ holds if resource $k \in \mathcal{R}^{\text{sa}}$ is transferred from either the start or the end of activity $i \in V_0$ to activity $j \in V_*$ while $\beta_{ijk} = 1$ only holds if resource k is transferred from the end of activity i to activity j. If resource units of resource k are transferred from both, the start as well as the end of activity i to activity j, it is sufficient to only consider the time at which the resource units from the end of activity i arrive at activity j as this time can not be earlier than for resource units sent from the start of activity i to activity j.

Next, integer variables z_{ijl} are introduced for all activities $i \in V_0$ and $j \in V_*$ as well as for all resources $l \in \mathcal{R}^{\text{ru}}$ denoting the amount of resource units of resource l that require supporting resources $k \in \mathcal{R}^{\text{sa}}$ for the transfer from activity i to activity j. Here, it should be noted that $z_{ijl} > 0$ only holds if $x^{1st}_{ijl} > 0$ and $\Delta_{ijl} > 0$ hold for the transfer of resource l from activity i to activity j. Now, integer variables u_{hijkl} are introduced for all h,$i \in V_0$, $j \in V_*$, $k \in \mathcal{R}^{\text{sa}}$, and $l \in \mathcal{R}^{\text{ru}}$ denoting the amount of resource k transferred from the end of activity h that is used to support the transfer of resource l from activity i to activity j. Similarly, integer variables v_{hijkl} for all $h,i \in V_0$, $j \in V_*$, $k \in \mathcal{R}^{\text{sa}}$, and $l \in \mathcal{R}^{\text{ru}}$ denote the amount of resource k transferred from the start of activity h that is used to support the transfer of resource l from activity i to activity j. Additionally, binary variables γ_{hijkl} are introduced for all $h,i \in V_0$, $j \in V_*$, $k \in \mathcal{R}^{\text{sa}}$, and $l \in \mathcal{R}^{\text{ru}}$ such that

$$\gamma_{hijkl} = \begin{cases} 1, \text{ if resource } k \text{ is transferred from activity } h \text{ to support} \\ \quad \text{the transfer of resource } l \text{ from activity } i \text{ to activity } j \\ 0, \text{ otherwise.} \end{cases}$$

Similarly, binary variables δ_{hijkl} for all $h,i \in V_0$, $j \in V_*$, $k \in \mathcal{R}^{\text{sa}}$, and $l \in \mathcal{R}^{\text{ru}}$ are used with

$$\delta_{hijkl} = \begin{cases} 1, \text{ if resource } k \text{ is transferred from the end of activity } h \text{ to} \\ \quad \text{support the transfer of resource } l \text{ from activity } i \text{ to activity } j \\ 0, \text{ otherwise.} \end{cases}$$

Finally, all parameters used in this formulation (i.e. processing times p_i, resource capacities R_k, resource requirements r_{ik}, support requirements μ_{kl}, as well as transfer times Δ_{ijk}) are assumed to be integer.

Now, the mixed-integer linear programming formulation itself is introduced. Again, the objective is to minimize the makespan (5.37) as given by the starting time S_{n+1} of dummy sink activity $n+1$.

$$\min \quad S_{n+1} \tag{5.37}$$

Next, inequalities (5.38) ensure that all precedence constraints $(i,j) \in A$ between activities $i,j \in V_{\text{all}}$ are adhered to.

$$S_i + p_i - S_j \leq 0 \quad ((i,j) \in A) \tag{5.38}$$

In order to ensure that a sufficient amount of units of resource $k \in \mathcal{R}$ is transferred to each activity $j \in V_*$ to satisfy its resource requirements r_{jk}, equations (5.39) are introduced.

$$\sum_{i \in V_0} x_{ijk}^{1st} = r_{jk} \quad (j \in V_*;\, k \in \mathcal{R}) \tag{5.39}$$

Similarly, equations (5.40) ensure that a sufficient amount of resource units of pure second-tier resource $k \in \mathcal{R}^{\text{sa}}$ is transferred to each activity $j \in V_*$ such that all support requirements μ_{kl} of resource units of resources $l \in \mathcal{R}^{\text{ru}}$ that are transferred to activity j can be satisfied. Using Theorem 5.1, it is possible to calculate the exact amount of required resource units of pure second-tier resource k based on the selected first-tier resource transfers of resources $l \in \mathcal{R}^{\text{sa}}$ from activities $i \in V_0$ to activity j that require supporting resources (as denoted by integer variables z_{ijl}).

$$\sum_{h \in V_0} x_{hjk}^{2nd} = \max\left\{0, \sum_{i \in V_0} \sum_{l \in \mathcal{R}^{\text{ru}}} \mu_{kl} \cdot z_{ijl} - r_{jk}\right\} \quad \begin{pmatrix} j \in V_*; \\ k \in \mathcal{R}^{\text{sa}} \end{pmatrix} \tag{5.40}$$

Below, equations (5.41) are used to calculate the amount of resource units of resource $l \in \mathcal{R}^{\text{ru}}$ that require supporting resources for the transfer from activity $i \in V_0$ to activity $j \in V_*$. Here, the binary constant $\lceil \Delta_{ijl}/(\Delta_{ijl}+\varepsilon) \rceil$ is used in order to ensure that only resource transfers of resource l from activity i to activity j with $\Delta_{ijl} > 0$ are considered (cf. Krüger (2009)). For this, ε is selected as a sufficiently small number.

$$z_{ijl} = \left\lceil \frac{\Delta_{ijl}}{(\Delta_{ijl} + \varepsilon)} \right\rceil \cdot x_{ijl}^{1st} \quad (i \in V_0;\, j \in V_*;\, l \in \mathcal{R}^{\text{ru}}) \tag{5.41}$$

After the incoming resource transfers have been modeled above, constraints related to outgoing resource transfers are introduced below. Here, equations (5.42) represent the outgoing resource transfers of resources $l \in \mathcal{R}^{\text{ru}}$ from activities $i \in V_0$ to activities $j \in V_*$.

$$\sum_{j \in V_*} x_{ijl}^{1st} = r_{il} \quad (i \in V_0;\, l \in \mathcal{R}^{\text{ru}}) \tag{5.42}$$

Next, equations (5.43) model the outgoing resource transfers of resources $k \in \mathcal{R}^{\text{sa}}$ from dummy source activity 0 to activities $j \in V_*$ while equations (5.44) model the outgoing resource transfers of resources $k \in \mathcal{R}^{\text{sa}}$ from real activities $i \in V$ to activities $j \in V_*$. Here, the amount of available resource units of resource $k \in \mathcal{R}^{\text{sa}}$ at dummy source activity 0 is always equal to the resource requirements $r_{0k} = R_k$ of activity 0. On the other hand, for real activities $i \in V$, all r_{ik} units of resource k required by activity i as well as all resource units of resource k that have been transferred to activity i by pure second-tier resource transfer have to be taken into account. For this reason, the actual amount of resource units of resource k available at a real activity i might be larger than r_{ik}.

$$\sum_{j \in V_*} \left(x_{0jk}^{1st} + x_{0jk}^{2nd}\right) = R_k \quad (k \in \mathcal{R}^{\text{sa}}) \tag{5.43}$$

$$\sum_{j \in V_*} \left(x_{ijk}^{1st} + x_{ijk}^{2nd}\right) = r_{ik} + \sum_{h \in V_0} x_{hik}^{2nd} \quad (i \in V;\, k \in \mathcal{R}^{\text{sa}}) \tag{5.44}$$

Now, the amount y_{ijk} of resource units of resource $k \in \mathcal{R}^{\text{sa}}$ that is transferred from the start of activity $i \in V_0$ to activity $j \in V_*$ is calculated. For dummy source activity 0, equations (5.45) ensure that all units of resource k only become available at the end of the activity.

$$\sum_{j \in V_*} y_{0jk} = 0 \quad (k \in \mathcal{R}^{\text{sa}}) \tag{5.45}$$

For the real activities $i \in V$, equations (5.46) ensure that the amount of resource units of resource $k \in \mathcal{R}^{\text{sa}}$ transferred from the start of activity i to other activities $j \in V_*$ is equal to the amount of resource units of resource k that has been transferred to activity i by pure second-tier resource transfer.

$$\sum_{j \in V_*} y_{ijk} = \sum_{h \in V_0} x^{2nd}_{hik} \quad (i \in V;\, k \in \mathcal{R}^{\text{sa}}) \tag{5.46}$$

Finally, inequalities (5.47) ensure that the amount of resource units of resource $k \in \mathcal{R}^{\text{sa}}$ transferred from the start of activity $i \in V$ to activity $j \in V_*$ can not be larger than the overall amount of resource units of resource k transferred from activity i to activity j by first- or second-tier resource transfer.

$$y_{ijk} \leq x^{1st}_{ijk} + x^{2nd}_{ijk} \quad (i \in V;\, j \in V_*;\, k \in \mathcal{R}^{\text{sa}}) \tag{5.47}$$

In the following, the constraints regarding the assignment of resource units of second-tier resources $k \in \mathcal{R}^{\text{sa}}$ to support the transfer of first-tier resources $l \in \mathcal{R}^{\text{ru}}$ are introduced. Here, equations (5.48) ensure that a sufficient amount of resource units of resource $k \in \mathcal{R}^{\text{sa}}$ from either the start (denoted by variables v_{hijkl}) or the end (denoted by variables u_{hijkl}) of activities $h \in V_0$ is assigned to support the transfer of resource $l \in \mathcal{R}^{\text{sa}}$ from activity $i \in V_0$ to activity $j \in V_*$.

$$\sum_{h \in V_0} (u_{hijkl} + v_{hijkl}) = \mu_{kl} \cdot z_{ijl} \quad \begin{pmatrix} i \in V_0;\, j \in V_*; \\ k \in \mathcal{R}^{\text{sa}};\, l \in \mathcal{R}^{\text{ru}} \end{pmatrix} \tag{5.48}$$

Next, inequalities (5.49) ensure that only resource units of resource $k \in \mathcal{R}^{\text{sa}}$ that are transferred from activity $h \in V_0$ to activity $j \in V_*$ by finish-to-start transfer can also be used to support the transfer of resource $l \in \mathcal{R}^{\text{ru}}$ from activity $i \in V_0$ to activity j by finish-to-start transfer.

$$\sum_{i \in V_0} \sum_{l \in \mathcal{R}^{\text{ru}}} u_{hijkl} \leq x^{1st}_{hjk} + x^{2nd}_{hjk} - y_{hjk} \quad \begin{pmatrix} h \in V_0;\, j \in V_*; \\ k \in \mathcal{R}^{\text{sa}} \end{pmatrix} \tag{5.49}$$

Similarly, inequalities (5.50) ensure that only resource units of resource $k \in \mathcal{R}^{\text{sa}}$ that are transferred from activity $h \in V_0$ to activity $j \in V_*$ by start-to-start transfer can also be used to support the transfer of resource $l \in \mathcal{R}^{\text{ru}}$ from activity $i \in V_0$ to activity j by start-to-start transfer.

$$\sum_{i \in V_0} \sum_{l \in \mathcal{R}^{\text{ru}}} v_{hijkl} \leq y_{hjk} \quad (h \in V_0;\, j \in V_*;\, k \in \mathcal{R}^{\text{sa}}) \tag{5.50}$$

Now, the remaining binary variables α^{sa}_{ijk}, α^{ru}_{ijl}, β_{ijk}, γ_{hijkl}, and δ_{hijkl} are calculated in inequalities (5.51) to (5.60).

$$x^{1st}_{ijk} + x^{2nd}_{ijk} \leq R_k \cdot \alpha^{\text{sa}}_{ijk} \quad (i \in V_0;\, j \in V_*; k \in \mathcal{R}^{\text{sa}}) \tag{5.51}$$

$$\alpha^{\text{sa}}_{ijk} \leq x^{1st}_{ijk} + x^{2nd}_{ijk} \quad (i \in V_0;\, j \in V_*; k \in \mathcal{R}^{\text{sa}}) \tag{5.52}$$

$$x^{1st}_{ijl} \leq R_l \cdot \alpha^{\text{ru}}_{ijl} \quad (i \in V_0;\, j \in V_*; l \in \mathcal{R}^{\text{ru}}) \tag{5.53}$$

$$\alpha^{\text{ru}}_{ijl} \leq x^{1st}_{ijl} \quad (i \in V_0;\, j \in V_*; l \in \mathcal{R}^{\text{ru}}) \tag{5.54}$$

$$x^{1st}_{ijk} + x^{2nd}_{ijk} - y_{ijk} \leq R_k \cdot \beta_{ijk} \quad (i \in V_0;\, j \in V_*; k \in \mathcal{R}^{\text{sa}}) \tag{5.55}$$

$$\beta_{ijk} \leq x^{1st}_{ijk} + x^{2nd}_{ijk} - y_{ijk} \quad (i \in V_0;\, j \in V_*; k \in \mathcal{R}^{\text{sa}}) \tag{5.56}$$

$$u_{hijkl} + v_{hijkl} \leq R_k \cdot \gamma_{hijkl} \quad \begin{pmatrix} h,i \in V_0;\, j \in V_*; \\ k \in \mathcal{R}^{\text{sa}}, l \in \mathcal{R}^{\text{ru}} \end{pmatrix} \tag{5.57}$$

$$\gamma_{hijkl} \leq u_{hijkl} + v_{hijkl} \quad \begin{pmatrix} h,i \in V_0;\, j \in V_*; \\ k \in \mathcal{R}^{\text{sa}}, l \in \mathcal{R}^{\text{ru}} \end{pmatrix} \tag{5.58}$$

$$u_{hijkl} \leq R_k \cdot \delta_{hijkl} \quad \begin{pmatrix} h,i \in V_0;\, j \in V_*; \\ k \in \mathcal{R}^{\text{sa}}, l \in \mathcal{R}^{\text{ru}} \end{pmatrix} \tag{5.59}$$

$$\delta_{hijkl} \leq u_{hijkl} \quad \begin{pmatrix} h,i \in V_0;\, j \in V_*; \\ k \in \mathcal{R}^{\text{sa}}, l \in \mathcal{R}^{\text{ru}} \end{pmatrix} \tag{5.60}$$

Finally, the starting times of the activities can be calculated based on the selected resource transfers. For this, the actual transfer times of the resources are calculated based on inequalities (5.28) to (5.32). Here, first of all, inequalities (5.61) calculate the earliest time at which first-tier resource $k \in \mathcal{R}^{\text{sa}}$ from activity $h \in V_0$ can arrive at activity $j \in V_*$. Depending on whether at least one unit of resource k is transferred from the end of activity h, these inequalities represent the two scenarios modeled by inequalities (5.28) and (5.29).

$$S_j \geq S_h + p_h \cdot \beta_{hjk} + \Delta_{hjk} - M \cdot (1 - \alpha^{\text{sa}}_{hjk}) \quad \begin{pmatrix} h \in V_0; \\ j \in V_*; \\ k \in \mathcal{R}^{\text{sa}} \end{pmatrix} \tag{5.61}$$

Here, M is a large integer value that should be chosen such that it is an upper bound on the time required to process all activities. Also, it should be noted that it does not matter if any unit of resource k is transferred from activity h to activity j as a pure first-tier resource (as long as any unit is transferred from activity h to activity j at all) because the transfer time for a direct transfer of resource k can never be larger than the transfer time if resource k is used to support the transfer of resources $l \in \mathcal{R}^{\mathrm{ru}}$ to activity j.

Next, inequalities (5.62) calculate the earliest time at which resource $k \in \mathcal{R}^{\mathrm{sa}}$ can arrive at activity $j \in V_*$ if it is transferred from either the start or the end of activity $h \in V_0$ and has to support the transfer of resource $l \in \mathcal{R}^{\mathrm{ru}}$ from activity $i \in V_0$ to activity j. These inequalities correspond to the two scenarios represented by inequalities (5.31) as well as (5.32) where resource k arrives at activity i not earlier than its completion time C_i, i.e. where the further transfer to activity j is not delayed.

$$S_j \geq S_h + p_h \cdot \delta_{hijkl} + \Delta_{hik} + \Delta_{ijl} - M \cdot (1 - \gamma_{hijkl}) \quad \begin{pmatrix} h,i \in V_0; \\ j \in V_*; \\ k \in \mathcal{R}^{\mathrm{sa}}; \\ l \in \mathcal{R}^{\mathrm{ru}} \end{pmatrix} \tag{5.62}$$

Alternatively, if resource $k \in \mathcal{R}^{\mathrm{sa}}$ arrives at activity $i \in V_0$ before its completion time C_i, the further transfer to activity $j \in V_*$ is delayed until the completion time C_i of activity i (as represented by inequalities (5.30)). In this case, the earliest time at which resource k from activity $h \in V_0$ arrives at activity j is calculated by inequalities (5.63).

$$S_j \geq S_i + p_i + \Delta_{ijl} - M \cdot (1 - \alpha^{\mathrm{ru}}_{ijl}) \qquad \begin{pmatrix} i \in V_0; \\ j \in V_*; \\ l \in \mathcal{R}^{\mathrm{ru}} \end{pmatrix} \tag{5.63}$$

As described above, these latter inequalities always have to hold if resource $l \in \mathcal{R}^{\mathrm{ru}}$ is transferred from activity $i \in V_0$ to activity $j \in V_*$ even if $\Delta_{ijl} = 0$ holds for the transfer time of resource l between the two activities and no supporting resources $k \in \mathcal{R}^{\mathrm{sa}}$ are required.

Finally, the domains of the various variables used in this mixed-integer linear programming formulation are defined by (5.64) through (5.75).

$$S_i \in \mathbb{N} \qquad (i \in V_{\mathrm{all}}) \tag{5.64}$$

$$x^{1st}_{ijk} \in \mathbb{N} \qquad (i \in V_0;\ j \in V_*;\ k \in \mathcal{R}) \tag{5.65}$$

$$x^{2nd}_{ijk} \in \mathbb{N} \qquad (i \in V_o;\ j \in V_*;\ k \in \mathcal{R}^{\mathrm{sa}}) \tag{5.66}$$

$$y_{ijk} \in \mathbb{N} \qquad (i \in V_0;\ j \in V_*;\ k \in \mathcal{R}^{\mathrm{sa}}) \tag{5.67}$$
$$z_{ijl} \in \mathbb{N} \qquad (i \in V_0;\ j \in V_*;\ l \in \mathcal{R}^{\mathrm{ru}}) \tag{5.68}$$
$$\alpha^{\mathrm{sa}}_{ijk} \in \{0,1\} \qquad (i \in V_0;\ j \in V_*;\ k \in \mathcal{R}^{\mathrm{sa}}) \tag{5.69}$$
$$\alpha^{\mathrm{ru}}_{ijl} \in \{0,1\} \qquad (i \in V_0;\ j \in V_*;\ l \in \mathcal{R}^{\mathrm{ru}}) \tag{5.70}$$
$$\beta_{ijk} \in \{0,1\} \qquad (i \in V_0;\ j \in V_*;\ k \in \mathcal{R}^{\mathrm{sa}}) \tag{5.71}$$
$$u_{hijkl} \in \mathbb{N} \qquad (h,i \in V_0;\ j \in V_*;\ k \in \mathcal{R}^{\mathrm{sa}};\ l \in \mathcal{R}^{\mathrm{ru}}) \tag{5.72}$$
$$v_{hijkl} \in \mathbb{N} \qquad (h,i \in V_0;\ j \in V_*;\ k \in \mathcal{R}^{\mathrm{sa}};\ l \in \mathcal{R}^{\mathrm{ru}}) \tag{5.73}$$
$$\gamma_{hijkl} \in \{0,1\} \qquad (h,i \in V_0;\ j \in V_*;\ k \in \mathcal{R}^{\mathrm{sa}};\ l \in \mathcal{R}^{\mathrm{ru}}) \tag{5.74}$$
$$\delta_{hijkl} \in \{0,1\} \qquad (h,i \in V_0;\ j \in V_*;\ k \in \mathcal{R}^{\mathrm{sa}};\ l \in \mathcal{R}^{\mathrm{ru}}) \tag{5.75}$$

5.3 Comparison of the Models

In this section, we compare the RCPSP with generalized resource transfers by Krüger (2009) as it has been described in Section 5.1.2 with the RCPSP with first- and second-tier resource transfers as it has been introduced in Section 5.2.1. In particular, we focus on the different management of second-tier resource transfers used in both models.

First of all, it should be noted that only first- and second-tier resource transfers are considered in the model presented in the previous section. Thus, a resource $k_1 \in \mathcal{R}$ that is required to support the transfer of a resource $l \in \mathcal{R}$ may not itself require the support of another resource $k_2 \in \mathcal{R}$. Also, it is not trivial to integrate higher-tier resource transfers into this model. In contrast to this, higher-tier resource transfers are possible in the RCPSP with generalized resource transfers by Krüger (2009). While this is a limitation of the RCPSP with first- and second-tier resource transfers, higher-tier resource transfers do not occur in the problem of hospital evacuations considered in this thesis. For this reason, this limitation of the model is accepted here.

Next, we consider the management of second-tier resources as it is handled in both models. For this, we assume that the transfer of one unit of first-tier resource $l \in \mathcal{R}$ from activity $i \in V_0$ to activity $j \in V_*$ has to be supported by an amount of $\mu_{kl} > 0$ units of second-tier resource $k \in \mathcal{R}$. Now, in the RCPSP with generalized resource transfers, these μ_{kl} units of resource k have to be explicitly available at the end of activity i before both, one unit of resource l as well as μ_{kl} units of resource k can be transferred to activity j by finish-to-start transfer. Here, these resource units of resource k

can either be available at the end of activity i after they have been used to process activity i, they can be pure-second-tier resource units that have been transferred to the start of activity i and remain at activity i until its end, or they can be transferred from activities $h \in V_0$ to activity i by start-to-finish or finish-to-finish transfer. In the latter case, however, the start of activity i is delayed until all units of resource k that are transferred from activities $h \in V_0$ to activity i by start-to-finish or finish-to-finish transfer can arrive at the end of activity i no later than the completion time of activity i (cf. inequalities (5.4) and (5.6)).

On the other hand, in the RCPSP with first- and second-tier resource transfers, resource units of second-tier resources $k \in \mathcal{R}^{\mathrm{sa}}$ that are required in order to transfer resource $l \in \mathcal{R}^{\mathrm{ru}}$ from activity $i \in V_0$ to activity $j \in V_*$ do not have to be explicitly available at the end of activity i. Instead, the resource units of resource k from activities $h \in V_0$ are specifically assigned to support the transfer of resource l from activity i to activity j. This results in the composite transfers of resource k from activities h to activity i and from there to activity j as shown in Figure 5.6. Furthermore, due to this approach of managing resource transfers, no explicit resource transfers to the end of an activity have to be considered, i.e. start-to-finish and finish-to-finish resource transfers can be neglected in this model. Thus, unlike in the model by Krüger (2009), activities do not have to be delayed unnecessarily and schedules based on this model either have an equal (i.e. all activities still start as early as possible) or better (i.e. no activity is unnecessarily delayed) makespan compared to schedules based on the model by Krüger (2009) for comparable resource flows.

Example 5.3 We consider a small project consisting of $n = 4$ activities $i = 1, \ldots, 4$ as well as the two dummy activities 0 and 5. Additionally, $r = 3$ renewable resources $k = 1{,}2{,}3$ with unit capacities $R_k = 1$ are available. Here, in order to support the transfer of one unit of resource 2, an amount of $\mu_{12} = 1$ unit of resource 1 is required. The remaining resources 1 and 3 can be transferred by stand-alone transfer. Below, the processing times p_i and resource requirements r_{ik} for all activities $i = 0, \ldots, 5$ as well as the transfer times Δ_{ijk} with $\Delta_{ijk} = \Delta_{ij}$ for all resources $k = 1{,}2{,}3$ between all pairs of activities $i{,}j = 0, \ldots, 5$ are given in Table 5.4. Finally, an activity-on-node network visualizing the precedence constraints between the activities is displayed in Figure 5.12.

Now, an optimal resource flow for this problem instance based on the model by Krüger (2009) (cf. Section 5.1.2) is displayed in Figure 5.13 while the

i	0	1	2	3	4	5
p_i	0	1	2	2	1	0
r_{i1}	1	0	1	0	1	1
r_{i2}	1	1	0	0	1	1
r_{i3}	1	1	0	1	0	1

(a) Activity parameters.

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	1	1	1	0
2	0	1	0	1	1	0
3	0	1	1	0	1	0
4	0	1	1	1	0	0
5	0	0	0	0	0	0

(b) Transfer times.

Table 5.4: The processing times p_i as well as resource requirements r_{ik} for all activities $i = 0, \dots, 5$ are displayed in (a) while the transfer times $\Delta_{ijk} = \Delta_{ij}$ for all resources $k = 1,2,3$ between all pairs of activities $i,j = 0, \dots, 5$ are displayed in (b).

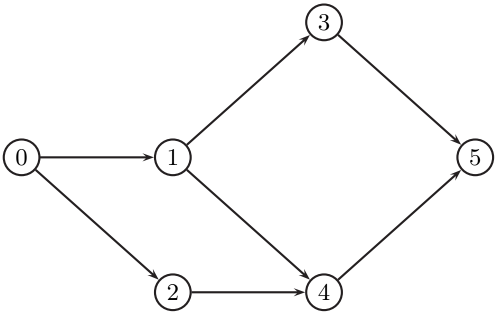

$$\begin{tikzcd}
 & & & 3 \arrow[dr] & \\
0 \arrow[r] \arrow[drr] & 1 \arrow[urr] \arrow[drr] & & & 5 \\
 & & 2 \arrow[r] & 4 \arrow[ur] &
\end{tikzcd}$$

Figure 5.12: Activity-on-node network displaying the precedence constraints between the $n = 4$ real activities $i = 1, \dots, 4$ as well as the two dummy activities 0 and 5.

corresponding earliest start schedule is shown in Figure 5.14. In this resource flow, one unit of resource 1 is transferred as a first-tier resource from the end of activity 2 to the end of activity 1. As a result of this, the start of activity 1 is delayed until this resource unit can arrive at the end of activity 1 before its completion time, i.e. activity 1 can only start at time $S_1 = 2$. Afterward, both, one unit of first-tier resource 2 as well as one supporting unit of second-tier resource 1 are transferred from the end of activity 1 to the start of activity 4. The makespan of the resulting schedule is $C_{max} = 6$.

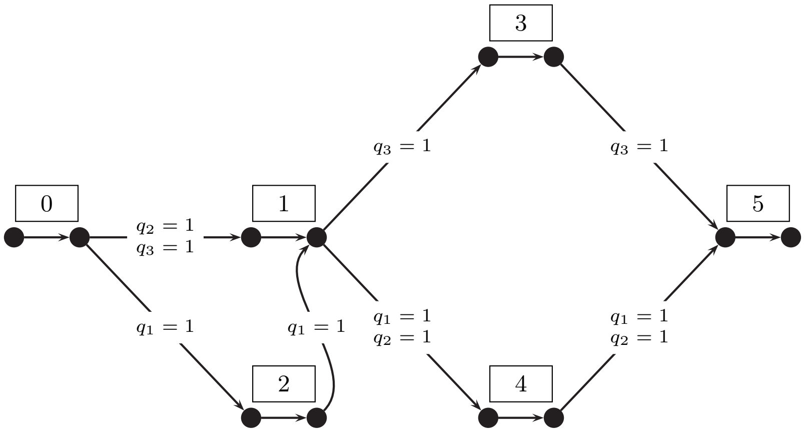


Figure 5.13: Optimal resource flow for the problem instance considered in Example 5.3 based on the model by Krüger (2009).

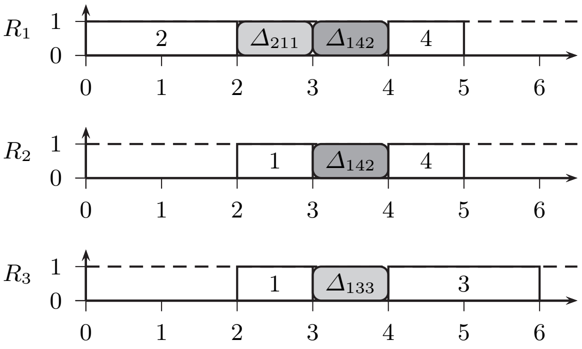


Figure 5.14: The earliest start schedule corresponding to the optimal resource flow displayed in Figure 5.13.

Next, we solve the same problem instance based on the model considered in this thesis (cf. Section 5.2.1). Here, an optimal resource flow is displayed in Figure 5.15 while the corresponding earliest start schedule is shown in Figure 5.16.

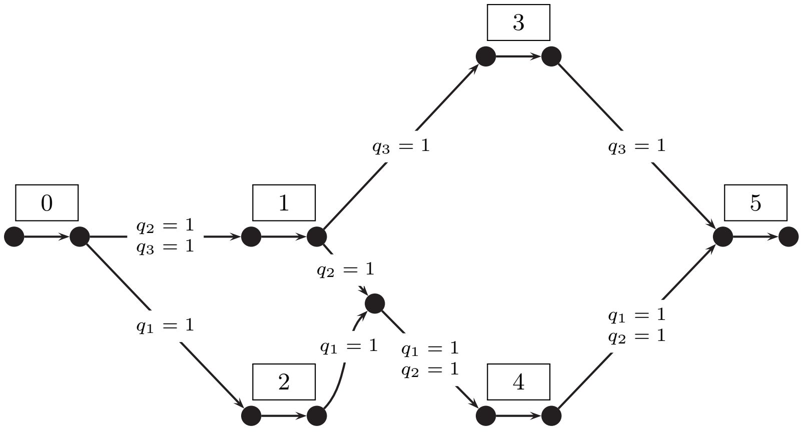


Figure 5.15: Optimal resource flow for the problem instance considered in Example 5.3 based on the model introduced in Section 5.2.

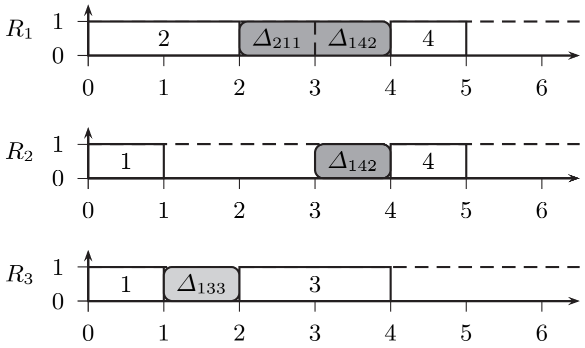


Figure 5.16: The earliest start schedule corresponding to the optimal resource flow displayed in Figure 5.15.

Unlike in the resource flow shown in Figure 5.13, here, one unit of resource 1 does not have to be explicitly transferred from activity 2 to activity 1 before it can support the transfer of resource 2 to activity 4. Instead, one unit of resource 1 from activity 2 is assigned to support the transfer of resource 2 from activity 1 to activity 4 such that the complete transfer of resource 1 from the end of activity 2 via activity 1 to the start of activity 4 can be regarded as a finish-to-start transfer between activity 2 and activity 4. Due to this different management of resource transfers, the start of activity 1 does not have to be delayed in this model and the resulting schedule has a makespan of $C_{max} = 5$. □

As can be seen in Example 5.3, the different management of resource transfers used in the RCPSP with first- and second-tier resource transfers results in a better optimal solution for the same problem instance. This is due to the delays of activities that can occur in the model by Krüger (2009) if resources are transferred by either start-to-finish or finish-to-finish transfers. Additionally, as shown in Theorem 4.1, it is not trivial to shift activities to the left in a given schedule such that all support requirements of first-tier resources are satisfied and the resulting schedule is feasible.

Finally, as already pointed out in Section 5.2.1, no non-renewable resources are integrated into the RCPSP with first- and second-tier resource transfers. For one reason, non-renewable resources are neglected because they are not required in the model for the problem of hospital evacuations presented in Chapter 4. For another reason, it is unlikely that new insights can be gained from incorporating non-renewable resources beyond what has already been described by Krüger (2009). If required, however, non-renewable resources can still be integrated similar to Section 5.1.2.

6 A Solution Approach for the RCPSP with Resource Transfers

After the resource-constrained project scheduling problem with first- and second-tier resource transfers has been introduced in the previous chapter, a solution approach based on resource flows is presented in this chapter. Below, this solution representation is described in more detail in Section 6.1. Also, in this section, some properties of this representation are discussed. In particular, unlike the solution representation based on activity lists often used for the classical RCPSP, this representation allows us to directly incorporate first- and second-tier resource transfers between activities.

Afterward, neighborhoods based on this solution representation are introduced in Section 6.2. For this, modifications are defined on resource flows that redirect resource transfers between activities in order to move between feasible solutions. Apart from a formal definition of these neighborhoods, we also consider some further properties of the neighborhoods, e.g. regarding their connectivity. Finally, a tabu search algorithm for the RCPSP with first- and second-tier resource transfers is presented in Section 6.3. In this algorithm, solutions are represented as resource flows and new solutions are generated based on the modifications described in Section 6.2.

In the following, both, the solution representation as well as the neighborhoods are discussed for three separate problems: the classical RCPSP, the RCPSP with first-tier resource transfers, as well as the RCPSP with first- and second-tier resource transfers. Here, the classical RCPSP is described in detail in Chapter 3 and is modeled based on resource flows by the mixed-integer linear programming formulation (3.6) to (3.14) as it has been introduced by Artigues et al. (2003) as well as Artigues et al. (2010) (cf. Section 3.1.3). Unlike in the other two problems considered in this chapter, resource transfers in this problem do not incur transfer times.

The RCPSP with first-tier resource transfers is discussed in detail by Krüger (2009) as well as Krüger and Scholl (2009) and is modeled by the mixed-integer linear program (3.20) to (3.29) (cf. Section 3.1.5). In this problem, all resources can be transferred as first-tier resources by finish-to-start stand-alone transfers. Finally, the RCPSP with first- and second-tier resource transfers is the topic of the previous chapter and is modeled by the mixed-integer linear programming formulation (5.37) to (5.75) (cf. Section 5.2.2).

6.1 Solution Representation

As described by Brucker and Knust (2006), solutions for the classical RSPSP are often represented by activity lists. These activity lists can then be transformed into a schedule by using, for example, a schedule generation scheme that schedules the activities in the order specified by the activity list. For the RCPSP with resource transfers where each resource transfer incurs a cost (e.g. a transfer time), a drawback of this representation is that it does not represent from which activities resource units are transferred to other activities. Instead, resource transfers have to be selected separately, for example based on a priority rule. As a result of this, a local search algorithm using this approach has only limited influence on how resource transfers are chosen (limited to, for example, selecting one out of several available priority rules).

Due to this drawback, an alternative solution representation based on resource flows is introduced in this section. Unlike activity lists, resource flows exactly represent the amount of resource units of a resource $k \in \mathcal{R}$ transferred from activity $i \in V_0$ to activity $j \in V_*$. Then, based on the neighborhoods described in Section 6.2, it is possible for a local search algorithm to directly operate on these resource flows and adapt resource transfers between activities. In the following, this solution representation is described for the classical RCPSP in Section 6.1.1, for the RCPSP with first-tier resource transfers in Section 6.1.2, as well as for the RCPSP with first- and second-tier resource transfers in Section 6.1.3.

6.1.1 Classical RCPSP

The resource flow representation can be regarded as an extension of the disjunctive graph model as it has been introduced by Roy and Sussmann (1964) which is often used as a solution representation for the job-shop scheduling problem (cf. Fortemps and Hapke (1997)). Here, while a disjunctive graph represents the sequence in which disjunctive machines (i.e.

resources with unit capacity) are used to process operations in a job-shop scheduling problem, this representation is extended for the RCPSP to also denote the amount of resource units of resources $k \in \mathcal{R}$ transferred from an activity $i \in V_0$ to another activity $j \in V_*$.

In the following, f_{ijk} denotes the amount of resource units of resource $k \in \mathcal{R}$ transferred from activity $i \in V_0$ to activity $j \in V_*$. All resource transfers f_{ijk} of a resource $k \in \mathcal{R}$ between activities $i \in V_0$ and $j \in V_*$ are then denoted by resource flow $\mathcal{F}_k$ while a set of resource flows $\mathcal{F}_k$ for all resources $k \in \mathcal{R}$ is denoted by resource flow $\mathcal{F}$. Now, a feasible resource flow $\mathcal{F}$ is defined as a flow that satisfies the following conditions. First of all, all resource units are initially located at dummy source activity 0 and have to be collected by dummy sink activity $n + 1$ after the completion of the project. Then, for each activity $i \in V$, the amount of all incoming resource units as well as the amount of all outgoing resource units of a resource $k \in \mathcal{R}$ has to be equal to the resource requirement r_{ik} of this activity. These constraints can also be regarded as flow conservation constraints. Finally, the resource flows have to be acyclic (i.e. no activity may directly or indirectly sent resource units to itself) and observe the given precedence constraints $(i, j) \in A$ (i.e. activity j may not directly or indirectly sent resource units to activity i).

Each feasible resource flow $\mathcal{F}$ can be represented by a graph such that each activity $i \in V_{\text{all}}$ is represented by a node and each resource transfer $f_{ijk} > 0$ of a resource $k \in \mathcal{R}$ from activity $i \in V_0$ to activity $j \in V_*$ is represented by a directed arc $(i, j)_k$ from node i to node j. It should be noted that the resulting graph for a given resource flow $\mathcal{F}$ is a multigraph if more than one resource $k \in \mathcal{R}$ is transferred from activity i to activity j, i.e. there may be a total of r arcs $(i, j)_k$ connecting two activities i and j.

Similar to a disjunctive graph representing both, machine sequences as well as precedence constraints, Artigues et al. (2003) describe a so-called AON-flow network that incorporates the graph representing a feasible resource flow $\mathcal{F}$ as well as an activity-on-node network representing the precedence constraints $(i, j) \in A$ between activities. Now, each node $i \in V_{\text{all}}$ is weighted with the processing time p_i of activity i. In the following, we will generally visualize the activity-on-node network as well as resource flows for problem instances separately in order to represent these graphs more clearly. Also, if multiple resources $k \in \mathcal{R}$ are transferred from an activity $i \in V_0$ to an activity $j \in V_*$, only one arc with an appropriate label is displayed in the corresponding graph.

Next, we consider the relationship between feasible schedules S and feasible resource flows $\mathcal{F}$. In particular, we are interested in the following questions:

- How can a (feasible) schedule S be calculated for a given feasible resource flow $\mathcal{F}$?
- How can a (feasible) resource flow $\mathcal{F}$ be calculated for a given feasible schedule S?

Regarding the first question, it is possible to generate an earliest start schedule S (i.e. a schedule in which all activities start as early as possible with respect to the given precedence constraints $(i, j) \in A$ as well as the resource transfers f_{ijk}) based on a feasible resource flow $\mathcal{F}$ by calculating the lengths l_{ij} of the longest paths between the activities in the corresponding AON-flow network. This can be done, for example, by using the Floyd-Warshall algorithm (cf. Roy (1959), Floyd (1962), and Warshall (1962)). Then, each activity $i \in V_{\text{all}}$ is assigned the starting time $S_i = l_{0i}$ equal to the length of a longest path from dummy source activity 0 to activity i such that the length is equal to the sum of the weights of all nodes on the path excluding node i. An example visualizing an AON-flow network as well as a corresponding earliest start schedule is given in Example 3.3 (cf. Section 3.1.3).

Algorithm 6.1: BuildFlowFromSchedule(f, S)

```
Set $\beta_{ik} := r_{ik}$ for all $i \in V$ and $k \in \mathcal{R}$;
Set $f_{i,n+1,k} := r_{ik}$ for all $i \in V_0$ and $k \in \mathcal{R}$;
for $j := \pi_1^C$ to $\pi_n^C$ do
    for $i := \pi_0^C$ to $\pi_n^C$ do
        if $i \neq j$ then
            for $k := 1$ to $r$ do
                if $C_i \leq S_j$ then
                    $\rho := \min\{f_{i,n+1,k}, \beta_{jk}\}$;
                    $\beta_{jk} := \beta_{jk} - \rho$;
                    $f_{i,n+1,k} := f_{i,n+1,k} - \rho$;
                    $f_{ijk} := f_{ijk} + \rho$;
                end
            end
        end
    end
end
```

Next, we regard the second question of computing a resource flow $\mathcal{F}$ corresponding to a feasible schedule S. For this problem, Artigues et al. (2010) present a greedy algorithm (cf. Algorithm 6.1). In this algorithm, β_{ik} denotes the amount of resource units of resource $k \in \mathcal{R}$ that still has to be sent to activity $i \in V$. Additionally, all activities $i \in V_0$ are stored in a list $L^C = (\pi_0^C, \dots, \pi_n^C)$ sorted according to non-decreasing completion times C_i.

The general idea of this algorithm is that, initially, only outgoing resource transfers $f_{i,n+1,k} = r_{ik}$ from all activities $i \in V_0$ to the dummy sink activity $n+1$ are given. Then, in each iteration, the incoming resource transfers to an activity $j \in V$ (considered in the order of the list L^C) are selected from activities $i \in V_0$ (also considered in the order of the list L^C) that have been completed before the start of activity j (i.e. for activities i with $C_i \leq S_j$). For this algorithm, we now show that it is not necessarily able to generate a resource flow $\mathcal{F}$ corresponding to a feasible schedule S.

Example 6.1 We consider a project consisting of $n = 3$ real activities $i = 1$,2,3 as well as the two dummy activities 0 and 4. Furthermore, $r = 1$ renewable resource with a capacity $R_1 = 2$ is given. The processing times p_i as well as the resource requirements r_{i1} for all activities are displayed in Table 6.1. Finally, no precedence constraints are given between the real activities. A feasible schedule for this project is displayed in Figure 6.1.

i	0	1	2	3	4
p_i	0	1	1	3	0
r_{i1}	2	1	1	1	2

Table 6.1: Processing times p_i and resource requirements r_{ik} of the activities $i = 0, \dots, 4$ for the project considered in Example 6.1.

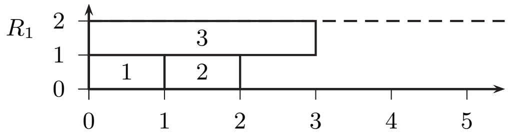
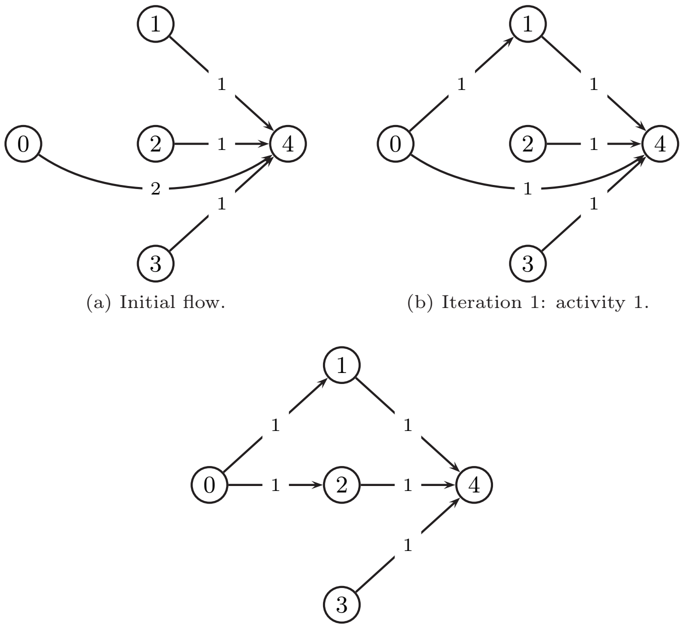


Figure 6.1: Feasible schedule for the project considered in Example 6.1.

In this schedule, the completion times of the activities are given by $C_0 = 0$, $C_1 = 1$, $C_2 = 2$, and $C_3 = 3$. As a result, the activity list $L^C = (0, 1, 2, 3)$

is calculated. Based on this list, Algorithm 6.1 tries to calculate a resource flow $\mathcal{F}$ as visualized in Figure 6.2.



(a) Initial flow. (b) Iteration 1: activity 1.

(c) Iteration 2: activity 2.

Figure 6.2: Calculation of a resource flow for the schedule from Figure 6.1 based on Algorithm 6.1. Starting with the initial flow displayed in (a), incoming resources transfers to activities 1 and 2 are calculated as displayed in (b) and (c), respectively.

Here, based on the initial flow, the incoming resource transfers to activity $\pi_1^C = 1$ are calculated first. For this, all activities $i \in V_0$ with $C_i \leq S_1$ are considered in the order of activity list L^C such that the amount of $r_{11} = 1$ unit of resource 1 required by activity 1 is transferred from activities i to activity 1. In order to redirect resource units of resource 1 from an activity $i \in V_0$ to activity 1, a resource transfer from activity i to dummy activity $n+1$ with $f_{i,n+1,1} > 0$ has to exist (i.e. there have to be resource units that actually can be redirected). In this case, one unit of resource 1 can be transferred from activity 0 to activity 1. Similarly, in the second iteration, one unit of resource 1 is redirected from activity 0 to activity $\pi_2^C = 2$.

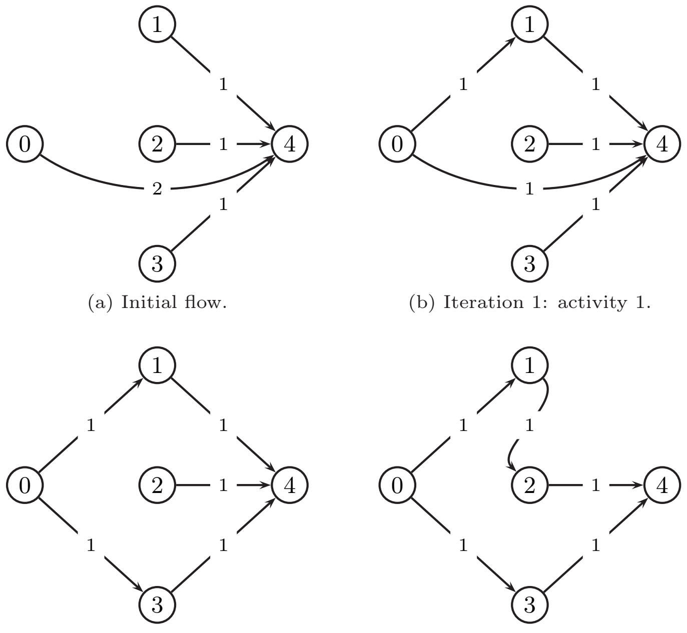
In the third iteration, however, when trying to redirect one resource unit of resource 1 to activity $\pi_3^C = 3$, no activity $i \in V_0$ with $C_i \leq S_3$ and $f_{i,n+1,1} > 0$ exists. Instead, both activities 1 and 2 are completed after the start of activity 3 (i.e. $S_1 > C_3$ as well as $S_2 > C_3$ hold) and no unit of resource 1 remains at activity 0 that could be redirected to activity 3 (i.e. $f_{0,n+1,1} = 0$ holds). □

As this example shows, the algorithm proposed by Artigues et al. (2010) is not able to generate a resource flow $\mathcal{F}$ from an arbitrary feasible schedule S. For this reason, we suggest an adaption of this algorithm using an additional list $L^S = (\pi_1^S, \ldots, \pi_n^S)$ of activities in which all activities $i \in V$ are sorted according to non-decreasing starting times S_i. Now, in line 3 of algorithm 6.1, we choose activity j from list L^S instead of from list L^C. The adapted algorithm is outlined in Algorithm 6.2.

Algorithm 6.2: Adapted BuildFlowFromSchedule(f, S)

```
1  Set $\beta_{ik} := r_{ik}$ for all $i \in V$ and $k \in \mathcal{R}$;
2  Set $f_{i,n+1,k} := r_{ik}$ for all $i \in V_0$ and $k \in \mathcal{R}$;
3  for $j := \pi_1^S$ to $\pi_n^S$ do
4      for $i := \pi_0^C$ to $\pi_n^C$ do
5          if $i \neq j$ then
6              for $k := 1$ to $r$ do
7                  if $C_i \leq S_j$ then
8                      $\rho := \min\{f_{i,n+1,k}, \beta_{jk}\}$;
9                      $\beta_{jk} := \beta_{jk} - \rho$;
10                     $f_{i,n+1,k} := f_{i,n+1,k} - \rho$;
11                     $f_{ijk} := f_{ijk} + \rho$;
12                 end
13             end
14         end
15     end
16 end
```

Example 6.2 We again consider the project from Example 6.1 with the feasible schedule displayed in Figure 6.1. Now, instead of using only the activity list $L^C = (0,1,2,3)$, we calculate the additional activity list $L^S = (1,3,2)$ in which the real activities $i = 1,2,3$ are sorted according to non-decreasing starting times (with $S_1 = 0$, $S_2 = 1$, and $S_3 = 0$). Based on these lists, Algorithm 6.2 calculates a resource flow $\mathcal{F}$ as visualized in Figure 6.3.


(a) Initial flow. (b) Iteration 1: activity 1.

(c) Iteration 2: activity 3. (d) Iteration 3: activity 2.

Figure 6.3: Calculation of a resource flow for the schedule from Figure 6.1 based on Algorithm 6.2. Starting with the initial flow displayed in (a), incoming resources transfers to activities 1, 3, and 2 are calculated as displayed in (b) to (d).

Here, the real activities $j \in V$ are considered in the order of activity list L^S. The incoming resource transfers are then selected as described in Example 6.1 by considering activities $i \in V_0$ with $C_i \leq S_j$ and $f_{i,n+1,1} > 0$ in the order of activity list L^C. The graph shown in Figure 6.3(d) is a feasible resource flow $\mathcal{F}$ corresponding to the given schedule S for this project. $\square$

Now, in Theorem 6.1, we prove the correctness of this algorithm.

Theorem 6.1 *For the RCPSP, Algorithm 6.2 always generates a corresponding resource flow $\mathcal{F}$ for any feasible schedule S.* □

Proof We consider the problem of generating a resource flow $\mathcal{F}$ corresponding to a feasible schedule S by using Algorithm 6.2. In each iteration, the algorithm selects an activity $j \in L^S$ with the smallest starting time S_j and redirects resource transfers for resources $k \in \mathcal{R}$ with $r_{jk} > 0$ from activities $i \in L^C$ to activity j. For this, an activity i with $f_{i,n+1,k} > 0$ and $C_i \leq S_j$ from the list L^C with the smallest completion time C_i is chosen.

Now, we assume that the algorithm is unable to select resource transfers for some resource $k \in \mathcal{R}$ required by an activity $j \in L^S$. In this case, $f_{i,n+1,k} = 0$ holds for all activities $i \in L^C$ with $C_i \leq S_j$. As the algorithm has only redirected resource units of resource k from activities $i \in L^C$ with $C_i \leq S_j$ to activities $j' \in L^S$ with $S_{j'} \leq S_j$ (i.e. no activity j' can receive resource units of resource k from activities $i \in L^C$ with $C_i > S_j$), not enough resource units of resource k are available to start processing activity j at starting time S_j. Thus, schedule S can not be feasible. This is a contradiction to the condition that S has to be a feasible schedule. ■

It should be noted that, while a unique earliest start schedule S can be calculated for each resource flow $\mathcal{F}$, multiple resource flows $\mathcal{F}$ might exist that represent the same feasible schedule S. For example, if multiple activities end at the same time t, it does not matter which resource units are transferred from these activities to the next activities starting at time t or later. On the other hand, if we do not restrict ourselves to earliest start schedules, infinitely many schedules correspond to a given resource flow $\mathcal{F}$, i.e. if subsets of activities are shifted to the right.

Next, in Theorem 6.2, we consider the relation between the starting times S'_i of activities $i \in V_{\text{all}}$ in the earliest start schedule S' that has been generated for the AON-flow network incorporating a resource flow $\mathcal{F}$ based on a feasible schedule S.

Theorem 6.2 *For the RCPSP, if S is a feasible schedule, $\mathcal{F}$ is a resource flow corresponding to this schedule as it is generated by Algorithm 6.2, and S' is the earliest start schedule corresponding to the AON-flow network incorporating resource flow $\mathcal{F}$, then the starting time S'_i of any activity $i \in V_{all}$ in schedule S' can not be later than the starting time S_i of activity i in schedule S (i.e. $S'_i \leq S_i$ holds).* □

Proof Let S be a feasible schedule and $\mathcal{F}$ a corresponding resource flow that has been generated by Algorithm 6.2. Furthermore, let S' be the earliest start schedule that has been computed for $\mathcal{F}$ based on the longest paths in the corresponding AON-flow network.

Now, we assume that $j \in V_*$ is the activity with the earliest starting time S'_j in schedule S' such that $S'_j > S_j$ holds, i.e. activity j starts later in schedule S' than it did in schedule S (where S_j is the starting time of activity j in schedule S). In this case, $S'_i \leq S_i$ holds for all other activities $i \in V_0$ with $S'_i < S'_j$. Then, becauseS' is an earliest start schedule, at least one resource transfer $f_{ijk} > 0$ of some resource $k \in \mathcal{R}$ from an activity $i \in V$ to activity j exists such that $C'_i > S_j$ holds, i.e. this resource transfer delays the start of activity j until after its original starting time S_j. Furthermore, because activity i has to start before activity j in schedule S' (i.e. $S'_i < S'_j$ holds), $C'_i \leq C_i$ has to hold for the completion time of activity i. As a result of this, $C_i > S_j$ holds for the starting and completion times of activities i and j in schedule S. This, however, is a contradiction to the condition that Algorithm 6.2 only selects resource transfers for a resource $k \in \mathcal{R}$ from activity $i \in L^C$ to activity $j \in L^S$ if $C_i \leq S_j$ holds. ■

Based on these results, we can now show that the set of schedules represented by resource flows always contains an optimal schedule.

Theorem 6.3 *For the RCPSP, an optimal schedule for a regular objective function is always contained in the set of schedules represented by resource flows.* □

Proof Let S be an optimal schedule. This schedule can now be transformed into a corresponding resource flow $\mathcal{F}$ based on Algorithm 6.2. For this resource flow $\mathcal{F}$, it is then possible to calculate a corresponding earliest start schedule S'. As shown in Theorem 6.2, the starting time S'_i of activity $i \in V_{\text{all}}$ in schedule S' can not be later than the starting time S_i of activity i in the optimal schedule S, i.e. $S'_i \leq S_i$ has to hold. Thus, for a regular objective function, schedule S' can not be worse than schedule S, i.e. schedule S' also has to be optimal. ■

Finally, it should be noted that the set of schedules represented by resource flows contains the set of semi-active (and hence active) schedules. This is due to the fact that each semi-active schedule can be represented as an earliest start schedule of a corresponding resource flow as it is generated by Algorithm 6.2. On the other hand, there are schedules represented by resource flows which are not semi-active (cf. Example 6.3).

Example 6.3 We again consider the project from Example 6.1. A feasible resource flow for this project is displayed in Figure 6.4 while the earliest start schedule corresponding to this resource flow is shown in Figure 6.5.

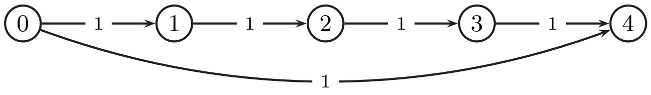


Figure 6.4: Feasible resource flow for the project considered in Example 6.3.

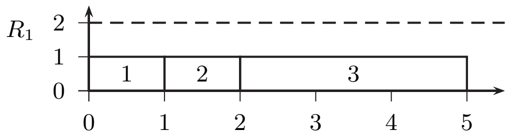


Figure 6.5: The earliest start schedule corresponding to the feasible resource flow displayed in Figure 6.4.

As can be seen here, this schedule is not a semi-active schedule as either activity 2 or activity 3 can still be shifted to the left. For this reason, the set of schedules represented by resource flows is a proper superset of the set of semi-active schedules. □

6.1.2 RCPSP with First-Tier Resource Transfers

Next, we consider the solution representation for the RCPSP with first-tier resource transfers. For this problem, the solution representation based on resource flows described above is adapted to include transfer times Δ_{ijk} denoting the amount of time required to transfer resource $k \in \mathcal{R}$ from activity $i \in V_0$ to activity $j \in V_*$. Then, in the AON-flow network corresponding to a feasible resource flow $\mathcal{F}$, each arc $(i,j)_k$ representing the resource transfer of resource k from activity i to activity j is assigned a weight equal to the corresponding transfer time Δ_{ijk}.

Apart from this adaption of the solution representation, no further modifications are required. Now, an earliest start schedule corresponding to a feasible resource flow $\mathcal{F}$ can again be calculated based on the longest paths in the respective AON-flow network. In this case, both, the processing times of the activities (i.e. the node weights) as well as the transfer times Δ_{ijk} (i.e. the arc weights) have to be taken into account. Here, if multiple arcs $(i,j)_k$ for resources $k \in \mathcal{R}$ connect two activities $i \in V_0$ and $j \in V_*$, an arc with the largest weight between the two activities is selected in order to calculate the longest paths.

Next, we again consider the problem of generating a resource flow $\mathcal{F}$ corresponding to a feasible schedule S. This problem can be modeled as a feasible flow problem which is an extension of the maximum flow problem (cf. Ahuja et al. (1993)). It should be noted that a similar approach is used by Artigues et al. (2010) for the RCPSP in order to show that a feasible resource flow exists for each feasible strict order P such that the makespan of the corresponding earliest start schedule is not larger than the makespan of the earliest start schedule induced by P. Here, a strict order P consists of additional precedence constraints such that inequality $S_i + p_i \leq S_j$ has to hold for all pairs of activities $(i,j) \in P$ in every schedule induced by the strict order P.

For a project consisting of n real activities $i = 1, \ldots, n$ as well as the two dummy activities 0 and $n+1$, a network consisting of $2 + 2n$ nodes is introduced for each resource $k \in \mathcal{R}$ such that dummy source activity 0 is represented by a supply node 0^{out} with a supply $b(0^{\text{out}}) = R_k$ while dummy sink activity $n+1$ is represented by a demand node $(n+1)^{\text{in}}$ with a demand $b((n+1)^{\text{in}}) = -R_k$. For the remaining activities $i = 1, \ldots, n$, two nodes i^{out} as well as i^{in} are introduced for each activity i. Here, demand node i^{in} represents the inflow of resource units of resource k to activity i with a demand $b(i^{\text{in}}) = -r_{ik}$ while supply node i^{out} represents the outflow of resource units of resource k from activity i with a supply $b(i^{\text{out}}) = r_{ik}$.

Next, a directed arc $(i^{\text{out}}, j^{\text{in}})$ is introduced between a supply node i^{out} and a demand node j^{in} if and only if the considered resource k can be transferred from activity $i \in V_0$ to activity $j \in V_*$, i.e. if inequality $S_i + p_i + \Delta_{ijk} \leq S_j$ holds for the starting times of activities i and j in schedule S. In this case, the arc $(i^{\text{out}},j^{\text{in}})$ is assigned a capacity $u_{i^{\text{out}},j^{\text{in}}} = \min\{r_{ik}, r_{jk}\}$ (i.e. the minimum of supply or demand of either node i^{out} or node j^{in}).

This feasible flow problem can then be transformed into a maximum flow problem by adding a global source node s as well as a global sink node t. Additionally, the global source node s is connected to each supply node i^{out} by a directed arc (s, i^{out}) with the capacity $u_{s,i^{\text{out}}} = b(i^{\text{out}})$ and each demand node j^{in} is connected to the global sink node t by a directed arc (j^{in}, t) with the capacity $u_{j^{\text{in}},t} = -b(j^{\text{in}})$. Now, a solution for the feasible flow problem exists if the maximum flow problem has a maximum flow from the global source s to the sink t with the value

$$f = \sum_{i \in V_0} b(i^{\text{out}}) = - \sum_{j \in V_*} b(j^{\text{in}}).$$

This maximum flow problem can then be solved, for example, by the polynomial time algorithm described by Ahuja et al. (1989).

Example 6.4 We again consider the project from Example 6.1 and extend it by transfer times Δ_{ijk} for resource $k = 1$ between all pairs of activities $i,j = 0, \ldots, 4$ as given in Table 6.2. A feasible schedule S for this project is displayed in Figure 6.4. It should be noted that the actual resource transfers are not shown in this schedule because a schedule S is usually only defined by the starting times S_i of activities $i \in V_{\text{all}}$.

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	3	2	0
2	0	3	0	1	0
3	0	2	1	0	0
4	0	0	0	0	0

Table 6.2: Transfer times Δ_{ijk} for resource $k = 1$ between all pairs of activities $i,j = 0, \ldots, 4$ for the project considered in Example 6.4.

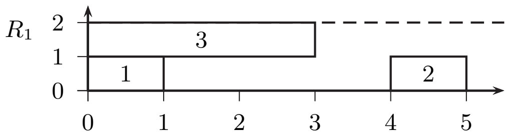


Figure 6.6: Feasible schedule for the project considered in Example 6.4.

Based on the starting times of the activities in this schedule, a network for the feasible flow problem can now be constructed for resource $k = 1$ as described above. The resulting network is displayed in Figure 6.7 while a feasible flow for this network is shown in Figure 6.8. Finally, the resource flow $\mathcal{F}$ corresponding to the feasible network flow from Figure 6.8 is displayed in Figure 6.9. This resource flow represents the feasible schedule S from Figure 6.6.

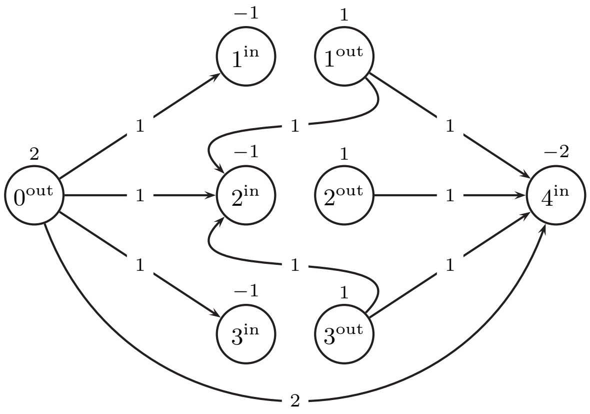


Figure 6.7: Network graph resulting from the starting times S_i of activities $i = 0, \ldots, 4$ in the feasible schedule displayed in Figure 6.6. The supply $b(i^{\text{out}})$ or demand $b(i^{\text{in}})$ of the nodes is given above the respective nodes while the capacities $u_{i^{\text{out}},j^{\text{in}}}$ of the arcs are given on the respective arcs.

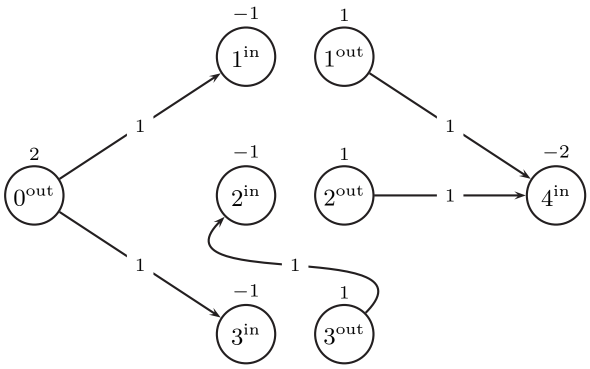


Figure 6.8: Feasible flow for the network displayed in Figure 6.7. The flow between the nodes is given on the respective arcs.

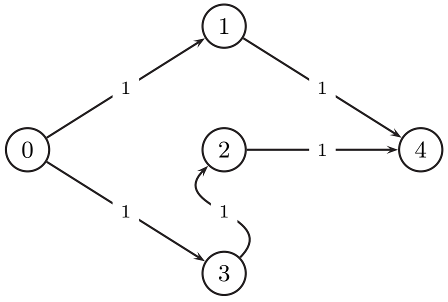


Figure 6.9: Resource flow corresponding to the feasible network flow displayed in Figure 6.8.

It should be noted that an alternative feasible flow for the flow network displayed in Figure 6.7 exists in which one unit of resource 1 is transferred from activity 1 to activity 2. This visualizes the fact that multiple resource flows might represent a given schedule S. On the other hand, both possible resource flows would result in different earliest start schedules, i.e. activity 2 would either start immediately after activity 1 (with makespan $C_{max} = 3$ or immediately after activity 3 (with makespan $C_{max} = 4$). □

Theorem 6.4 *For the RCPSP with first-tier resource transfers, the approach based on the feasible flow problem outlined above generates a corresponding resource flow $\mathcal{F}$ for any feasible schedule S.* □

Proof Let S be a feasible schedule for which a resource flow $\mathcal{F}$ is generated by calculating a solution for the corresponding feasible flow problem for each resource $k \in \mathcal{R}$ as described above.

First of all, we show that any solution of the feasible flow problem for a resource $k \in \mathcal{R}$ represents a feasible resource flow $\mathcal{F}_k$ for this resource. This follows directly from the model. In particular, the amount of incoming resource units of resource k to each activity $j \in V_*$ as well as the amount of outgoing resource units of resource k from each activity $i \in V_0$ has to be equal to the respective resource requirements because the demand $b(j^{\text{in}})$ or supply $b(i^{\text{out}})$ of the corresponding node j^{in} or i^{out} has to be satisfied in any feasible flow for the feasible flow problem. Furthermore, the resource flow $\mathcal{F}_k$ has to be acyclic and observe the given precedence constraints $(i, j) \in A$ because arcs $(i^{\text{out}}, j^{\text{in}})$ are only inserted if $S_i + p_i + \Delta_{ijk} \leq S_j$ holds for the starting times of activities $i \in V_0$ and $j \in V_*$ in schedule S. Thus, a feasible solution for the feasible flow problem represents a resource flow $\mathcal{F}_k$ for the corresponding resource $k \in \mathcal{R}$. Furthermore, any set consisting of exactly one feasible resource flow $\mathcal{F}_k$ for each resource $k \in \mathcal{R}$ constitutes a resource flow $\mathcal{F}$ corresponding to schedule S because resource transfers for resources $k \in \mathcal{R}$ can be chosen independently for the RCPSP with first-tier resource transfers (i.e. resource transfers of different resources $k \in \mathcal{R}$ do not influence each other).

Next, we show that it is possible to calculate a resource flow $\mathcal{F}$ as described above for any feasible schedule S. Here, we assume to the contrary that a feasible schedule S exists such that the corresponding feasible flow problem does not posses a feasible solution for some resource $k \in \mathcal{R}$. In this case, the supply $b(i^{\text{out}})$ of at least one node i^{out} corresponding to an activity $i \in V_0$ as well as the demand $b(j^{\text{in}})$ of at least one node j^{out} corresponding to an activity $j \in V_*$ can not be satisfied. This follows from the fact that the overall supply of the nodes always has to be equal to the overall demand of the nodes. Then, a subset of nodes j^{in} exists such that no flow exists in which the demands of all of these nodes can be fulfilled simultaneously, i.e. there is at least always one node j^{in} corresponding to an activity $j \in V_*$ to which not a sufficient amount of resource k can be transferred until its starting time S_j. This, however, implies that schedule S can not be feasible which is a contradiction to the condition that S has to be a feasible schedule.

Thus, it can be concluded that this approach is able to generate a corresponding resource flow $\mathcal{F}$ for any feasible schedule S for the RCPSP with first-tier resource transfers. ■

Now, using a similar argumentation as for Theorem 6.2, it can be shown that the starting time S'_i of any activity $i \in V_{\text{all}}$ in the earliest start schedule S' for a resource flow $\mathcal{F}$ corresponding to a feasible schedule S can not be later than the starting time S_i of activity i in schedule S. This property holds because arcs $(i^{\text{out}}, j^{\text{in}})$ for a resource $k \in \mathcal{R}$ are only inserted in this approach if $S_i + p_i + \Delta_{ijk} \leq S_j$ holds, i.e. if resource k can be transferred from activity i to activity j until its original starting time S_j.

Finally, the results from Theorem 6.3 as well as from Example 6.3 also hold for the RCPSP with first-tier resource transfers. In particular, the set of schedules represented by resource flows always contains an optimal solution for a regular objective function. This again follows from the results discussed above.

6.1.3 RCPSP with First- and Second-Tier Resource Transfers

Now, we consider the RCPSP with first- and second-tier resource transfers. For this problem, the solution representation has to be adapted to include both, first-tier and second-tier resource transfers as well as resource transfers of resources $k \in \mathcal{R}^{\text{sa}}$ from either the start or the end of activities $i \in V$.

Formally, this problem can be represented by a hypergraph such that each activity $i \in V_{\text{all}}$ is represented by two nodes i^{in} and i^{out} denoting the start and the end of the activity, respectively. Then, for each resource $k \in \mathcal{R}$ required to process an activity $i \in V_{\text{all}}$ (i.e. for each resource k with $r_{ik} > 0$), an arc $(i^{\text{in}},i^{\text{out}})_k$ is introduced between these two nodes denoting the execution of the activity by this resource such that a total of r_{ik} units of resource k are sent on the respective arc. In the following, these arcs are weighted with the processing time p_i of the corresponding activity i.

Next, we consider resource transfers of resources $l \in \mathcal{R}^{\text{ru}}$ from the end of activity $i \in V_0$ to the start of activity $j \in V_*$. In the hypergraph, these resource transfers $f_{ijl}^{FS} > 0$ are represented by directed arcs $(i^{\text{out}}, j^{\text{in}})_l$ that are weighted with the respective transfer time Δ_{ijl}. Additionally, if $\Delta_{ijl} > 0$ holds for the transfer time of resource l from activity i to activity j, second-tier resources $k \in \mathcal{R}^{\text{sa}}$ are required to support the transfer. In this case, if an amount of $f_{hijkl}^{FS} > 0$ units of second-tier resource $k \in \mathcal{R}^{\text{sa}}$ from the end of activity $h \in V_0$ is used to support the transfer of resource l from activity i to activity j, a directed arc $(h^{\text{out}}, i^{\text{out}}, j^{\text{in}})_{kl}$ is introduced. Similarly, if an amount of $f_{hijkl}^{SS} > 0$ units of second-tier resource $k \in \mathcal{R}^{\text{sa}}$ from the start of activity $h \in V_0$ is used to support the transfer of resource l from activity i to activity j, a directed arc $(h^{\text{in}}, i^{\text{out}}, j^{\text{in}})_{kl}$ is introduced. It should be noted that nodes h^{in}, h^{out}, and i^{out} are tails of the corresponding hyperarc while node j^{in} is the head of the corresponding hyperarc. In the following, these arcs are weighted with the transfer time $\Delta_{hik} + \Delta_{ijl}$ such that all three scenarios denoted by inequalities (5.30) to (5.32) are accounted for.

Finally, if an amount of $f_{ijk}^{FS} > 0$ units of (pure) first-tier resource $k \in \mathcal{R}^{\text{sa}}$ is transferred from the end of activity $i \in V_0$ to the start of activity $j \in V_*$, an arc $(i^{\text{out}}, j^{\text{in}})_k$ is introduced. Similarly, an arc $(i^{\text{in}}, j^{\text{in}})_k$ is introduced if an amount of $f_{ijk}^{SS} > 0$ units of resource k is transferred from the start of activity i to activity j. In both cases, the respective arc is weighted with the corresponding transfer time Δ_{ijk}. These two types of arcs represent the two scenarios denoted by inequalities (5.28) and (5.29).

Now, all first- and second-tier resource transfers of a resource $k \in \mathcal{R}$ again denote a resource flow $\mathcal{F}_k$ for the corresponding resource k while a resource flow $\mathcal{F}$ for a given project consists of exactly one resource flow $\mathcal{F}_k$ for each resource $k \in \mathcal{R}$. Finally, we have to adapt the conditions any feasible resource flow $\mathcal{F}$ has to satisfy for the RCPSP with first- and second-tier resource transfers. In particular, the amount of incoming resource units of a resource $k \in \mathcal{R}^{\text{sa}}$ at a node j^{in} might be larger than the resource requirements r_{jk} of activity j if pure second-tier resource units are required to support the transfer of first-tier resources $l \in \mathcal{R}^{\text{ru}}$ to activity j.

As described on page 126, the actual amount of resource units of pure second-tier resource $k \in \mathcal{R}^{\text{sa}}$ that has to be transferred to an activity $j \in V$ can be calculated based on the selected resource transfers of resources $l \in \mathcal{R}^{\text{ru}}$ to activity j (cf. Theorem 5.1). Here, for each resource transfer $f_{ijl}^{FS} > 0$ of a resource $l \in \mathcal{R}^{\text{ru}}$ from activity $i \in V_0$ to activity $j \in V_*$ with $\Delta_{ijl} > 0$, an amount of $\mu_{kl} \cdot f_{ijl}$ units of resource $k \in \mathcal{R}^{\text{ru}}$ from activities $h \in V_0$ are required to support this transfer as second-tier resources (as denoted by resource transfers f_{hijkl}^{FS} and f_{hijkl}^{SS}). These support requirements have to be satisfied in any feasible resource flow $\mathcal{F}$. Now, if the requirement of such second-tier resource units is larger than the actual resource requirement r_{jk} of activity j, the surplus of resource units of second-tier resource k have to be transferred as pure second-tier resource units to node j^{in}. Additionally, they also have to be transferred from node j^{in} to the next activities by start-to-start transfer in order to maintain flow conservation. Apart from these adaptions, the remaining conditions a feasible resource flow has to satisfy remain the same.

Example 6.5 We consider a project consisting of $n = 3$ real activities $i = 1{,}2{,}3$ as well as the two dummy activities 0 and 4. Furthermore, $r = 2$ renewable resources with capacities $R_k = 2$ for $k = 1{,}2$ are available. Now, an amount of $\mu_{12} = 1$ unit of resource 1 is required to support the transfer of one unit of resource 2. The processing times p_i and resource requirements r_{ik} as well as the transfer times Δ_{ijk} with $\Delta_{ijk} = \Delta_{ij}$ for both resources $k = 1{,}2$ are given in Table 6.3. Finally, no precedence constraints $i \to j$ are given between the real activities.

A feasible resource flow $\mathcal{F}$ for this project is visualized as a hypergraph in Figure 6.10. In this graph, each activity $i \in V_{\text{all}}$ is represented by two nodes i^{in} and i^{out}. These two nodes are connected by arcs $(i^{\text{in}}, i^{\text{out}})_k$ for resources $k \in \mathcal{R}$ with $r_{ik} > 0$ that represent the resource requirements of activity i. It should be noted that one unit of pure second-tier resource

i	0	1	2	3	4
p_i	0	1	1	1	0
r_{i1}	2	2	0	2	2
r_{i2}	2	1	1	0	2

(a) Activity parameters.

	0	1	2	3	4
0	0	1	1	2	0
1	1	0	1	1	0
2	1	1	0	1	0
3	2	1	1	0	0
4	0	0	0	0	0

(b) Transfer times.

Table 6.3: The processing times p_i as well as resource requirements r_{ik} are displayed in (a) while the transfer times $\Delta_{ijk} = \Delta_{ij}$ for both resources $k = 1{,}2$ are displayed in (b).

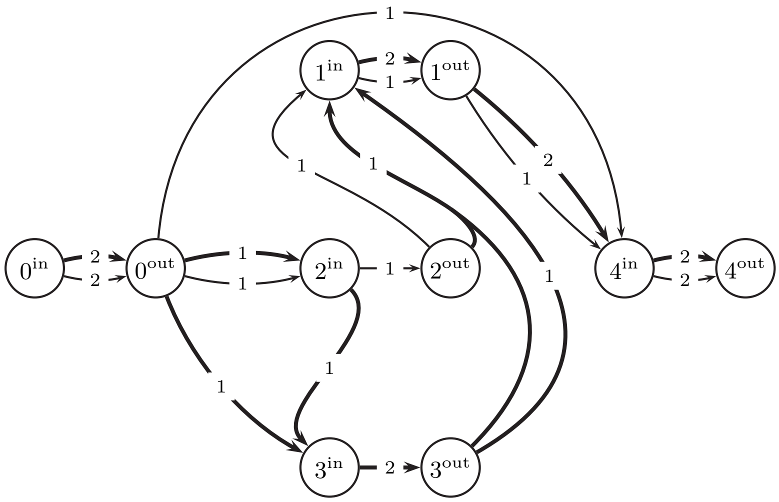


Figure 6.10: Feasible resource flow $\mathcal{F}$ for the project from Example 6.5. In this hypergraph, resource transfers of resource 1 are visualized by thick arcs while resource transfers of resource 2 are visualized by thin arcs.

1 is transferred from the end of activity 0 to the start of activity 2 and is immediately sent from there to activity 3 (i.e. is is not used to process activity 2). Also, the transfer of first-tier resource 2 from the end of activity

2 to activity 1 is modeled by an arc $(2^{\text{out}}, 1^{\text{in}})_2$ representing the actual transfer of resource 2 as well as an hyperarc $(3^{\text{out}}, 2^{\text{out}}, 1^{\text{in}})_{12}$ representing the transfer of a supporting resource unit of resource 1 from the end of activity 3.

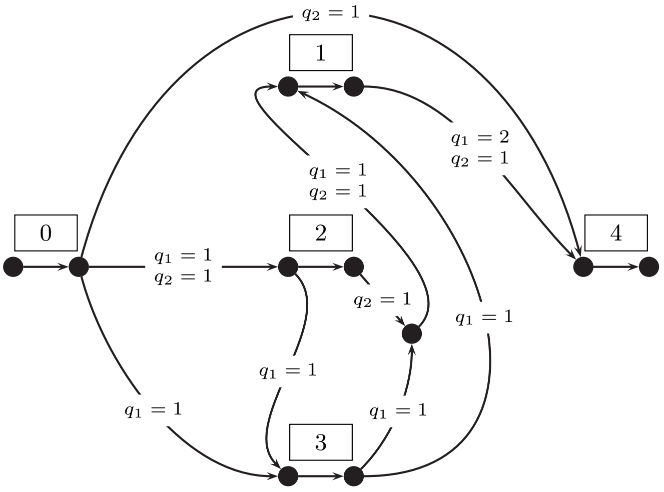


Figure 6.11: Alternative representation of the feasible resource flow $\mathcal{F}$.

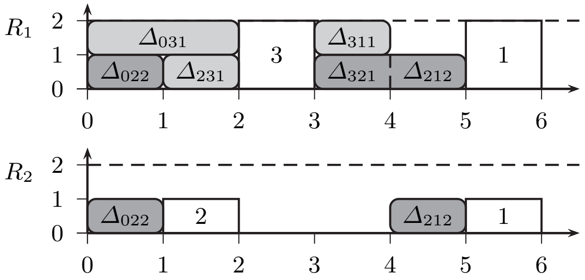


Figure 6.12: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}$ displayed in Figure 6.10.

Now, resource flow $\mathcal{F}$ is visualized again in Figure 6.11 using the same network representation as it has been used before in this thesis. In the following, we will use this representation as it represents resource flows more clearly. Finally, the earliest start schedule corresponding to this feasible resource flow $\mathcal{F}$ is displayed in Figure 6.12. □

Similar to the other two problems considered in this chapter, it is again possible to calculate an earliest start schedule for a given feasible resource flow $\mathcal{F}$ based on the longest paths. In order to do this, the AON-flow network for a given resource flow $\mathcal{F}$ (represented as a hypergraph as described above) is considered such that each precedence constraint $(i, j) \in A$ between two activities $i \in V_0$ and $j \in V_*$ is represented by an arc $(i^{\text{out}}, j^{\text{in}})$. The initial distances can then be determined from the arc weights. It should be noted that a hyperarc $(h^{\text{in}}, i^{\text{out}}, j^{\text{in}})_{kl}$ (or $(h^{\text{out}}, i^{\text{out}}, j^{\text{in}})_{kl}$) is only used to calculate the initial distances between nodes h^{in} (or h^{out}) and j^{in} and not between nodes i^{out} and j^{in}. Instead, the distance between these two nodes is calculated based on the transfer of the corresponding first-tier resource $l \in \mathcal{R}^{\text{ru}}$ as it is represented by an arc $(i^{\text{out}}, j^{\text{in}})_l$. The starting time S_i of activity $i \in V_{\text{all}}$ is then given by the length of a longest path from the start of activity 0 (represented by node 0^{in}) to the start of activity i (represented by node i^{in}), i.e. it is set to $S_i = l_{0^{\text{in}} i^{\text{in}}}$.

Next, we again consider the problem of generating a resource flow $\mathcal{F}$ for a feasible schedule S. Unlike for the RCPSP with first-tier resource transfers, the resource transfers of resources $k \in \mathcal{R}$ can not be selected independently here because a different amount of supporting second-tier resources $k \in \mathcal{R}^{\text{sa}}$ might be required based on the selection of resource transfers of resources $l \in \mathcal{R}^{\text{ru}}$. At the same time, the arrival time of the selected first-tier resources $l \in \mathcal{R}^{\text{ru}}$ at activity $j \in V_*$ depends on the second-tier resources $k \in \mathcal{R}^{\text{sa}}$ assigned to support the transfer. For this reason, we adapt the approach based on the feasible flow problem as it has been introduced in Section 6.1.2 for the RCPSP with first-tier resource transfers as follows.

First of all, resource flows $\mathcal{F}_l$ are calculated for all resources $l \in \mathcal{R}^{\text{ru}}$ based on the feasible flow problem as described on page 156. For this, each activity $i \in V$ is represented by two nodes i^{in} and i^{out} and arcs $(i^{\text{out}}, j^{\text{in}})$ are inserted between two nodes i^{out} and j^{in} if it is possible to transfer resource $l \in \mathcal{R}^{\text{ru}}$ from activity $i \in V_0$ to activity $j \in V_*$ such that $S_i + p_i + \Delta_{ijl} \leq S_j$ holds for the starting times S_i and S_j of the activities in the feasible schedule S.

Based on these resource flows $\mathcal{F}_l$ it is then possible to calculate the actual amount of resource $k \in \mathcal{R}^{\text{sa}}$ required to support the transfer of resources

$l \in \mathcal{R}^{\text{ru}}$ (cf. Theorem 5.1). Now, the problem of calculating resource flows $\mathcal{F}_k$ for resources $k \in \mathcal{R}^{\text{sa}}$ can be tackled. For this, the dummy source activity 0 is again represented by a single node 0^{out}_{FS} with a supply $b(0^{\text{out}}_{FS}) = R_k$. On the other hand, the dummy sink activity $n+1$ may be represented by multiple nodes such that, for each resource transfer $f^{FS}_{i,n+1,l} > 0$ of a resource $l \in \mathcal{R}^{\text{ru}}$ from an activity $i \in V_0$ with a transfer time $\Delta_{i,n+1,l} > 0$, one node $(n+1)^{\text{in}}_{il}$ with a demand $b((n+1)^{\text{in}}_{il}) = -\mu_{kl} \cdot f^{FS}_{i,n+1,l}$ is introduced. These nodes represent the amount of resource units of second-tier resource k required to support the transfer of these $f^{FS}_{i,n+1,l}$ units of first-tier resource l to activity $n+1$. Additionally, a node $(n+1)^{\text{in}}$ with a demand

$$b((n+1)^{\text{in}}) = -\max\left\{0, R_k - \sum_{i \in V_0} \sum_{l \in \mathcal{R}^{\text{ru}}} \mu_{kl} \cdot f^{FS}_{i,n+1,l} \cdot z_{i,n+1,l}\right\}$$

is introduced where $z_{i,n+1,l} = 1$ denotes that the transfer of resource l from activity i to activity $n+1$ has a transfer time $\Delta_{i,n+1,l} > 0$. This node represents the transfer of all units of resource k that are not required to support the transfer of first-tier resources $l \in \mathcal{R}^{\text{ru}}$ to activity $n+1$. Similarly, for each activity $j \in V$, a node j^{in}_{il} with a demand $b(j^{\text{in}}_{il}) = -\mu_{kl} \cdot f^{FS}_{ijl}$ is introduced for each resource transfer $f^{FS}_{ijl} > 0$ of a resource $l \in \mathcal{R}^{\text{ru}}$ from activity $i \in V_0$ to activity j with $\Delta_{ijl} > 0$ as well as a node j^{in} with a demand

$$b(j^{\text{in}}) = -\max\left\{0, r_{jk} - \sum_{i \in V_0} \sum_{l \in \mathcal{R}^{\text{ru}}} \mu_{kl} \cdot f^{FS}_{ijl} \cdot z_{ijl}\right\}$$

where $z_{ijl} = 1$ again denotes that the transfer of resource l from activity i to activity j requires a transfer time $\Delta_{ijl} > 0$. Additionally, a node j^{out}_{FS} with a supply $b(j^{\text{out}}_{FS}) = r_{jk}$ as well as a node j^{out}_{SS} with a supply

$$b(j^{\text{out}}_{SS}) = \max\left\{0, \sum_{i \in V_0} \sum_{l \in \mathcal{R}^{\text{ru}}} \mu_{kl} \cdot f^{FS}_{ijl} \cdot z_{ijl} - r_{jk}\right\}$$

equal to the amount of resource units of pure second-tier resource k transferred to activity j are introduced for each activity $j \in V$. It should be noted that all supply as well as demand nodes with a supply or demand equal to zero can be omitted from the graph.

Finally, an arc $(h^{\text{out}}_{FS}, j^{\text{in}})$ is inserted if $S_h + p_h + \Delta_{hjk} \leq S_j$ holds between the starting time S_h of activity $h \in V_0$ and the starting time S_j of activity $j \in V_*$ while an arc $(h^{\text{out}}_{SS}, j^{\text{in}})$ is inserted if $S_h + \Delta_{hjk} \leq S_j$ holds. These arcs represent the direct transfer of resource k from activity h to activity j by either finish-to-start or start-to-start transfer. Similarly, an arc $(h^{\text{out}}_{FS}, j^{\text{in}}_{il})$ for $i \in V_0$ and $l \in \mathcal{R}^{\text{ru}}$ is inserted if $S_h + p_h + \Delta_{hik} + \Delta_{ijl} \leq S_j$ holds while an arc $(h^{\text{out}}_{SS}, j^{\text{in}}_{il})$ is inserted if $S_h + \Delta_{hik} + \Delta_{ijl} \leq S_j$ holds. These arcs represent the transfer of second-tier resource k from activity h to support the transfer of first-tier resource l from activity i to activity j by either finish-to-start or start-to-start transfer. For all of these arcs (e, f) between supply nodes $e \in \{h^{\text{out}}_{SS}, h^{\text{out}}_{FS}\}$ (with $h \in V_0$) and demand nodes $f \in \{j^{\text{in}}, j^{\text{in}}_{il}\}$ (with $j \in V_*$, $i \in V_0$ and $l \in \mathcal{R}^{\text{ru}}$), the capacity is set to $u(e, f) = \min\{b(e), -b(f)\}$, i.e. it is set to the minimum of supply or demand required by either node e or node f.

This feasible flow problem can now be transformed into a maximum flow problem as described in Section 6.1.2. It should be noted, however, that no solution may exist for the feasible flow problem corresponding to some resource $k \in \mathcal{R}^{\text{sa}}$ based on a given set of resource flows $\mathcal{F}_l$ for all resources $l \in \mathcal{R}^{\text{ru}}$. For this reason, all possible sets of resource flows $\mathcal{F}_l$ for resources $l \in \mathcal{R}^{\text{ru}}$ have to be enumerated until it is possible to generate resource flows $\mathcal{F}_k$ for all resources $k \in \mathcal{R}^{\text{sa}}$.

Example 6.6 We again consider the project from Example 6.5. A schedule for this project with the starting times $S_0 = 0$, $S_1 = 5$, $S_2 = 1$, $S_3 = 2$, and $S_4 = 6$ is displayed in Figure 6.12. Based on these starting times it is now possible to calculate a flow network for the corresponding feasible flow problem for resource 2 as displayed in Figure 6.13. A feasible flow for this network is then shown in Figure 6.14.

Based on the corresponding resource flow $\mathcal{F}_2$, we can now calculate the flow network for the feasible flow problem for resource 1 as shown in Figure 6.15. In this network, for example, node 1^{in}_{02} ensures that an amount of $\mu_{12} = 1$ unit of resource 1 has to support the transfer of resource 2 from activity 0 to activity 1 (i.e. based on the corresponding resource transfer selected in Figure 6.14) while node 1^{in} ensures that a sufficient amount of resource 1 has to be transferred to activity 1 to satisfy its resource requirements $r_{11} = 2$ (i.e. one additional unit of resource 1 has to be transferred to activity 1 as a first-tier resource). Also, it should be noted that one unit of resource 1 can be transferred from activity 2 by start-to-start transfer. This is modeled by node 2^{out}_{SS} in this network.

It is easy to see that no feasible flow exists for this flow network such that the demands of all nodes 1^{in}, 1^{in}_{02}, 2^{in}_{02}, and 3^{in} can be satisfied simultaneously. In particular, these nodes have an overall demand of 5 resource units but

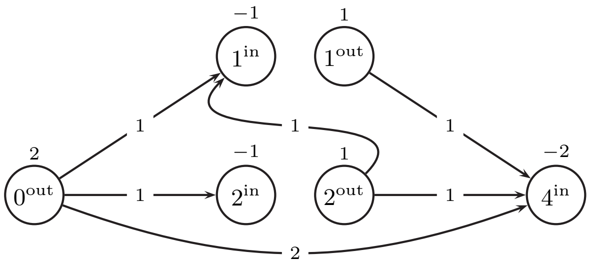


Figure 6.13: Flow network for resource 2 resulting from the starting times S_i of the activities $i = 0, \ldots, 4$ in the given schedule displayed in Figure 6.12. Activity 3 is missing in this graph because it does not require resource 2 (i.e. $r_{32} = 0$ holds).

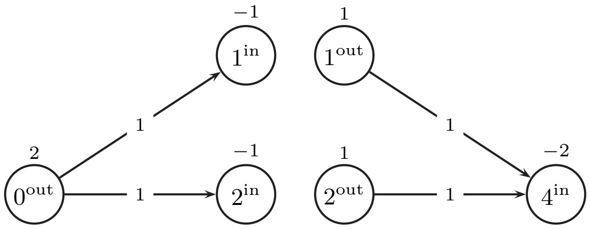


Figure 6.14: Feasible flow for the network displayed in Figure 6.13.

can only be supplied a total of 4 resource units by nodes 0^{out}_{FS}, 2^{out}_{SS}, and 3^{out}_{FS}. At the same time, it is also not possible for node 4^{in} to consume the total supply of 3 resource units from nodes 1^{out}_{FS} and 3^{out}_{FS} (i.e. these resource units can not be sent to any other node). For this reason, we calculate an alternative feasible flow for resource 2 based on the flow network from Figure 6.13 as displayed in Figure 6.16.

Now, the flow network displayed in Figure 6.17 for the feasible flow problem for resource 1 can be calculated based on the resource flow $\mathcal{F}_2$ corresponding to this alternative feasible flow. A feasible flow for this resulting flow network can then be calculated because one unit of resource 1 can now be sent from node 3^{out}_{FS} to node 1^{in}_{22} such that all supplies and demands of the nodes are satisfied (cf. Figure 6.18).

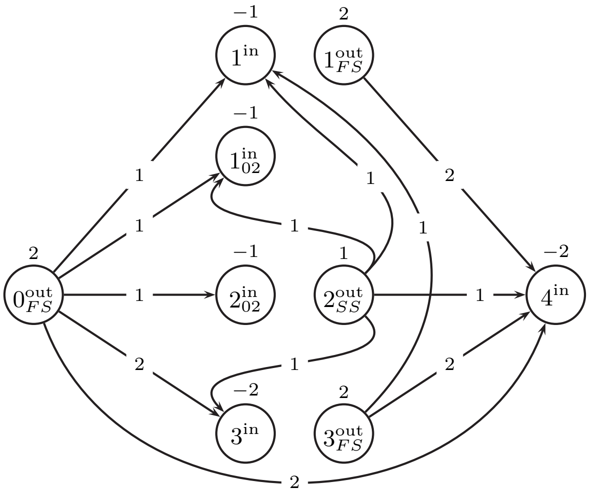


Figure 6.15: Network graph for resource 1 resulting from the starting times S_i of activities $i = 0, \ldots, 4$ in the given schedule displayed in Figure 6.12 as well as the resource flow $\mathcal{F}_2$ corresponding to the feasible flow shown in Figure 6.14.

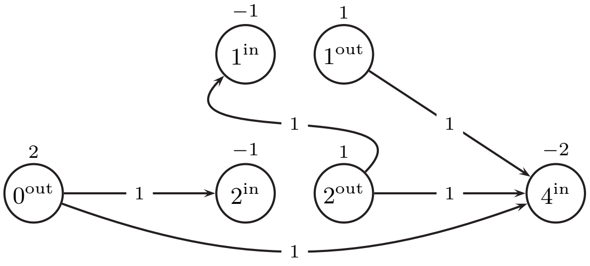


Figure 6.16: Alternative flow for the network displayed in Figure 6.13.

It should be noted that the two resource flows $\mathcal{F}_1$ and $\mathcal{F}_2$ resulting from the feasible flows displayed in Figures 6.16 and 6.18 constitute a resource flow $\mathcal{F}$ corresponding to the feasible schedule S from Figure 6.12. This resource flow $\mathcal{F}$ is visualized in Figures 6.10 and 6.11. $\square$

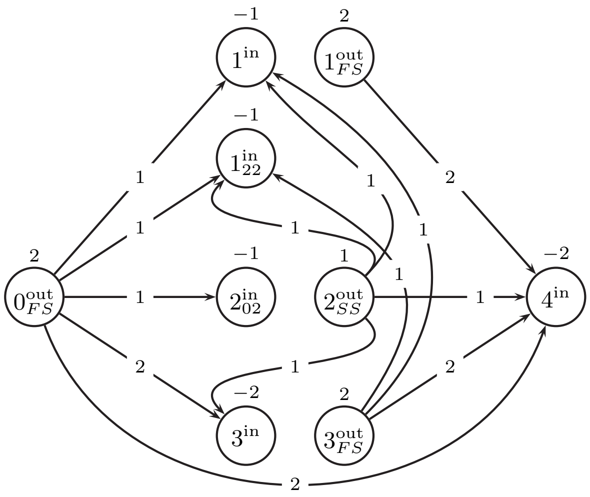


Figure 6.17: Network graph for resource 1 resulting from the resource flow $\mathcal{F}_2$ corresponding to the feasible flow shown in Figure 6.16.

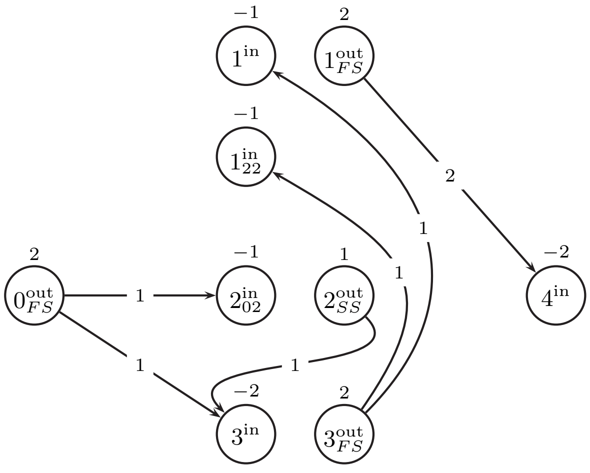


Figure 6.18: Feasible flow for the network displayed in Figure 6.17.

Below, we prove the correctness of this approach in Theorem 6.5.

Theorem 6.5 *For the RCPSP with first- and second-tier resource transfers, the approach based on the feasible flow problem always generates a corresponding resource flow $\mathcal{F}$ for any feasible schedule S.* □

Proof First of all, it can be shown as in Theorem 6.4 that it is always possible to generate resource flows $\mathcal{F}_l$ for resources $l \in \mathcal{R}^{\text{ru}}$ for any feasible schedule S.

Thus, it remains to be shown that it is also possible to generate resource flows $\mathcal{F}_k$ for all resources $k \in \mathcal{R}^{\text{sa}}$ based on schedule S as well as on at least one set of feasible resource flows $\mathcal{F}_l$ for all resources $l \in \mathcal{R}^{\text{ru}}$ as they have been calculated before. We assume to the contrary that no set of resource flows $\mathcal{F}_l$ can be generated such that it is possible to calculate feasible flows for all resources $k \in \mathcal{R}^{\text{sa}}$ for a given feasible schedule S. In this case, there is always at least one resource $k \in \mathcal{R}^{\text{sa}}$ for which no feasible flow can be generated, i.e. at least the demand of one demand node as well as the supply of one supply node can not be satisfied. Similar to Theorem 6.4, this can only occur if not a sufficient amount of resource k can be transferred to an activity $j \in V_*$ until its starting time S_j (or not a sufficient amount of resource k can support the transfer of resources $l \in \mathcal{R}^{\text{ru}}$ to activity j until time S_j), i.e. schedule S can not be feasible. This, however, contradicts the condition that schedule S is feasible.

As a result, it is always possible to calculate feasible flows for all resources $k \in \mathcal{R}^{\text{sa}}$ for at least one set of resource flows $\mathcal{F}_l$. These flows correspond to resource flows $\mathcal{F}_k$ which again follows from the model as all conditions for a feasible resource flow are also fulfilled in any solution for the corresponding feasible flow problem. Both, resource flows $\mathcal{F}_l$ for all resources $l \in \mathcal{R}^{\text{ru}}$ as well as resource flows $\mathcal{F}_k$ for all resources $k \in \mathcal{R}^{\text{sa}}$ then constitute a resource flow $\mathcal{F}$ corresponding to the feasible schedule S. This again follows from the fact that resource flows for resources $l \in \mathcal{R}^{\text{ru}}$ as well as resource flows for resources $k \in \mathcal{R}^{\text{sa}}$ can be generated independently because resource transfers of different resources do not interfere with each other and all support requirements for the selected resource transfers of resources $l \in \mathcal{R}^{\text{ru}}$ are satisfied in the resource flows for resources $k \in \mathcal{R}^{\text{sa}}$. ■

It should be noted that this approach is not a polynomial time algorithm because all possible sets of resource flows $\mathcal{F}_l$ for resources $l \in \mathcal{R}^{\text{ru}}$ may have to be evaluated in order to generate a resource flow $\mathcal{F}$ for a given feasible schedule S. Due to Theorem 4.1, already selecting resource transfers as well as assigning supporting resources for one activity such as to ensure that all resource units arrive until the starting time of the activity in the feasible schedule is NP-hard (this problem corresponds to the decision version of the problem considered in Theorem 4.1 with a given upper bound T). For this reason, this problem can not be any easier because the same problem has to be solved for all activities $i \in V_{\text{all}}$ simultaneously. This has no further practical implications, however, because we only require these results for the following conclusions.

First of all, similar to Theorem 6.2, it can be shown that the starting times S'_j of activities $j \in V_{\text{all}}$ in schedule S' can not be later than the starting times S_j of activities j in schedule S where schedule S is the original schedule and schedule S' is the earliest start schedule corresponding to the resource flow $\mathcal{F}$ that has been calculated for schedule S based on the approach outlined above. This again follows from the model as a resource transfer of resource $k \in \mathcal{R}$ from activity $i \in V_0$ to activity j can only be selected if the resource units can arrive at activity j until its original starting time S_j in schedule S. Here, all five scenarios as modeled by inequalities (5.28) to (5.32) are accounted for.

Finally, based on these results, it can be shown that the results from Theorem 6.3 also hold for the RCPSP with first- and second-tier resource transfers, i.e. the set of schedules represented by resource flows always contains an optimal schedule for a regular objective function.

6.2 Neighborhoods

In this section, neighborhoods for the three problems considered in this chapter are presented. Based on the solution representation introduced in the previous section, these neighborhoods are defined on solutions represented by resource flows. Before these neighborhoods are introduced, however, Section 6.2.1 gives a short overview of modifications defined on resource flows that have previously been described in literature. Afterward, neighborhoods for the classical RCPSP are introduced in Section 6.2.2. These are then extended to the RCPSP with first-tier resource transfers in Section 6.2.3 as well as the RCPSP with first- and second-tier resource transfers in Section 6.2.4. Apart from a description of these neighborhoods, some theoretical results related to these neighborhoods are reported as well. In particular, we consider the connectivity of these neighborhoods.

6.2.1 Previous Work

One of the first papers dealing with neighborhoods defined on solutions that are represented by resource flows for the classical RCPSP has been published by Fortemps and Hapke (1997). In this paper, they describe an extension of a neighborhood in which solutions are represented as disjunctive graphs as it has been introduced by van Laarhoven et al. (1992) for the job-shop scheduling problem. Here, while van Laarhoven et al. (1992) define modifications that reverse critical arcs in disjunctive graphs in order to change the sequence in which operations are processed, Fortemps and Hapke (1997) describe two types of modifications that redirect specific amounts of resource units between activities in resource flows in order to change the sequence in which activities are processed. These two types of modifications are referred to as parallel and serial modifications and are described below.

First of all, for a serial modification, two activities $i \in V$ and $j \in V$ are selected in a resource flow $\mathcal{F}$ such that at least one arc $(i, j)_k$ for a resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ exists between these activities and the activities lie on a critical path in the graph representing the resource flow. Furthermore, groups U_k with $f_{uik} > 0$ for all activities $u \in U_k$ as well as groups V_k with $f_{jvk} > 0$ for all activities $v \in V_k$ have to be selected for each resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ such that

$$\sum_{u \in U_k} f_{uik} \geq f_{ijk} \qquad \text{and} \qquad \sum_{v \in V_k} f_{jvk} \geq f_{ijk}$$

hold for these groups of activities, respectively. Then, a serial modification first reverses the direction of all arcs $(i, j)_k$ for resources $k \in \mathcal{R}$ with $f_{ijk} > 0$ between activities i and j such that an amount of $f'_{jik} = f_{ijk}$ units of resource $k \in \mathcal{R}$ is transferred from activity j to activity i in the resulting resource flow $\mathcal{F}'$. Additionally, an amount of f_{ijk} units of each resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ has to be redirected from activities $u \in U_k$ to activity j as well as from activity i to activities $v \in V_k$ in order to maintain flow conservation in the resulting resource flow $\mathcal{F}'$. It should be noted that Fortemps and Hapke (1997) do not describe how the sets U_k and V_k are selected or how the resource units are redirected. Instead, they focus on the trivial case that a single activity $u \in V_0$ with $f_{uik} \geq f_{ijk}$ as well as a single activity $v \in V_*$ with $f_{jvk} \geq f_{ijk}$ can be selected for each resource $k \in \mathcal{R}$ with $f_{ijk} > 0$.

In this case, an amount of $f'_{ujk} = f_{ujk} + f_{ijk}$ units of resource k are redirected from activity u to activity j in the resulting resource flow $\mathcal{F}'$ while $f'_{uik} =$

$f_{uik} - f_{ijk}$ units of resource k are still transferred from activity u to activity i. Similarly, an amount of $f'_{ivk} = f_{ivk} + f_{ijk}$ units of resource k are redirected from activity i to activity v while $f'_{jvk} = f_{jvk} - f_{ijk}$ units of resource k are still transferred from activity j to activity v. Even for this trivial case, however, Fortemps and Hapke (1997) do not describe how activities u and v are selected.

It should be noted that this serial modification closely resembles the neighborhood for the job-shop scheduling problem based on disjunctive graphs as it has been introduced by van Laarhoven et al. (1992). Because resources for the classical RCPSP do not necessarily have unit capacity (i.e. unlike for the job-shop scheduling problem, they are not necessarily disjunctive resources), it is not sufficient to only change the sequence in which activities are processed by the same resource units. For this reason, Fortemps and Hapke (1997) propose a second modification, referred to as parallel modification, that changes the resource plan for a selected resource $k \in \mathcal{R}$. These modifications can be used, for example, to allow different activities to be processed in parallel by the same type of resource if a sufficient amount of resource units of this resource is available.

Now, for a parallel modification, two arcs $(i{,}j)_k$ and $(u{,}v)_k$ for activities $i, u \in V_0$ and $j, v \in V_*$ representing resource transfers $f_{ijk} > 0$ and $f_{uvk} > 0$ of a resource $k \in \mathcal{R}$ are selected in a resource flow $\mathcal{F}$ such that no directed path exists between activities j and u or between activities v and i. Then, an amount of $q = \min\{f_{ijk}, f_{uvk}\}$ units of resource k is rerouted from activity i to activity v as well as from activity u to activity j. This results in a resource flow $\mathcal{F}'$ with the modified resource transfers $f'_{ijk} = f_{ijk} - q$, $f'_{uvk} = f_{uvk} - q$, $f'_{ivk} = f_{ivk} + q$, and $f'_{ujk} = f_{ujk} + q$ between the activities.

Apart from introducing these two types of modifications, Fortemps and Hapke (1997) do not report any further results (neither experimental not theoretical) regarding this neighborhood. To the best of our knowledge, neither has any additional literature been published until now that deals with this neighborhood any further.

Finally, another neighborhood for the classical RCPSP based on resource flows has been introduced by Artigues et al. (2003). This neighborhood is based on modifications that first remove an activity from a given resource flow $\mathcal{F}$ and then re-insert it. Here, resource transfers to and from this activity have to be redirected both, when removing the activity as well as when re-inserting it. For this neighborhood, Artigues et al. (2003) introduce a polynomial time insertion algorithm that inserts activities in an optimal position in relation to the current resource flow.

6.2.2 Classical RCPSP

In this section, we introduce neighborhoods for the RCPSP based on the parallel as well as serial modifications introduced by Fortemps and Hapke (1997). We focus on these modifications because they can more easily be adapted to the RCPSP with first- and second-tier resource transfers than, for example, the neighborhood introduced by Artigues et al. (2003) (cf. Section 6.2.4). In particular, in order to re-insert an activity into a resource flow, it is necessary to redirect incoming resource transfers to this activity as well as outgoing resource transfers from this activity. While Artigues et al. (2003) present a polynomial-time algorithm to find an optimal insertion position for the classical RCPSP, the same problem for the RCPSP with first- and second-tier resource transfers is NP-hard. This again follows from Theorem 4.1 because the resource transfers to the activity have to be selected such that the resource units arrive as early as possible.

Below, apart from describing these neighborhoods, we consider some theoretical aspects of them. In particular, we regard their connectivity. As described for example by Brucker and Knust (2006), a neighborhood $\mathcal{N}$ is called connected if it is possible to transform an arbitrary solution s into each other solution s' by a finite number of modifications in $\mathcal{N}$. Here, for neighborhoods defined on solutions represented by resource flows, two resource flows $\mathcal{F}$ and $\mathcal{F}'$ (i.e. the two solutions) are equal if $f'_{ijk} = f_{ijk}$ holds for all $i \in V_0$, $j \in V_*$, and $k \in \mathcal{R}$. Even if a neighborhood $\mathcal{N}$ is not connected, the weaker concept of opt-connectivity may still hold for $\mathcal{N}$. In this case, a neighborhood $\mathcal{N}$ is called opt-connected if it is possible to transform an arbitrary solution s into an optimal solution s^* by a finite number of modifications in $\mathcal{N}$.

Now, we introduce a neighborhood $\mathcal{N}_{\text{reroute}}$ that defines modifications (referred to as reroute moves) based on the parallel modification by Fortemps and Hapke (1997). Here, similar to parallel modifications, two arbitrary arcs $(i,j)_k$ and $(u,v)_k$ with $f_{ijk} > 0$ and $f_{uvk} > 0$ are selected for a resource $k \in \mathcal{R}$ in resource flow $\mathcal{F}$ such that no directed path exists from activity v to activity i or from activity j to activity u in the AON-flow network. Then, an amount of $q \in \{1, \dots, \min\{f_{ijk}, f_{uvk}\}\}$ units of resource k are redirected from activity i to activity v as well as from activity u to activity j. As described above, this results in a resource flow $\mathcal{F}'$ with the modified resource transfers $f'_{ijk} = f_{ijk} - q$, $f'_{uvk} = f_{uvk} - q$, $f'_{ivk} = f'_{ivk} + q$, and $f'_{ujk} = f_{ujk} + q$ between the activities (cf. Figure 6.19). It should be noted that, unlike for parallel modifications, we do not restrict these moves to always reroute the maximal amount of resource units.

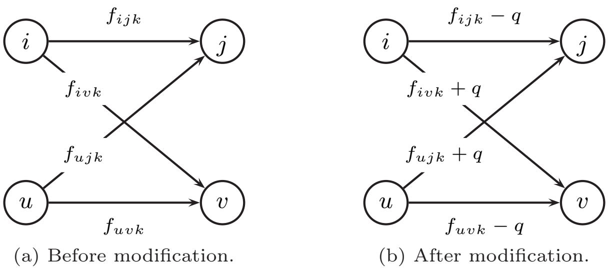


Figure 6.19: The resource transfers of a resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ and $f_{uvk} > 0$ between activities $i,u \in V_0$ and $j,v \in V_*$ before the resource transfers are modified by a reroute modification are displayed in (a) while the resource transfers after an amount of $q \in \{1, \ldots, \min\{f_{ijk}, f_{uvk}\}\}$ units of resource k has been rerouted from activity i to activity v as well as from activity u to activity j are displayed in (b).

Next, we introduce a neighborhood $\mathcal{N}_{\text{reverse}}$ that defines modifications (referred to as reverse moves) based on the serial modification by Fortemps and Hapke (1997). Here, instead of restricting ourselves to arcs $(i, j)_k$ between critical activities $i \in V$ and $j \in V$, arbitrary activities $i \in V$ and $j \in V$ with $f_{ijk} > 0$ for at least one resource $k \in \mathcal{R}$ are selected in resource flow $\mathcal{F}$ such that activity i is no direct or indirect predecessor of activity j according to the precedence constraints and no other directed path from activity i to activity j exists in the AON-flow network. Otherwise, if either of these two conditions does not hold, reversing all arcs $(i, j)_k$ between the two selected activities would result in a cyclic (and hence infeasible) resource flow.

As described in the previous section, Fortemps and Hapke (1997) do not explain in any detail how incoming as well as outgoing resource transfers for a serial modification are modified in the general case. Here, for a reverse modification in the neighborhood $\mathcal{N}_{\text{reverse}}$, we adapt and extend their general idea and select sets U_k of activities from which resource transfers are redirected to activity j as well as sets V_k of activities to which resource transfers are redirected from activity i based on a priority rule as follows. First of all, sets U_k of activities $u \in V_0$ are selected for each resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ based on a priority rule such that an arc $(u,i)_k$ with $f_{uik} > 0$ exists between each activity $u \in U_k$ as well as activity i and

$$\sum_{u \in U_k} f_{uik} \geq f_{ijk}$$

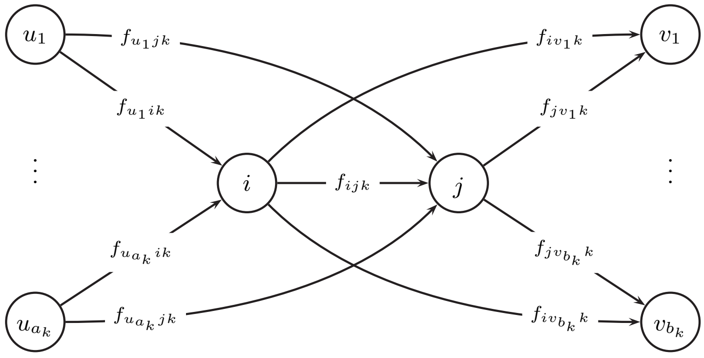
holds for the amount of resource units of resource k transferred from these activities to activity i. These sets are selected such that no activity $u \in U_k$ can be removed from the set U_k without violating this inequality (i.e. the inequality does not hold for any proper subset of the set U_k). Similarly, sets V_k of activities $v \in V_*$ are selected for each resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ based on a priority rule such that an arc $(j,v)_k$ with $f_{jvk} > 0$ exists between activity j as well as each activity $v \in V_k$ and

$$\sum_{v \in V_k} f_{jvk} \geq f_{ijk}$$

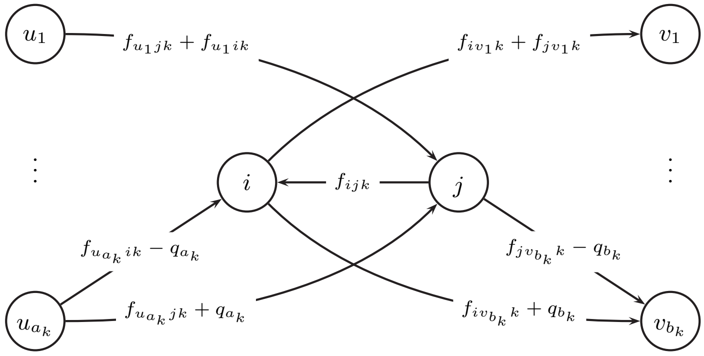
holds for the amount of resource units of resource k transferred from activity j to these activities. Again, these sets V_k are chosen such that no activity $v \in V_k$ can be removed from the set V_k without violating this inequality. In the following, we assume that the subset U_k contains a total of a_k activities $u_1, \ldots, u_{a_k}$ for each resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ while the subset V_k contains a total of b_k activities $v_1, \ldots, v_{b_k}$. Furthermore, we assume that the activities contained in these sets are sorted according to the same priority rule that has been used to calculate the sets. This initial situation for a resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ before a reverse modification is applied to a resource flow $\mathcal{F}$ is visualized in Figure 6.20(a).

As stated above, the sets U_k and V_k of activities are selected based on a priority rule. In the remainder of this section, we assume that the activities are selected according to increasing numbers. The actual priority rules used in the tabu search algorithm are then described in Section 6.3. It should be noted that the priority rules used here have no influence on the connectivity of the neighborhood.

Now, a reverse modification first reverses the direction of all arcs $(i,j)_k$ for all resources $k \in \mathcal{R}$ with $f_{ijk} > 0$ between activities i and j such that an



(a) Before modification.


u_1
$f_{u_1jk} + f_{u_1ik}$
$f_{iv_1k} + f_{jv_1k}$
v_1
i
f_{ijk}
j
$f_{u_{a_k}ik} - q_{a_k}$
$f_{jv_{b_k}k} - q_{b_k}$
u_{a_k}
$f_{u_{a_k}jk} + q_{a_k}$
$f_{iv_{b_k}k} + q_{b_k}$
v_{b_k}

(b) After modification.

Figure 6.20: The resource transfers of a resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ between two activities $i \in V$ and $j \in V$ as well as two sets $U_k = \{u_1, \dots, u_{a_k}\}$ and $V_k = \{v_1, \dots, v_{b_k}\}$ before the resource transfers are modified by a reverse modification are displayed in (a) while the resource transfers after a reverse modification has been applied are displayed in (b).

amount of $f'_{jik} = f_{ijk}$ units of resource $k \in \mathcal{R}$ is transferred from activity j to activity i in the resulting resource flow $\mathcal{F}'$. Additionally, for each resource $k \in \mathcal{R}$ with $f_{ijk} > 0$, the resource transfers for all activities $u \in U_k$ as well as for all activities $v \in V_k$ are adapted. Here, for all activities $u_\lambda \in U_k$

with $\lambda = 1, \ldots, a_{k-1}$, all units of resource k transferred from activity u_λ to activity i are redirected such that an amount of $f'_{u_\lambda jk} = f_{u_\lambda jk} + f_{u_\lambda ik}$ units of resource k is transferred from activity u_λ to activity j in the resulting resource flow $\mathcal{F}'$. For the remaining activity u_{a_k}, an amount of

$$q_{a_k} = f_{ijk} - \sum_{\lambda=1}^{a_{k-1}} f_{u_\lambda ik}$$

units of resource k is redirected such that $f'_{u_{a_k} jk} = f_{u_{a_k} jk} + q_{a_k}$ units of resource k are transferred from activity u_{a_k} to activity j in the resulting resource flow $\mathcal{F}'$ and an amount of $f'_{u_{a_k} ik} = f_{u_{a_k} ik} - q_{a_k}$ units of resource k is still transferred from activity u_{a_k} to activity i. Similarly, for all activities v_ν with $\nu = 1, \ldots, b_{k-1}$, all units of resource k transferred from activity j to activity v_ν are redirected such that an amount of $f'_{iv_\nu k} = f_{iv_\nu k} + f_{jv_\nu k}$ units of resource k is transferred from activity i to activity u_ν in the resulting resource flow $\mathcal{F}'$. For the remaining activity v_{b_k}, an amount of

$$q_{b_k} = f_{ijk} - \sum_{\nu=1}^{b_{k-1}} f_{jv_\nu k}$$

units of resource k is redirected such that $f'_{iv_{b_k} k} = f_{iv_{b_k} k} + q_{b_k}$ units of resource k are transferred from activity i to activity v_{b_k} in the resulting resource flow $\mathcal{F}'$ and an amount of $f'_{jv_{b_k} k} = f_{jv_{b_k} k} - q_{b_k}$ units of resource k is still transferred from activity j to activity v_{b_k}. These modifications ensure that the amount of resource k transferred from activities $u \in U_k$ to activity j is equal to the amount f_{ijk} on the reversed arc and, similarly, the amount of resource k transferred from activity i to activities $v \in V_k$ is also equal to the amount f_{ijk} for each resource $k \in \mathcal{R}$ with $f_{ijk} > 0$. As a result of this, flow conservation is maintained in the resulting resource flow $\mathcal{F}'$. Here, a modified graph for a resource $k \in \mathcal{R}$ with $f_{ijk} > 0$ after a reverse modification has been applied to a given resource flow $\mathcal{F}$ is visualized in Figure 6.20(b).

As described above, some conditions have to apply for the selected activities for a reroute or reverse modification in order to ensure that the resulting resource flow is acyclic. Here, ensuring that no directed path exists from either activity $j \in V_*$ to activity $u \in V_0$ or from activity $v \in V_*$ to activity $i \in V_0$ for a modification in the neighborhood $\mathcal{N}_{\text{reroute}}$ can be done in $\mathcal{O}(1)$ time based on the matrix containing the longest path lengths between the activities. Similarly, it can be ensured in $\mathcal{O}(1)$ time that activity $i \in V$ is no direct or indirect predecessor of activity $j \in V$ for a modification in the neighborhood $\mathcal{N}_{\text{reverse}}$. On the other hand, however, additional computational time is required in order to ensure for a reverse modification that no other directed path from activity $i \in V$ to activity $j \in V$ exists in the AON-flow network. This can, for example, be determined by a depth-first search in $\mathcal{O}(n^2)$ time if an adjacency matrix is used to represent which activities are connected by arcs. It should be noted that this approach is also used in the tabu search algorithm introduced in Section 6.3 if the neighborhood $\mathcal{N}_{\text{reverse}}$ (or a corresponding neighborhood for the RCPSP with first- and second-tier resources) is used.

Example 6.7 We consider a project consisting of $n = 5$ real activities $i = 1, \ldots, 5$ as well as the two dummy activities 0 and 6. Furthermore, $r = 2$ renewable resources with capacities $R_1 = 3$ and $R_2 = 2$ are available. The processing times p_i as well as the resource requirements r_{ik} of the activities are given in Table 6.4 while the activity-on-node network visualizing the precedence constraints between the activities is displayed in Figure 6.21.

i	0	1	2	3	4	5	6
p_i	0	1	2	2	1	1	0
r_{i1}	3	2	1	2	2	0	3
r_{i2}	2	0	0	1	1	2	2

Table 6.4: Processing times p_i and resource requirements r_{ik} of the activities of the project from Example 6.7.

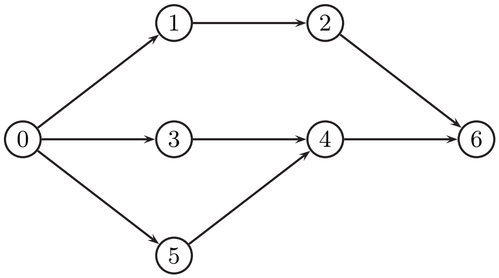


Figure 6.21: Activity-on-node network displaying the precedence constraints between the activities of the project from Example 6.7.

A feasible resource flow $\mathcal{F}$ for this project is displayed in Figure 6.22 while the earliest start schedule corresponding to this resource flow is shown in Figure 6.23.

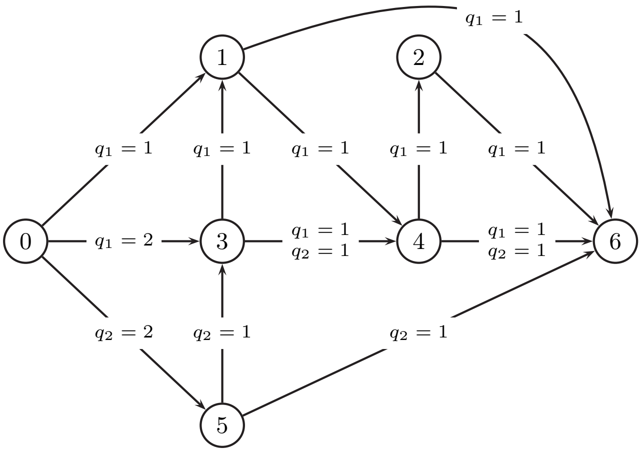


Figure 6.22: A feasible resource flow $\mathcal{F}$ for the project from Example 6.7.

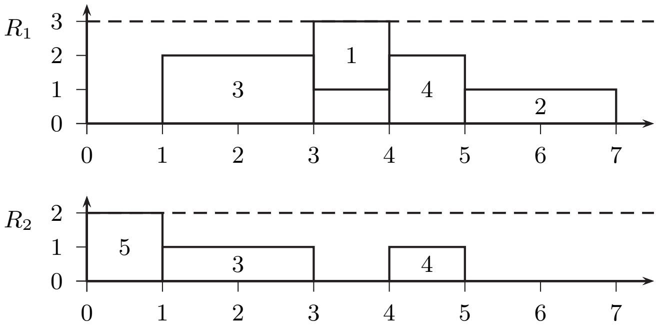


Figure 6.23: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}$ displayed in Figure 6.22.

Below, we now use a modification in the neighborhood $\mathcal{N}_{\text{reverse}}$ in order to reverse the direction of the arc $(5,3)_2$ with $f_{532} = 1$ between activities 5

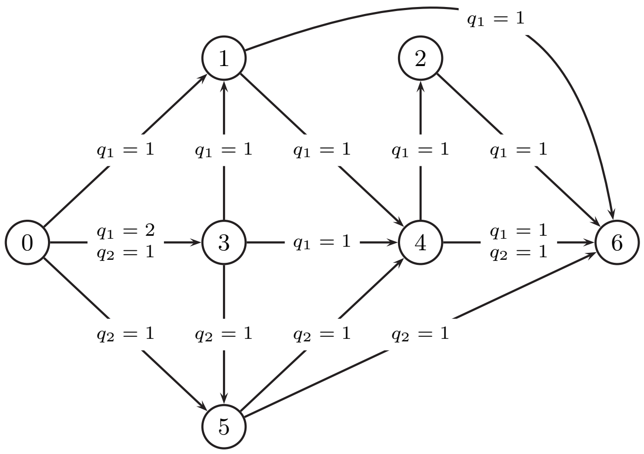


Figure 6.24: A feasible resource flow $\mathcal{F}'$ for the project from Example 6.7 after a reverse modification has been used on resource flow $\mathcal{F}$ from Figure 6.22 to reverse all arcs between activities 5 and 3.

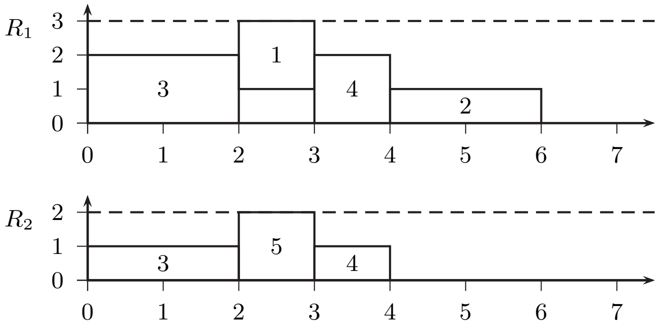


Figure 6.25: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}'$ displayed in Figure 6.24.

and 3. In addition to these two activities, two sets $U_2 = \{0\}$ and $V_2 = \{4\}$ are selected as described above. The resource flow $\mathcal{F}'$ resulting from this modification is displayed in Figure 6.24 while the earliest start schedule

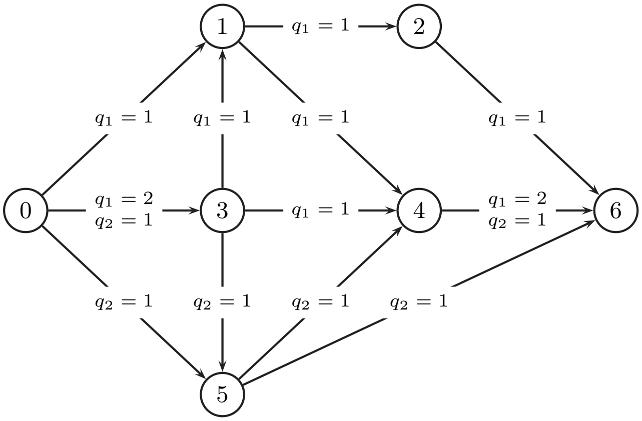


Figure 6.26: A feasible resource flow $\mathcal{F}''$ for the project from Example 6.7 after a reroute modification has been used on resource flow $\mathcal{F}'$ from Figure 6.24 to redirect one unit of resource 1 from activity 1 to activity 2 as well as from activity 4 to activity 6.

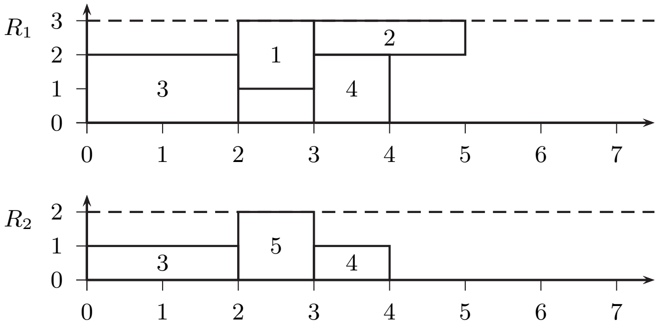


Figure 6.27: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}''$ displayed in Figure 6.26.

corresponding to this resource flow is shown in Figure 6.25. In this resource flow $\mathcal{F}'$, the modification of the selected arc $(5,3)_2$ results in the reversed arc $(3,5)_2$ such that activity 3 is processed before activity 5. Additionally, one unit of resource 2 is transferred from activity 0 to activity 3 and one unit of resource 2 is transferred from activity 5 to activity 4 in order to maintain flow conservation.

Next, we use a modification in the neighborhood $\mathcal{N}_{\text{reroute}}$ on resource flow $\mathcal{F}'$ in order to redirect one unit of resource 1 on the arcs $(1,6)_1$ and $(4,2)_1$. These arcs can be selected because no directed path exists from either activity 6 to activity 4 or from activity 2 to activity 1 in the AON-flow network. In the resulting resource flow $\mathcal{F}''$, one unit of resource 1 is rerouted from activity 1 to activity 2 and one unit of resource 1 is rerouted from activity 4 to activity 6. This resource flow $\mathcal{F}''$ is displayed in Figure 6.26 while the corresponding earliest start schedule is shown in Figure 6.27.

In this schedule, it can be seen that activity 2 is now processed by another resource unit of resource 1 than before, i.e. instead of changing the order in which activities are processed by the same resource units of a resource, reroute modifications change the resource plan and allow activities that require the same types of resources to be scheduled in parallel. Furthermore, it should be noted that this example also shows that both modifications are actually able to improve a given resource flow $\mathcal{F}$. Here, while the earliest start schedule for the initial resource flow $\mathcal{F}$ as it is displayed in Figure 6.23 has a makespan of $C_{max} = 7$, the earliest start schedule for resource flow $\mathcal{F}'$ as it is displayed in Figure 6.25 has a makespan of $C'_{max} = 6$ after a reverse modification has been used. Finally, the earliest start schedule for resource flow $\mathcal{F}''$ as it is displayed in Figure 6.27 has a makespan of $C''_{max} = 5$ after a reroute modification has been used. This latter schedule also is an optimal solution for this project. □

Now, we consider the size of the neighborhoods $\mathcal{N}_{\text{reroute}}$ and $\mathcal{N}_{\text{reverse}}$. Here, because an acyclic directed graph representing a resource flow $\mathcal{F}_k$ for a resource $k \in \mathcal{R}$ consists of at most $\frac{n\cdot(n-1)}{2}$ directed arcs, the size of the neighborhood $\mathcal{N}_{\text{reverse}}$ is bounded by $\mathcal{O}(n^2)$. It should be noted that the size of this neighborhood does not depend on the number of arcs connecting two activities $i \in V$ and $j \in V$ because all arcs between these activities are reversed simultaneously by a modification in the neighborhood $\mathcal{N}_{\text{reverse}}$.

On the other hand, two arbitrary arcs $(i,j)_k$ and $(u,v)_k$ as well as an amount $q \in \{1,\dots,\min\{f_{ijk}, f_{uvk}\}\}$ of resource units have to be selected for a modification in the neighborhood $\mathcal{N}_{\text{reroute}}$. Thus, the size of this neighborhood is bounded by $\mathcal{O}(rn^4 R_{max})$ where

$$R_{max} = \max_{k\in\mathcal{R}}\{R_k\}$$

is the maximal amount of available resource units of a resource $k \in \mathcal{R}$. For this neighborhood, if multiple arcs for resources $k \in \mathcal{R}$ connect two activities, each of these arcs has to be considered separately.

Due to the size of these neighborhoods (in particular, the pseudo-polynomial neighborhood $\mathcal{N}_{\text{reroute}}$ may be quite large), we now reduce the neighborhoods. First of all, similar to Fortemps and Hapke (1997), we introduce a neighborhood $\mathcal{N}^{\text{ca}}_{\text{reverse}}$ in which reverse moves are limited to critical activities $i \in V$ and $j \in V$ with $f_{ijk} > 0$ for at least one resource $k \in \mathcal{R}$. As before, in order to reverse the arcs $(i,j)_k$ for all resources $k \in \mathcal{R}$ with $f_{ijk} > 0$ between these activities, activity i may not be a direct or indirect predecessor of activity j according to the precedence constraints. If this condition is satisfied, however, reversing these arcs always results in a feasible resource flow, i.e. no additional computational time is required in order to ensure that no other directed path from activity i to activity j exists in resource flow $\mathcal{F}$ (cf. Theorem 6.6). The corresponding proof is based on the proof by van Laarhoven et al. (1992) for the neighborhood $\mathcal{N}_{\text{ca}}$ which is based on modifications of critical arcs in disjunctive graphs for the job-shop scheduling problem.

Theorem 6.6 *Let $h \in V$ and $j \in V$ be two consecutive critical activities in a feasible resource flow $\mathcal{F}$ with $f_{hjk} > 0$ for at least one resource $k \in \mathcal{R}$ such that activity h is no direct or indirect predecessor of activity j according to the precedence constraints. Then, reversing the critical arcs $(h,j)_k$ for all resources $k \in \mathcal{R}$ with $f_{hjk} > 0$ between these activities based on a modification in the neighborhood $\mathcal{N}^{ca}_{reverse}$ always results in a feasible (acyclic) resource flow $\mathcal{F}'$.* □

Proof Let $\mathcal{F}$ be a feasible (acyclic) resource flow. Furthermore, let $h \in V$ and $j \in V$ be two consecutive critical activities with $f_{hjk} > 0$ for at least one resource $k \in \mathcal{R}$ such that activity h is no direct or indirect predecessor of activity j according to the precedence constraints. Now, we assume to the contrary that reversing all critical arcs $(h,j)_k$ for resources $k \in \mathcal{R}$ with $f_{hjk} > 0$ between these two activities based on a modification in the neighborhood $\mathcal{N}^{\text{ca}}_{\text{reverse}}$ results in a cyclic (infeasible) resource flow $\mathcal{F}'$. In this case, because $\mathcal{F}$ is acyclic and only the order of activities h and j is changed in the resulting resource flow (cf. the proof of Theorem 6.7), the reversed arcs $(j,h)_k$ have to be part of the cycle. Then, a path $(h,i,\ldots,j)$ from activity h to activity j via at least one activity $i \in V$ must exist in resource flow $\mathcal{F}'$ which also exists in resource flow $\mathcal{F}$. This path $(h,i,\ldots,j)$ must be longer than the direct path (h,j) from activity h to activity j because all processing times p_i of real activities $i \in V$ can be assumed to be positive integers. Thus, the arcs $(h, j)_k$ between activities i and j can not be critical arcs, which contradicts the assumption. ■

The size of this reduced neighborhood $\mathcal{N}^{\text{ca}}_{\text{reverse}}$ is limited to $\mathcal{O}(n)$ if only one critical path in the AON-flow network is considered. Next, we introduce a reduced neighborhood $\mathcal{N}^{\max}_{\text{reroute}}$ in which always the maximal amount of $q = \min\{f_{ijk}, f_{uvk}\}$ units of resource k is rerouted (i.e. similar to the parallel modification described by Fortemps and Hapke (1997)). The size of this neighborhood is then limited to $\mathcal{O}(rn^4)$. This neighborhood can be further reduced by always selecting a critical arc $(i, j)_k$ as well as an arbitrary arc $(u, v)_k$ (instead of two arbitrary arcs) representing resource transfers $f_{ijk} > 0$ and $f_{uvk} > 0$ such that no directed path exists from activity $v \in V_*$ to activity $i \in V_0$ or from activity $j \in V_*$ to activity $u \in V_0$ in the AON-flow network. In this case, the size of the resulting neighborhood $\mathcal{N}^{\max,\text{ca}}_{\text{reroute}}$ is bounded by $\mathcal{O}(rn^3)$ if only one critical path in the AON-flow network is considered.

In the following, we deal with the question whether the defined neighborhoods (as well as combinations of these neighborhoods) are connected or at least opt-connected. First of all, in Example 6.8, we show that neither the neighborhood $\mathcal{N}_{\text{reroute}}$ nor the neighborhood $\mathcal{N}_{\text{reverse}}$ alone is opt-connected (and hence also not connected) for the RCPSP.

Example 6.8 We consider a project consisting of $n = 3$ real activities as well as the two dummy activities 0 and 4. Additionally, $r = 1$ renewable resource with a capacity of $R_1 = 2$ is given. The processing times p_i and resource requirements r_{i1} of the activities are given in Table 6.5. Finally, the precedence constraint $1 \rightarrow 3$ is given for this project.

i	0	1	2	3	4
p_i	0	1	1	1	0
r_{i1}	2	2	1	1	2

Table 6.5: Processing times p_i and resource requirements r_{ik} of the activities of the project considered in Example 6.8.

The unique optimal resource flow $\mathcal{F}^*$ for this project is displayed in Figure 6.28 while the earliest start schedule corresponding to this resource flow with the optimal makespan $C^*_{max} = 2$ is shown in Figure 6.29.

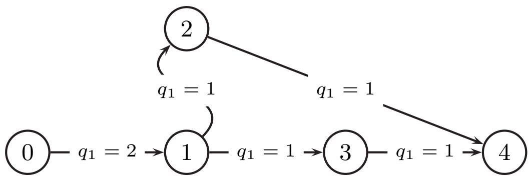


Figure 6.28: The unique optimal resource flow $\mathcal{F}^*$ for the project considered in Example 6.8.

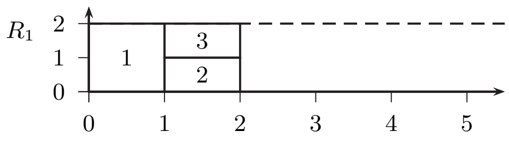


Figure 6.29: The earliest start schedule corresponding to the optimal resource flow $\mathcal{F}^*$ displayed in Figure 6.28.

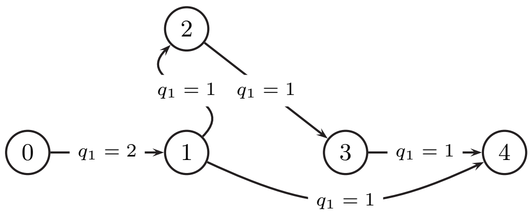


Figure 6.30: A feasible resource flow $\mathcal{F}$ for the project from Example 6.8.

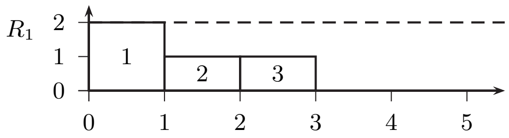


Figure 6.31: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}$ displayed in Figure 6.30.

Now, we consider a feasible resource flow $\mathcal{F}$ for this project as it is displayed in Figure 6.30. The earliest start schedule corresponding to this resource flow with the makespan $C_{max} = 3$ is shown in Figure 6.31. Here, if only the neighborhood $\mathcal{N}_{\text{reverse}}$ is available, it is not possible to improve the makespan of this schedule by reversing all arcs $(i,j)_1$ between two activities $i \in V$ and $j \in V$ because this only changes the sequence in which the activities are processed but not the resource plan. As a result of this, it is not possible to transform this resource flow $\mathcal{F}$ into the optimal resource flow $\mathcal{F}^*$ by using only the neighborhood $\mathcal{N}_{\text{reverse}}$, i.e. the neighborhood $\mathcal{N}_{\text{reverse}}$ is not opt-connected. Instead, in order to transform this resource flow into the optimal resource flow $\mathcal{F}^*$, the arcs $(2{,}3)_1$ and $(1{,}4)_1$ would have to be rerouted based on a modification in the neighborhood $\mathcal{N}_{\text{reroute}}$.

Finally, we consider another feasible resource flow $\mathcal{F}'$ as it is displayed in Figure 6.32. The earliest start schedule corresponding to this resource flow with the makespan $C'_{max} = 3$ is shown in Figure 6.33.

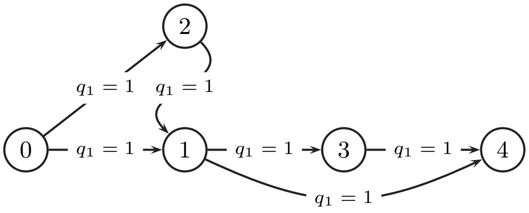


Figure 6.32: A feasible resource flow $\mathcal{F}'$ for the project from Example 6.8.

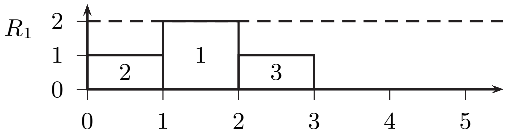


Figure 6.33: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}'$ displayed in Figure 6.32.

In this case, if only the neighborhood $\mathcal{N}_{\text{reroute}}$ is available, it is not possible to improve the makespan of the resulting schedule by only rerouting arcs. In particular, it is not possible to select an arc from the left side of the graph (i.e. between activities 0, 1, and 2) as well as one arc from the right side of the graph (i.e. between activities 1, 3, and 4) for a modification because these arcs are always connected by a path via activity 1. For this reason, it is not possible to transform this resource flow into the optimal resource flow $\mathcal{F}^*$ by using only the neighborhood $\mathcal{N}_{\text{reroute}}$, i.e. the neighborhood $\mathcal{N}_{\text{reroute}}$ is not opt-connected. Instead, in order to transform this resource flow into the optimal resource flow $\mathcal{F}^*$, it is necessary to reverse the critical arc $(2,1)_1$ between activities 2 and 1 based on a modification in the neighborhood $\mathcal{N}_{\text{reverse}}$. Using a priority rule that selects activities according to increasing numbers, this results in resource flow $\mathcal{F}$ as it is displayed in Figure 6.30 which can then be transformed into the optimal resource flow $\mathcal{F}^*$ as described above. □

Thus, as visualized in this example, neither the neighborhood $\mathcal{N}_{\text{reroute}}$ nor the neighborhood $\mathcal{N}_{\text{reverse}}$ (nor any of the reduced neighborhoods $\mathcal{N}^{\text{max}}_{\text{reroute}}$, $\mathcal{N}^{\text{max,ca}}_{\text{reroute}}$, and $\mathcal{N}^{\text{ca}}_{\text{reverse}}$) is opt-connected (and hence also not connected) by itself. For this reason, we now consider the neighborhood $\mathcal{N}_1 = \mathcal{N}_{\text{reroute}} \cup \mathcal{N}_{\text{reverse}}$. Below,we show that this neighborhood is connected in Theorem 6.7. In order to prove this theorem, it is necessary to calculate topological orderings for AON-flow networks based on given resource flows. A topological ordering then defines an order in which the activities can be processed such that all precedence constraints as well as all resource transfers between the activities are observed (i.e. if a precedence constraint $i \to j$ or a resource transfer $f_{ijk} > 0$ for some resource $k \in \mathcal{R}$ exists between activities $i \in V_0$ and $j \in V_*$, activity i comes before activity j in the topological ordering).

It should be noted, however, that a topological ordering is not necessarily unique. In particular, if no directed path exists from either activity i to activity j or from activity j to activity i, no particular order has to exist between activities i and j (i.e. activity i can come either before or after activity j in a topological ordering). This is the case, for example, if both activities can be processed in parallel by different resource units. In this case, we assume that activities that can be processed in parallel are ordered according to increasing numbers. Then, a unique topological ordering can be calculated for each AON-flow network.

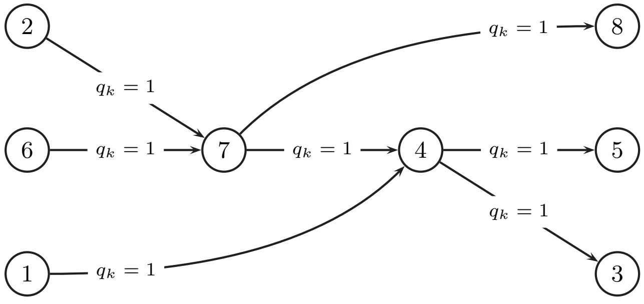
Theorem 6.7 *The neighborhood $\mathcal{N}_1$ is connected for the RCPSP.* □

Proof Let $\mathcal{F}$ and $\mathcal{F}'$ be two arbitrary feasible resource flows with $\mathcal{F} \neq \mathcal{F}'$. Furthermore, let $\pi = (\pi_0, \ldots, \pi_{n+1})$ and $\pi' = (\pi'_0, \ldots, \pi'_{n+1})$ be the topological orderings of the corresponding AON-flow networks. We now show that the neighborhood $\mathcal{N}_1$ is connected by transforming resource flow $\mathcal{F}'$ into resource flow $\mathcal{F}$ by a finite number of modifications in $\mathcal{N}_1$.

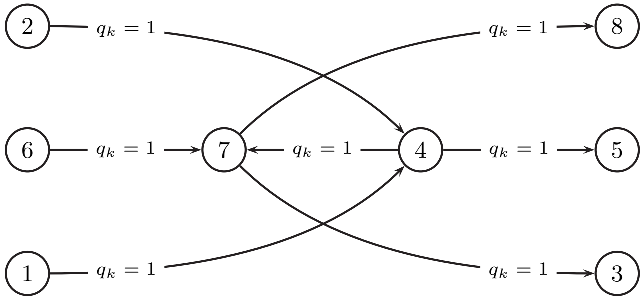
First, we consider all activities $i \in V$ and $j \in V$ with $f'_{ijk} > 0$ for some resource $k \in \mathcal{R}$ in resource flow $\mathcal{F}'$ (i.e. activity i comes before activity j in the corresponding topological ordering π') where activity j comes before activity i in the topological ordering π for resource flow $\mathcal{F}$. This implies that there exist no direct or indirect resource transfers from activity i to activity j in resource flow $\mathcal{F}$. If such activities i and j exist, it is always possible to select two of these activities in resource flow $\mathcal{F}'$ such that no other path exists from activity i to activity j via other activities. This follows from the following reasoning.

Let $u \in V$ and $w \in V$ be two of these candidates for which at least one path $P = (u, v_1, \ldots, v_\mu, w)$ via μ other activities $v_1, \ldots, v_\mu \in V$ exists from activity u to activity w. In this case, none of the activities v_1 to v_μ can simultaneously be a successor of activity u as well as a predecessor of activity w in the topological ordering π for resource flow $\mathcal{F}$. Otherwise, if an activity $v \in \{v_1, \ldots, v_\mu\}$ came after activity u as well as before activity w in the topological ordering π for resource flow $\mathcal{F}$, this would imply that also activity u comes before activity w which contradicts the condition that activity w comes before activity u in the topological ordering π. Thus, it is always possible to identify two consecutive activities $i \in \{u, v_1, \ldots, v_\mu\}$ and $j \in \{v_1, \ldots, v_\mu, w\}$ on this path such that activity i comes after activity j in the topological ordering π. By this reasoning, we find two activities $i \in V$ and $j \in V$ with the above property.

Now, the arcs $(i,j)_k$ for all resources $k \in \mathcal{R}$ with $f'_{ijk} > 0$ between the selected activities i and j can be reversed based on a modification in the neighborhood $\mathcal{N}_{\text{reverse}}$. In the resulting resource flow, activity j comes before activity i, i.e. the order of these two activities is reversed and matches the order of the activities in the topological ordering π for resource flow $\mathcal{F}$. Additionally, incoming resource transfers to activity i as well as outgoing resource transfers from activity j have to be adapted as described above. Here, it can be inferred from Figure 6.20 as well as from Figure 6.34 that the order in which these activities have to be processed in relation to activities i and j does not change. Instead, all activities $u \in V_0$ from which resource units are redirected to activity j are still predecessors of activity i in the resulting topological ordering and all activities $v \in V_*$ to which resource units are redirected from activity i are still successors of activity j.



(a) Before modification (with $\pi = (1, 2, 6, 7, 4, 3, 5, 8)$).



(b) After modification (with $\pi = (1, 2, 4, 5, 6, 7, 3, 8)$).

Figure 6.34: Extract of a resource flow before (cf. (a)) as well as after (cf. (b)) the only arc between activities 7 and 4 has been reversed. In the resulting resource flow, activity 4 comes before activity 5. Additionally, activity 6 (from which no resource unit is redirected to activity 4) can be processed in parallel to activities 4 and 5, and activity 5 (to which no resource unit is redirected from activity 7) can be processed in parallel to activities 7, 3, and 8 such that the topological ordering of these activities changes based on their numerical ordering.

It should be noted that activities $u \in V_0$ from which resource units are transferred to activity i in resource flow $\mathcal{F}'$ (as well as direct and indirect predecessors of these activities u) that are not affected by a reverse modification (i.e. no resource units are redirected from these activities u to activity j) can be processed in parallel to activity j (as well as in parallel to direct and indirect successors of activity j) in the resulting resource flow unless other directed paths exist between these activities. Similarly, activities $v \in V_*$ to which resource units are transferred from activity j in resource flow $\mathcal{F}'$ (as well as direct and indirect successors of these activities v) that are not affected by a reverse modification (i.e. no resource units are redirected from activity i to these activities v) can be processed in parallel to activity i (as well as in parallel to direct and indirect successors of activity i) in the resulting resource flow unless other directed paths exist between these activities. Here, while the topological ordering of these activities may be different in the resulting resource flow, no additional arcs are inserted into the resource flow that might have to be reversed in order to obtain π. An example for this situation is visualized in Figure 6.34.

Thus, each reverse modification restores the order of two activities $i \in V$ and $j \in V$ to the same order as in the topological ordering π for resource flow $\mathcal{F}$ and no additional arcs result from these moves that have to be reversed in order to obtain π. For this reason, it is possible to transform resource flow $\mathcal{F}'$ into a resource flow $\mathcal{F}''$ by a finite number of modifications in the neighborhood $\mathcal{N}_{\text{reverse}}$ such that the topological ordering of all activities $i \in V$ and $j \in V$ on directed paths in resource flow $\mathcal{F}''$ is the same as for resource flow $\mathcal{F}$.

Now, if $f''_{ijk} = f_{ijk}$ holds for all resource transfers of resource $k \in \mathcal{R}$ between activities $i \in V_0$ and $j \in V_*$ in resource flows $\mathcal{F}''$ and $\mathcal{F}$, resource flow $\mathcal{F}'$ has been transformed into resource flow $\mathcal{F}$ and we are finished. Otherwise, due to flow conservation, at least two resource transfers $f''_{ijk} > 0$ and $f''_{uvk} > 0$ between activities $i, u \in V_0$ and $j, v \in V_*$ have to exist with the following properties:

- The amount of resource units of resource k transferred on these arcs is larger than in resource flow $\mathcal{F}$, i.e. $f_{ijk} < f''_{ijk}$ and $f_{uvk} < f''_{uvk}$ hold.
- Without loss of generality it can be assumed that the amount of resource units of resource k transferred from activity i to activity v in resource flow $\mathcal{F}''$ is smaller than in resource flow $\mathcal{F}$, i.e. $f_{ivk} > f''_{ivk}$ holds.
- Activity j does not come before activity u in the topological ordering π for resource flow $\mathcal{F}$. This is assured because all activities share at least one common predecessor (i.e. dummy activity 0) as well as one common successor (i.e. dummy activity $n + 1$) and all activities on

directed paths from dummy source node 0 to dummy sink node $n + 1$ are ordered according to the topological ordering π.

After such activities have been identified in resource flow $\mathcal{F}''$, it is possible to reroute an amount of $q = \min\{f''_{ijk} - f_{ijk}, f''_{uvk} - f_{uvk}, f_{ivk} - f''_{ivk}\}$ units of resource k from activity i to activity v as well as from activity u to activity j by a modification in the neighborhood $\mathcal{N}_{\text{reroute}}$. Now, we can consider the four affected resource transfers before as well as after the modification in comparison to resource flow $\mathcal{F}$:

- We have $f_{ijk} < f''_{ijk}$ before the modification. After the modification, $q \leq f''_{ijk} - f_{ijk}$ units of resource k are redirected from activity i to activity v, i.e. we have $f_{ijk} \leq f''_{ijk} - q$. Thus, the deviation of the resulting resource transfers is reduced by q units.
- We have $f_{uvk} < f''_{uvk}$ before the modification. After the modification, $q \leq f''_{uvk} - f_{uvk}$ units of resource k are redirected from activity u to activity j, i.e. we have $f_{uvk} \leq f''_{uvk} - q$. Thus, the deviation of the resulting resource transfers is reduced by q units.
- We have $f_{ivk} > f''_{ivk}$ before the modification. After the modification, $q \leq f_{ivk} - f''_{ivk}$ units of resource k are redirected from activity i to activity v, i.e. we have $f_{ivk} \geq f''_{ivk} + q$. Thus, the deviation of the resulting resource transfers is reduced by q units.
- In the worst case, we have $f_{ujk} \leq f''_{ujk}$ before the modification. After the modification, q units of resource k are redirected from activity u to activity j, i.e. we have $f_{ujk} < f''_{ujk} + q$. Thus, the deviation of the resulting resource transfers is increased by q units.

As a result of this, the deviation for three of the resulting resource transfers is reduced by q units each while it is increased by at most q units for one resource transfer. Then, because all other resource transfers remain unchanged, the deviation of the resulting resource flow from resource flow $\mathcal{F}$ is always reduced by at least $2q$ units compared to the deviation of resource flow $\mathcal{F}''$ and resource flow $\mathcal{F}$. Furthermore, because the modified resource transfers are selected such that activity u comes before activity j in the topological ordering π for resource flow $\mathcal{F}$, no arc is inserted that might have to be reversed based on a modification in the neighborhood $\mathcal{N}_{\text{reverse}}$.

Thus, because each reroute modification reduces the deviation of the corresponding flows (and at least one resource transfer is set to the same value as in resource flow $\mathcal{F}$), it is possible to transform resource flow $\mathcal{F}''$ into a resource flow $\mathcal{F}'''$ by a finite number of modifications in the neighborhood $\mathcal{N}_{\text{reroute}}$ such that $f'''_{ijk} = f_{ijk}$ holds for all $i \in V_0$, $j \in V_*$, and $k \in \mathcal{R}$, i.e. resource flow $\mathcal{F}'$ can be transformed into resource flow $\mathcal{F}$ by a finite number of modifications in the neighborhood $\mathcal{N}_1$. ■

Next, we consider the connectivity of the reduced neighborhood $\mathcal{N}_2 = \mathcal{N}^{\text{max,ca}}_{\text{reroute}} \cup \mathcal{N}^{\text{ca}}_{\text{reverse}}$. For this neighborhood, it is easy see that it is no longer connected. In particular, limiting the neighborhoods $\mathcal{N}^{\text{max,ca}}_{\text{reroute}}$ and $\mathcal{N}^{\text{ca}}_{\text{reverse}}$ to critical arcs prevents the neighborhoods from modifying resource transfers between activities that are not critical. Thus, if two resource flows $\mathcal{F}$ and $\mathcal{F}'$ with $\mathcal{F} \neq \mathcal{F}'$ have the same critical paths, it is not always possible to transform resource flow $\mathcal{F}$ into $\mathcal{F}'$ by modifications in the neighborhood $\mathcal{N}_2$ (cf. Example 6.9).

Example 6.9 We consider a project consisting of $n = 5$ real activities as well as two dummy activities 0 and 5. Furthermore, $r = 2$ renewable resources with capacities $R_1 = 1$ and $R_2 = 1$ are available. The processing times p_i and resource requirements r_{ik} of the activities are given in Table 6.6. Finally, no precedence constraints are given between the real activities.

i	0	1	2	3	4	5
p_i	0	1	1	2	3	0
r_{i1}	1	0	0	1	1	1
r_{i2}	2	1	1	1	0	2

Table 6.6: Processing times p_i and resource requirements r_{ik} for the project considered in Example 6.9.

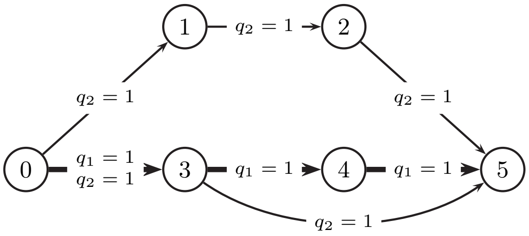


Figure 6.35: Feasible resource flow $\mathcal{F}$ for the project from Example 6.9. The unique critical path is highlighted in this graph.

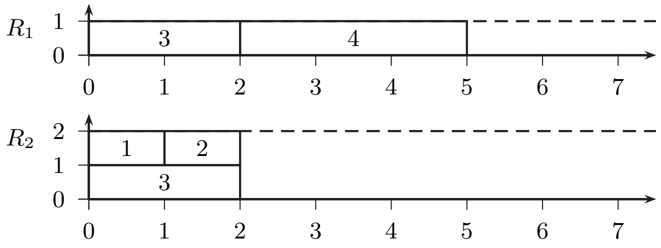


Figure 6.36: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}$ displayed in Figure 6.35.

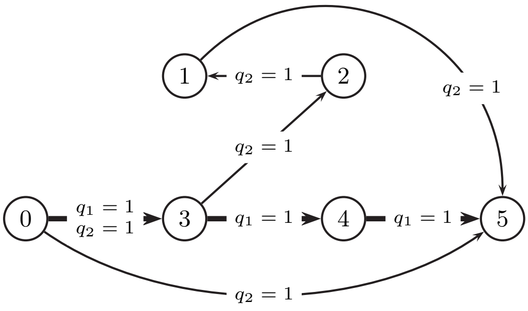


Figure 6.37: Feasible resource flow $\mathcal{F}'$ for the project from Example 6.9. The unique critical path is highlighted in this graph.

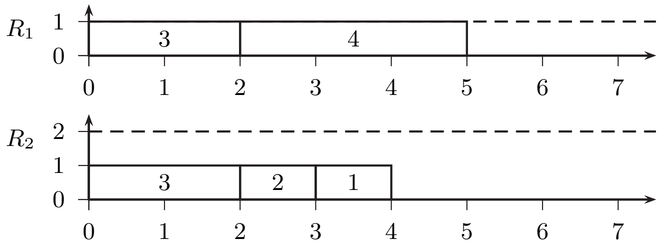


Figure 6.38: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}'$ displayed in Figure 6.37.

A feasible resource flow $\mathcal{F}$ for this project is displayed in Figure 6.35 while another feasible resource flow $\mathcal{F}'$ is given in Figure 6.37. The earliest start schedules corresponding to these resource flows are shown in Figures 6.36 and 6.38, respectively. Here, it is not possible to transform resource flow $\mathcal{F}'$ into resource flow $\mathcal{F}$ based on modifications in the neighborhood $\mathcal{N}_2$ because the unique critical path is the same in both graphs. Instead, in order to transform $\mathcal{F}'$ into $\mathcal{F}$, it would be required to reverse the arc $(2{,}1)_2$ based on a modification in the neighborhood $\mathcal{N}_{\text{reverse}}$ and then reroute $q = 1$ unit of resource 1 on the arcs $(0{,}5)_2$ and $(3{,}1)_2$ based on a modification in the neighborhood $\mathcal{N}_{\text{reroute}}$. □

It should be noted that this example does not imply that the reduced neighborhood $\mathcal{N}_2$ is not opt-connected. In particular, schedules that share at least one critical path from dummy source activity 0 to dummy sink activity $n+1$ also have the same makespan. Thus, both schedules displayed in Figures 6.36 and 6.38 are optimal solutions for the project considered in Example 6.9. Finally, we show in Example 6.10 that already the neighborhood $\mathcal{N}_3 = \mathcal{N}^{\max}_{\text{reroute}} \cup \mathcal{N}_{\text{reverse}}$ is no longer connected.

Example 6.10 We consider a project consisting of $n = 4$ real activities as well as $r = 1$ renewable resource with a capacity of $R_1 = 4$. The processing times of the real activities are $p_i = 1$ for $i = 1, \ldots, 4$ while the resource requirements are $r_{i1} = 2$ for all real activities $i = 1, \ldots, 4$. Finally, no precedence constraints are given between the real activities. A feasible resource flow $\mathcal{F}$ for this project is displayed in Figure 6.39 while another feasible resource flow $\mathcal{F}'$ is shown in Figure 6.40.

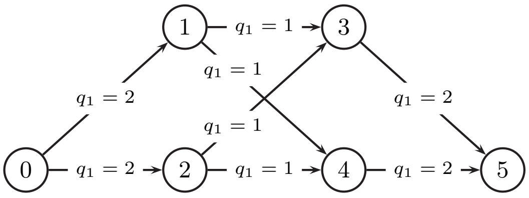


Figure 6.39: Feasible resource flow $\mathcal{F}$ for the project from Example 6.10.

Here, because modifications in both neighborhoods $\mathcal{N}^{\max}_{\text{reroute}}$ and $\mathcal{N}_{\text{reverse}}$ always redirect the maximal amount of $q = 2$ units of resource 1 in resource flow $\mathcal{F}'$ as well as in any resulting resource flow, it is impossible to transform resource flow $\mathcal{F}'$ into resource flow $\mathcal{F}$. □

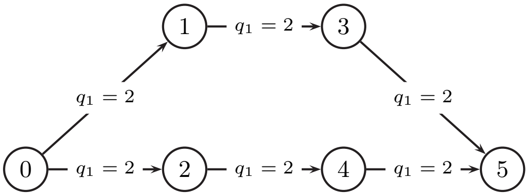


Figure 6.40: Feasible resource flow $\mathcal{F}'$ for the project from Example 6.10.

While this example shows that the neighborhood $\mathcal{N}_3$ is not connected, it again does not imply that it is not opt-connected. In particular, because the amount of resource units transferred between two activities has no influence on the quality of a solution with respect to the makespan C_{max}, a feasible resource flow $\mathcal{F}'$ can not have a larger makespan than another feasible resource flow $\mathcal{F}$ if no arc $(i, j)_k$ between activities $i \in V_0$ and $j \in V_*$ for a resource $k \in \mathcal{R}$ exists in resource flow $\mathcal{F}'$ that does not exist in resource flow $\mathcal{F}$. Thus, both resource flows displayed in Figures 6.39 and 6.40 are optimal solutions for the project considered in Example 6.10.

Neighborhood	connected	opt-connected
$\mathcal{N}_1 = \mathcal{N}_{\text{reroute}} \cup \mathcal{N}_{\text{reverse}}$	✓	✓
$\mathcal{N}_2 = \mathcal{N}^{\text{max,ca}}_{\text{reroute}} \cup \mathcal{N}^{\text{ca}}_{\text{reverse}}$	$\times$	open
$\mathcal{N}_3 = \mathcal{N}^{\text{max}}_{\text{reroute}} \cup \mathcal{N}_{\text{reverse}}$	$\times$	open

Table 6.7: Results concerning the connectivity of the three neighborhoods $\mathcal{N}_1$, $\mathcal{N}_2$, and $\mathcal{N}_3$ considered in Section 6.2.2.

The results concerning the connectivity of the three neighborhoods $\mathcal{N}_1$, $\mathcal{N}_2$, and $\mathcal{N}_3$ considered in this section are summarized in Table 6.7. As can be seen in this table, it remains an open question whether the two reduced neighborhoods $\mathcal{N}_2$ and $\mathcal{N}_3$ are opt-connected.

6.2.3 RCPSP with First-Tier Resource Transfers

For the RCPSP with first-tier resource transfers, the only difference of the solution representation in comparison to the classical RCPSP is that arcs

$(i,j)_k$ representing resource transfers of resource $k \in \mathcal{R}$ between two activities $i \in V_0$ and $j \in V_*$ are weighted with the transfer time Δ_{ijk} between the two activities (cf. Section 6.1.2). Thus, because the neighborhoods introduced in the previous section have been defined on modifications that are independent of the arc weights, these neighborhoods can be used for the RCPSP with first-tier resource transfers as they have been described above. Additionally, all results related to the neighborhoods $\mathcal{N}_{\text{reroute}}$, $\mathcal{N}_{\text{reverse}}$, and $\mathcal{N}_1 = \mathcal{N}_{\text{reroute}} \cup \mathcal{N}_{\text{reverse}}$ also hold for the RCPSP with first-tier resource transfers because all of these results are independent of the arc weights. In particular, the neighborhood $\mathcal{N}_1$ is also connected for the RCPSP with first-tier resource transfers.

Regarding the reduced neighborhoods, however, the result from Theorem 6.6 does not hold for the RCPSP with first-tier resource transfers. In particular, another shorter path via other activities $i \in V$ might exist between two consecutive critical activities $h \in V$ and $j \in V$ with $f_{hjk} > 0$ for some resource $k \in \mathcal{R}$ even if the triangle inequality $\Delta_{hik} + \Delta_{ijk} \geq \Delta_{hjk}$ holds for the transfer times of resources $k \in \mathcal{R}$. For this reason, it may not always be possible to reverse all arcs $(h,j)_k$ between these activities even if activity h is no direct or indirect predecessor of activity j according to the precedence constraints. Based on this result, it can be shown that the neighborhood $\mathcal{N}_2 = \mathcal{N}^{\text{max,ca}}_{\text{reroute}} \cup \mathcal{N}^{\text{ca}}_{\text{reverse}}$ is not opt-connected for the RCPSP with first-tier resource transfers.

Example 6.11 We consider a project consisting of $n = 4$ real activities as well as $r = 3$ renewable resources with the capacities $R_1 = 1$, $R_2 = 1$, and $R_3 = 1$. The processing times p_i of the activities are given in Table 6.8 while the transfer times Δ_{ijk} for resources $k = 1{,}2$ are given in Table 6.9. For this project, it is assumed that $\Delta_{ij3} = 0$ holds for all transfer times of resource $k = 3$ between activities $i{,}j = 0, \dots, 5$. Finally, the precedence constraints $1 \rightarrow 2$ and $3 \rightarrow 4$ are given between real activities.

i	0	1	2	3	4	5
p_i	0	1	3	1	1	0
r_{i1}	1	1	0	1	0	1
r_{i2}	1	1	0	1	1	1
r_{i3}	1	1	1	0	0	1

Table 6.8: Processing times p_i and resource requirements r_{ik} of the activities for the project considered in Example 6.11.

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	4	4	4	0
2	0	4	0	4	4	0
3	0	4	4	0	4	0
4	0	4	4	4	0	0
5	0	0	0	0	0	0

(a) Transfer times $\varDelta_{ij1}$.

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	1	1	1	0
2	0	1	0	1	1	0
3	0	1	1	0	1	0
4	0	1	1	1	0	0
5	0	0	0	0	0	0

(b) Transfer times $\varDelta_{ij2}$.

Table 6.9: The transfer times $\varDelta_{ij1}$ for resource $k = 1$ are displayed in (a) while the transfer times $\varDelta_{ij2}$ for resource $k = 2$ are shown in (b).

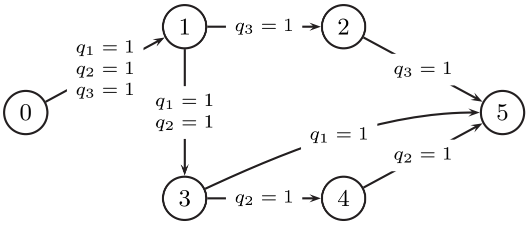


Figure 6.41: The unique optimal resource flow $\mathcal{F}^*$ for the project considered in Example 6.11.

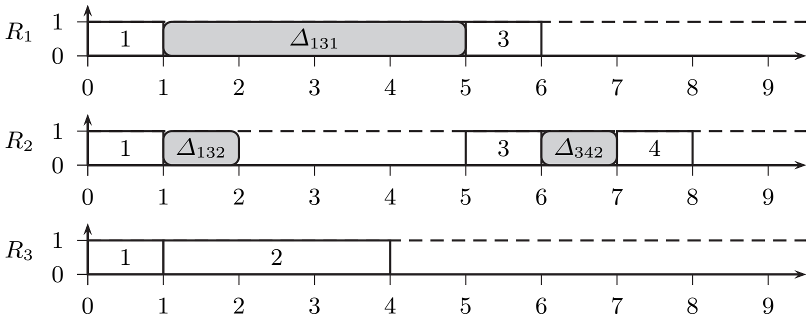


Figure 6.42: Earliest start schedule corresponding to the optimal resource flow $\mathcal{F}^*$ from Figure 6.41.

The unique optimal resource flow $\mathcal{F}^*$ for this project is displayed in Figure 6.41 while another feasible resource flow $\mathcal{F}$ is shown in Figure 6.43. The corresponding earliest start schedules with makespan $C^*_{max} = 8$ for resource flow $\mathcal{F}^*$ and makespan $C_{max} = 9$ for resource flow $\mathcal{F}$ are visualized in Figures 6.42 and 6.44, respectively.

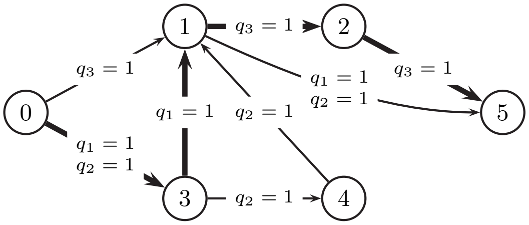


Figure 6.43: Feasible resource flow $\mathcal{F}$ for the project from Example 6.11. The unique critical path is highlighted in this graph.

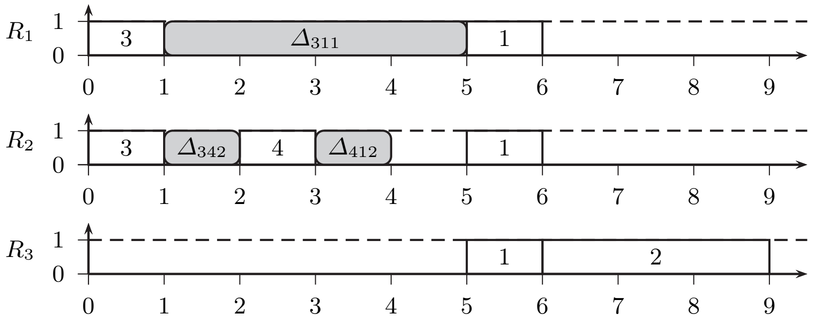


Figure 6.44: Earliest start schedule corresponding to the feasible resource flow $\mathcal{F}$ from Figure 6.43.

Here, resource flow $\mathcal{F}$ can not be transformed into the optimal resource flow $\mathcal{F}^*$ based on modifications in the neighborhood $\mathcal{N}_2 = \mathcal{N}^{\text{max,ca}}_{\text{reroute}} \cup \mathcal{N}^{\text{ca}}_{\text{reverse}}$ for the following reasons. First of all, no modification in the neighborhood $\mathcal{N}^{\text{max,ca}}_{\text{reroute}}$ (or even in the neighborhood $\mathcal{N}_{\text{reroute}}$) can be used because all resources have unit capacity (i.e. no resource units can be rerouted). Furthermore, on the critical path, only the order of activities 3 and 1 is not predetermined by precedence constraints. The arc $(3, 1)_1$ between these activities can not be reversed based on a modification in the neighborhood $\mathcal{N}^{\text{ca}}_{\text{reverse}}$, however, because another directed path from activity 3 to activity 1 via activity 4 exists (i.e. reversing the arc $(3,1)_1$ would result in a cyclic and hence infeasible resource flow). Thus, the neighborhood $\mathcal{N}_2$ is not opt-connected for the RCPSP with first-tier resource transfers. $\square$

Neighborhood	connected	opt-connected
$\mathcal{N}_1 = \mathcal{N}_{\text{reroute}} \cup \mathcal{N}_{\text{reverse}}$	✓	✓
$\mathcal{N}_2 = \mathcal{N}^{\text{max,ca}}_{\text{reroute}} \cup \mathcal{N}^{\text{ca}}_{\text{reverse}}$	$\times$	$\times$
$\mathcal{N}_3 = \mathcal{N}^{\text{max}}_{\text{reroute}} \cup \mathcal{N}_{\text{reverse}}$	$\times$	open
$\mathcal{N}_4 = \mathcal{N}^{\text{max,ca}}_{\text{reroute}} \cup \mathcal{N}_{\text{reverse}}$	$\times$	open

Table 6.10: Results concerning the connectivity of the four neighborhoods $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{N}_3$, and $\mathcal{N}_4$ considered in Section 6.2.3.

Based on these results, Table 6.7 from Section 6.2.2 can be updated for the RCPSP with first-tier resource transfers (cf. Table 6.10). Here, in particular, it remains an open question whether the two reduced neighborhoods $\mathcal{N}_3 = \mathcal{N}^{\text{max}}_{\text{reroute}} \cup \mathcal{N}_{\text{reverse}}$ or $\mathcal{N}_4 = \mathcal{N}^{\text{max,ca}}_{\text{reroute}} \cup \mathcal{N}_{\text{reverse}}$ are opt-connected.

6.2.4 RCPSP with First- and Second-Tier Resource Transfers

In this section, we consider the RCPSP with first- and second-tier resource transfers. For this problem, the neighborhoods introduced above have to be adapted in order to accommodate second-tier resource transfers that require supporting resources. Below, we introduce modified neighborhoods $\tilde{\mathcal{N}}_{\text{reroute}}$ and $\tilde{\mathcal{N}}_{\text{reverse}}$ based on the neighborhoods $\mathcal{N}_{\text{reroute}}$ and $\mathcal{N}_{\text{reverse}}$ described above. For this, resource transfers of resources $k \in \mathcal{R}^{\text{sa}}$ as well as resource transfers of resources $l \in \mathcal{R}^{\text{ru}}$ are regarded separately.

First of all, we deal with resource transfers of resources $k \in \mathcal{R}^{\text{sa}}$. In this case, arcs $(h^{\text{in}}, j^{\text{in}})_k$ represent start-to-start transfers of a first-tier resource $k \in \mathcal{R}^{\text{sa}}$ from activity $h \in V_0$ to activity $j \in V_*$ while arcs $(h^{\text{out}}, j^{\text{in}})_k$ represent finish-to-start transfers. Similarly, arcs $(h^{\text{in}}, i^{\text{out}}, j^{\text{in}})_{kl}$ represent start-to-start transfer of a second-tier resource $k \in \mathcal{R}^{\text{sa}}$ from activity $h \in V_0$ that supports the transfer of first-tier resource $l \in \mathcal{R}^{\text{ru}}$ from activity $i \in V_0$ to activity $j \in V_*$ while arcs $(h^{\text{out}}, i^{\text{out}}, j^{\text{in}})_{kl}$ represent finish-to-start transfers.

Now, if two first-tier resource transfers of resource $k \in \mathcal{R}^{\text{sa}}$ are rerouted based on a modification in the neighborhood $\tilde{\mathcal{N}}_{\text{reroute}}$, the resulting resource transfers are calculated as described above for the classical RCPSP. For example, if two arbitrary arcs $(h^{\text{in}}, j^{\text{in}})_k$ and $(u^{\text{out}}, w^{\text{in}})_k$ are selected such that no directed path exists from either node j^{in} to node u^{out} or from node w^{in} to node h^{in}, an amount of $q \in \{1, \ldots, \min\{f^{SS}_{hjk}, f^{FS}_{uwk}\}\}$ units of resource k are redirected from the start of activity h to the start of activity w as well as from the end of activity u to the start of activity j.

On the other hand, if at least one of the selected arcs represents a second-tier resource transfer, it has to be ensured that the supported first-tier resource $l \in \mathcal{R}^{\text{ru}}$ is still supported after the modification has been applied. For example, we assume that an arbitrary arc $(h^{\text{in}}, j^{\text{in}})_k$ representing a first-tier resource transfer as well as an arbitrary arc $(u^{\text{out}}, v^{\text{out}}, w^{\text{in}})_{kl}$ representing a second-tier resource transfer are selected such that no directed path exists from either node j^{in} to node u^{out} or from node w^{in} to node h^{in}. Then, an amount of $q \in \{1, \ldots, \min\{f^{SS}_{hjk}, f^{FS}_{uvwkl}\}\}$ units of resource k are rerouted from the end of activity u to the start of activity j as a first-tier resource transfer, i.e. this transfer is represented by an arc $(u^{\text{out}}, j^{\text{in}})_k$. Additionally, an amount of q units of resource k are rerouted from the start of activity h to support the transfer of resource l from activity v to activity w as a second-tier resource transfer, i.e. this transfer is represented by an arc $(h^{\text{in}}, v^{\text{out}}, w^{\text{in}})_{kl}$. Thus, the amount of supporting second-tier resource units is conserved in the resulting resource flow.

It should be noted that resource transfers from the start of an activity in resource flow $\mathcal{F}$ still originate from the start of this activity in the resulting resource flow after a reroute modification has been used while resource transfers from the end of an activity in resource flow $\mathcal{F}$ still originate from the end of this activity in the resulting resource flow.

Example 6.12 We consider a project consisting of $n = 4$ real activities as well as $r = 2$ renewable resources with capacities $R_1 = 3$ and $R_2 = 2$ such that a transfer of resource 2 has to be supported by $\mu_{12} = 1$ unit of resource 1. The processing times p_i and resource requirements r_{ik} as well as the transfer times Δ_{ijk} with $\Delta_{ijk} = \Delta_{ij}$ for $k = 1{,}2$ are given in Table 6.11. Finally, no precedence constraints are given between the real activities.

A feasible resource flow $\mathcal{F}$ for this project is displayed in Figure 6.45 while the corresponding earliest start schedule with the makespan $C_{max} = 9$ is shown in Figure 6.46. Now, we modify this resource flow $\mathcal{F}$ by rerouting an amount of $q = 1$ unit of resource 1 on the arcs $(3^{\text{out}}, 0^{\text{out}}, 1^{\text{in}})_{12}$

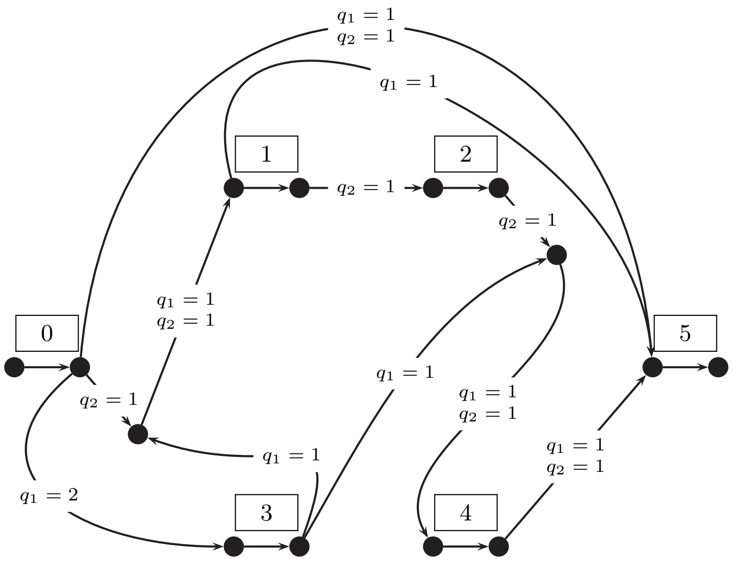


Figure 6.45: Feasible resource flow $\mathcal{F}$ for the project from Example 6.12.

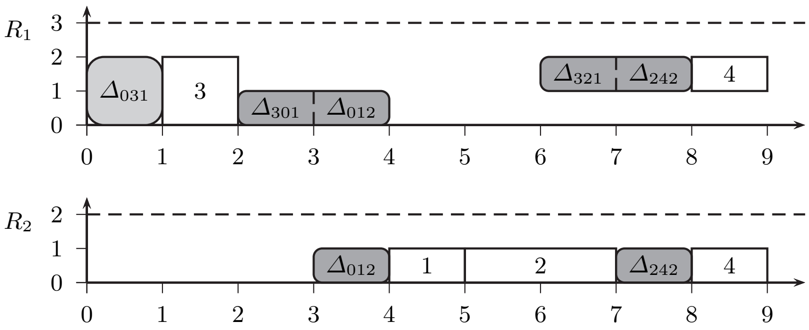


Figure 6.46: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}$ displayed in Figure 6.45.

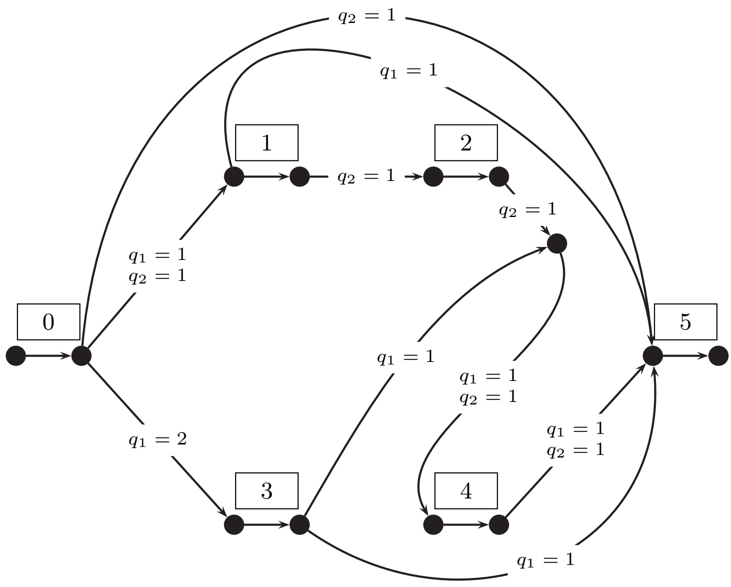


Figure 6.47: Resulting resource flow $\mathcal{F}^{(1)}$ after one unit of resource 1 has been rerouted from activity 3 to activity 5 as well as from activity 0 to activity 1.

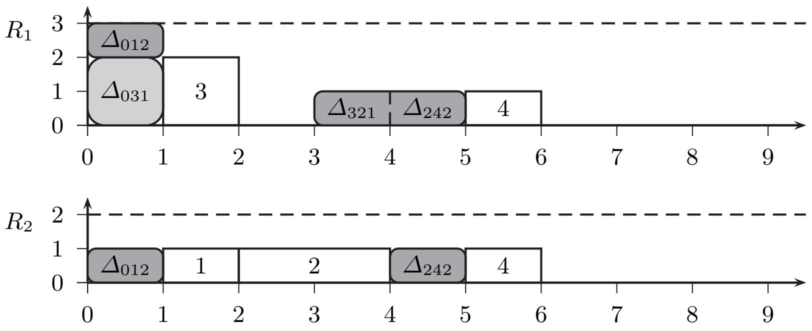


Figure 6.48: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}^{(1)}$ displayed in Figure 6.47.

i	0	1	2	3	4	5
p_i	0	1	2	1	1	0
r_{i1}	3	0	0	2	1	3
r_{i2}	2	1	1	0	1	2

(a) Activity parameters.

	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	1	0
2	1	0	0	1	1	0
3	1	1	1	0	1	0
4	1	1	1	1	0	0
5	0	0	0	0	0	0

(b) Transfer times.

Table 6.11: The processing times p_i and resource requirements r_{ik} of the activities are displayed in (a) while the transfer times $\varDelta_{ijk}$ with $\varDelta_{ijk} = \varDelta_{ij}$ for $k = 1{,}2$ are displayed in (b).

and $(0^{\text{out}}, 5^{\text{in}})_1$. The resulting resource flow $\mathcal{F}^{(1)}$ is shown in Figure 6.47 while the earliest start schedule corresponding to this resource flow with the makespan $C_{max}^{(1)} = 6$ is displayed in Figure 6.48. In this resource flow $\mathcal{F}^{(1)}$, one unit of resource 1 is transferred on the resulting arcs $(3^{\text{out}}, 5^{\text{in}})_1$ and $(0^{\text{out}}, 0^{\text{out}}, 1^{\text{in}})_{12}$, respectively. It should be noted that this latter arc is represented by an arc $(0^{\text{out}}, 1^{\text{in}})_{12}$ in the resulting graph because both resource units originate from the end of activity 0. $\square$

Next, modifications of resource transfers of resources $k \in \mathcal{R}^{\text{sa}}$ in the neighborhood $\tilde{\mathcal{N}}_{\text{reverse}}$ are considered. For a modification in this neighborhood, two arbitrary activities $h \in V$ and $j \in V$ are selected such that activity h is no direct or indirect predecessor of activity j and no other directed path exists from activity h to activity j in the AON-flow network. In this case, up to $2 + 2n \cdot |\mathcal{R}^{\text{ru}}|$ arcs might exist for each resource $k \in \mathcal{R}^{\text{sa}}$ between the two selected activities, i.e. one arc representing a pure first-tier resource transfer $f_{hjk}^{SS} > 0$ from the start of activity h to the start of activity j, one arc representing a pure first-tier resource transfer $f_{hjk}^{FS} > 0$ from the end of activity h to the start of activity j, $n \cdot |\mathcal{R}^{\text{ru}}|$ arcs representing second-tier resource transfers $f_{hijkl}^{SS} > 0$ from the start of activity h to the start of activity j, as well as $n \cdot |\mathcal{R}^{\text{ru}}|$ arcs representing second-tier resource transfers $f_{hijkl}^{FS} > 0$ from the end of activity h to the start of activity j. Then, in order to reverse all arcs between the two selected activities, each of these resource transfers has to be considered separately as follows.

In order to reverse one arc between the two selected activities, additional incoming resource transfers to activity h as well as outgoing resource transfers from activity j have to be modified in order to ensure flow conservation. Unlike before, however, it is not sufficient to select two sets of activities because up to $2 + 2n \cdot |\mathcal{R}^{\text{ru}}|$ arcs representing resource transfers of resource $k \in \mathcal{R}^{\text{sa}}$ might exist between the selected activities (i.e. first-tier and second-tier resource transfers from either the start or the end of activity $u \in V_0$ to the start of activity h as well as first-tier and second-tier resource transfers from either the start or the end of activity j to the start of activity $v \in V_*$). For this reason, we calculate two sets $\tilde{U}_k$ and $\tilde{V}_k$ consisting of a sufficient amount of resource transfers based on priority rules such that the amount of resource units transferred on the corresponding arcs is at least as large as the amount of resource units transferred on the considered arc between the selected activities h and j. Furthermore, no resource transfer can be removed from either of these sets without violating this condition.

Now, the resource transfers are modified as follows by a modification in the neighborhood $\tilde{\mathcal{N}}_{\text{reverse}}$. First of all, a sufficient amount of resource units of resource k are redirected from activities $u \in V_0$ to activity j based on the resource transfers from the set $\tilde{U}_k$. Here, depending on whether the considered arc between the two selected activities h and j represents a first-tier or a second-tier resource transfer, the resulting resource transfers are also first-tier or second-tier resource transfers from the start or the end of activities $u \in V_0$ to the start of activity j. Similarly, a sufficient amount of resource units of resource k is redirected from activity h to activities $v \in V_*$ based on the resource transfers from the set $\tilde{V}_k$. In this case, first-tier resource transfers from either the start or the end of activity j to the start of activity v are replaced by first-tier resource transfers from either the start of activity h (i.e. if the considered arc originates at the start of activity h) or the end of activity h (i.e. if the considered arc originates at the end of activity h) to the start of activity v while second-tier resource transfers are replaced by corresponding second-tier resource transfers.

Finally, the considered arc between the selected activities h and j itself has to be reversed. Here, all first-tier as well as all second-tier resource transfers of resource k from the set $\tilde{U}_k$ that have been redirected to activity j have to be replaced by corresponding first-tier or second-tier resource transfers from either the start or the end of activity j to the start of activity h. The actual amount of resource units of resource k that is transferred from either the start or the end of activity j to the start of activity h depends on the amount of resource units that has previously been transferred from either the start or the end of activity j to activities $v \in V_*$ before the corresponding resource transfers from the set $\tilde{V}_k$ have been modified as described above.

For example, we assume that two activities $h \in V$ and $j \in V$ are selected such that activity h is no predecessor of activity j and no other directed path exists from activity h to activity j in the AON-flow network. Between these two activities, one unit of a resource $k \in \mathcal{R}^{\text{sa}}$ is transferred from the start of activity $h \in V$ to the start of activity $j \in V$ as a first-tier resource (represented by an arc $(h^{\text{in}}, j^{\text{in}})_k$) and one unit of resource k from the end of activity h supports the transfer of a resource $l_1 \in \mathcal{R}^{\text{ru}}$ from the end of activity $i \in V_0$ to the start of activity j as a second-tier resource (represented by an arc $(h^{\text{out}}, i^{\text{out}}, j^{\text{in}})_{kl_1}$). Furthermore, we assume that two units of resource k from the end of activity $u_1 \in V_0$ support the transfer of a resource $l_2 \in \mathcal{R}^{\text{ru}}$ from the end of activity $u_2 \in V_0$ to the start of activity h as a second-tier resource (represented by an arc $(u_1^{\text{out}}, u_2^{\text{out}}, h^{\text{in}})_{kl_2}$). Similarly, we assume that one unit of resource k is transferred from the end of activity j to the start of activity $v_1 \in V_*$ as a first-tier resource (represented by an arc $(j^{\text{out}}, v_1^{\text{in}})_k$) and one unit of resource k from the start of activity j supports the transfer of a resource $l_3 \in \mathcal{R}^{\text{ru}}$ from the end of activity $v_2 \in V_0$ to the start of activity $v_3 \in V_*$ as a second-tier resource (represented by an arc $(j^{\text{in}}, v_2^{\text{out}}, v_3^{\text{in}})_{kl_3}$).

Now, we first reverse the arc $(h^{\text{in}}, j^{\text{in}})_k$ between the two selected activities. For this, we assume that the set $\tilde{U}_k = \{f^{FS}_{u_1u_2hkl_2}\}$ as well as the set $\tilde{V}_k = \{f^{SS}_{jv_2v_3kl_3}\}$ are calculated based on the selected priority rules. Then, one unit of resource k is transferred from the end of activity u_1 to the start of activity j as a first-tier resource (represented by an arc $(u_1^{\text{out}}, j^{\text{in}})_k$), one unit of resource k from the start of activity h supports the transfer of resource l_3 from the end of activity v_2 to the start of activity v_3 as a second-tier resource (represented by an arc $(h^{\text{in}}, v_2^{\text{out}}, v_3^{\text{in}})_{kl_3}$), and one unit of resource k from the start of activity j supports the transfer of resource l_2 from the end of activity u_2 to the start of activity h as a second-tier resource (represented by an arc $(j^{\text{in}}, u_2^{\text{out}}, h^{\text{in}})_{kl_2}$).

Next, we reverse the arc $(h^{\text{out}}, i^{\text{out}}, j^{\text{in}})_{kl_1}$ between nodes h^{out} and j^{in}. Here, we assume that the set $\tilde{U}_k = \{f^{FS}_{u_1u_2hkl_2}\}$ as well as the set $\tilde{V}_k = \{f^{FS}_{jv_1k}\}$ are calculated based on the selected priority rules. Then, one unit of resource k from the end of activity u_1 supports the transfer of resource l_1 from the end of activity i to the start of activity j as a second-tier resource (represented by an arc $(u_1^{\text{out}}, i^{\text{out}}, j^{\text{in}})_{kl_1}$), one unit of resource k is transferred from the end of activity h to the start of activity v_1 as a first-tier resource (represented by an arc $(h^{\text{out}}, v_1^{\text{in}})_k$), and one unit of resource k from the end of activity j supports the transfer of resource l_2 from the end of activity u_2 to the start of activity h as a second-tier resource (represented by an arc $(j^{\text{out}}, u_2^{\text{out}}, h^{\text{in}})_{kl_2}$).

If more than one arc representing a resource transfer of a resource $k \in \mathcal{R}^{\mathrm{sa}}$ exists between the two selected activities $h \in V$ and $j \in V$, it should be noted that these arcs have to be reversed one after the other as performed in the example above. Otherwise, if all arcs were reversed simultaneously, the sets $\tilde{U}_k$ and $\tilde{V}_k$ that are calculated for each of these arcs might contain resource transfers such that the same resource units have to be redirected for multiple arcs. In the following, we assume that the arcs in this case are reversed in an arbitrary order such that always the intermediate resource flow is considered in order to calculate the sets $\tilde{U}_k$ and $\tilde{V}_k$.

Example 6.13 We again consider the project from Example 6.12. Now, we modify the feasible resource flow $\mathcal{F}$ displayed in Figure 6.45 by reversing the arc $(3^{\mathrm{out}}, 0^{\mathrm{out}}, 1^{\mathrm{in}})_{12}$ between activities 3 and 1 based on a modification in the neighborhood $\tilde{\mathcal{N}}_{\mathrm{reverse}}$. The resulting resource flow $\mathcal{F}^{(2)}$ is displayed in Figure 6.49 while the corresponding earliest-start schedule with the makespan $C^{(2)}_{max} = 6$ is shown in Figure 6.50.

In the resulting resource flow $\mathcal{F}^{(2)}$, one unit of resource 1 of the outgoing resource transfer from activity 1 to activity 5 (represented by an arc $(1^{\mathrm{in}}, 5^{\mathrm{in}})_1$) is redirected from activity 3 to activity 5 (represented by an arc $(3^{\mathrm{out}}, 5^{\mathrm{in}})_1$). Similarly, one unit of resource 1 of the incoming resource transfer from activity 0 to activity 3 (represented by an arc $(0^{\mathrm{out}}, 3^{\mathrm{in}})_1$) is redirected from activity 0 to support the transfer of resource 2 from activity 0 to activity 1 (represented by an arc $(0^{\mathrm{out}}, 0^{\mathrm{out}}, 1^{\mathrm{in}})_{12}$). As before, this latter arc is represented by an arc $(0^{\mathrm{out}}, 1^{\mathrm{in}})_{12}$ in the resulting graph. Finally, the selected arc is reversed, i.e. one unit of resource 1 is transferred from activity 1 to activity 3 as a first-tier resource (represented by an arc $(1^{\mathrm{in}}, 3^{\mathrm{in}})_1$). □

Next, we consider resource transfers of resources $l \in \mathcal{R}^{\mathrm{ru}}$. These resources can only be transferred as first-tier resources from the end of an activity to the start of another activity. For this reason, arcs $(i^{\mathrm{out}}, j^{\mathrm{in}})_l$ representing resource transfers of first-tier resources $l \in \mathcal{R}^{\mathrm{ru}}$ between activities $i \in V_0$ and $j \in V_*$ can be handled as described for the classical RCPSP in both neighborhoods $\tilde{\mathcal{N}}_{\mathrm{reroute}}$ and $\tilde{\mathcal{N}}_{\mathrm{reverse}}$. Additionally to rerouting or reversing these arcs, however, it is also necessary to adapt the supporting second-tier resource transfers of resources $k \in \mathcal{R}^{\mathrm{sa}}$. In the following, four different cases have to be considered depending on the transfer times between the activities. Here, without loss of generality, we assume that q_l units of resource $l \in \mathcal{R}^{\mathrm{ru}}$ are transferred from activity $i \in V_0$ to activity $j \in V_*$ before the modification and from activity $u \in V_0$ to activity j after the modification. The transfer times of these resource transfers are denoted by Δ_{ijl} and Δ_{ujl}, respectively.

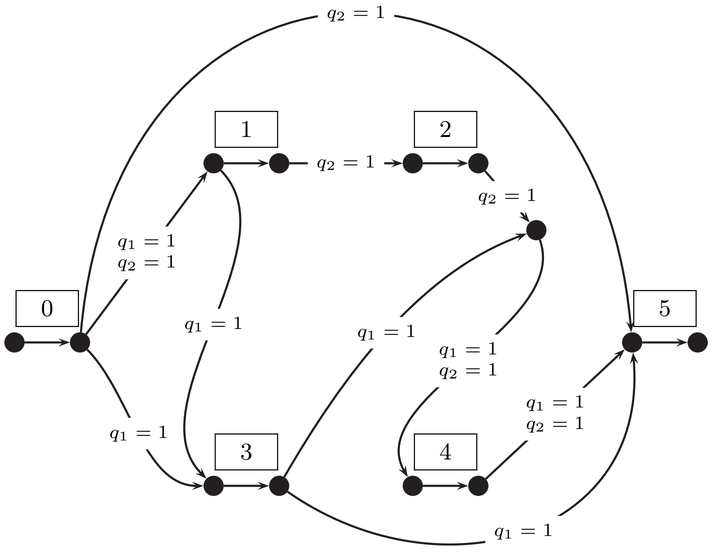


Figure 6.49: Resulting resource flow $\mathcal{F}^{(2)}$ after the arc $(3^{\text{out}}, 0^{\text{out}}, 1^{\text{in}})_{12}$ between activities 3 and 1 has been reversed.

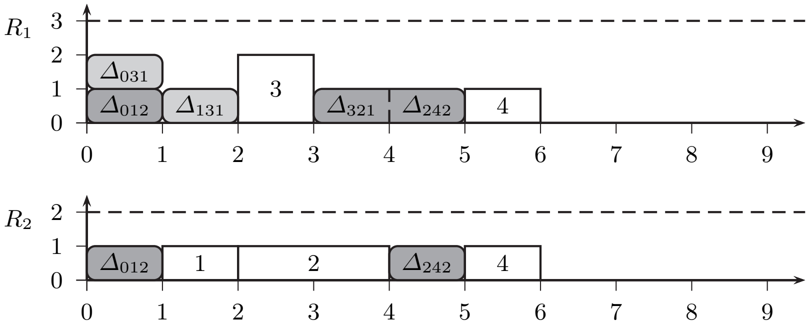
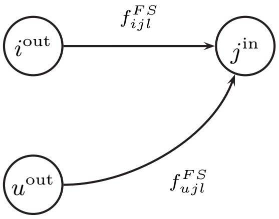
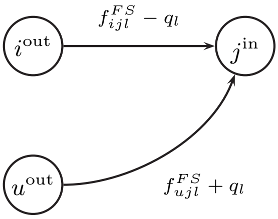


Figure 6.50: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}^{(2)}$ displayed in Figure 6.49.

- If $\varDelta_{ijl} = \varDelta_{ujl} = 0$ holds, no supporting resources $k \in \mathcal{R}^{\text{sa}}$ are required either before or after the modification. Thus, no additional adaptions of the resource flow are required (cf. Figure 6.51).



(a) Before modification.



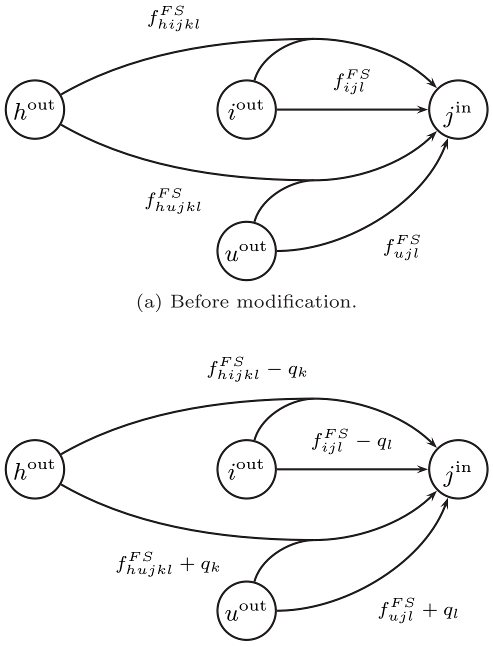
(b) After modification.

Figure 6.51: If $\varDelta_{ijl} = \varDelta_{ujl} = 0$ holds, no supporting resources $k \in \mathcal{R}^{\text{sa}}$ are required either before or after the modification.

- If $\varDelta_{ijl} > 0$ and $\varDelta_{ujl} > 0$ hold, supporting resources $k \in \mathcal{R}^{\text{sa}}$ are required before as well as after the modification. In this case, it is sufficient to redirect resource units of second-tier resources $k \in \mathcal{R}^{\text{sa}}$ that have previously been used to support the transfer of resource l from activity i to activity j to now support the transfer of resource l from activity u to activity j. For this, a set $\tilde{U}_k$ is calculated for each resource $k \in \mathcal{R}^{\text{sa}}$ with $\mu_{kl} > 0$ consisting of a sufficient amount of second-tier resource transfers of resource k from activities $h \in V_0$ that support the transfer of resource l from activity i to activity j. These resource transfers are then redirected to support the transfer of resource l from activity u to activity j. As before, the sets $\tilde{U}_k$ are calculated based on priority rules.

  Below, this situation is shown in Figure 6.52. Here, we assume that a sufficient amount of $q_k = q_l \cdot \mu_{kl}$ units of second-tier resource $k \in \mathcal{R}^{\text{sa}}$ from the end of activity $h \in V_0$ support the transfer of q_l units of resource l from activity i to activity j before the modification (i.e. the set $\tilde{U}_k = \{f^{FS}_{hijkl}\}$ can be calculated). Then, after the modification, q_k units of resource k from the end of activity h are used to support the transfer of q_l units of resource l from activity u to activity j instead.

- If $\varDelta_{ijk} > 0$ and $\varDelta_{ujl} = 0$ hold, supporting resources $k \in \mathcal{R}^{\text{sa}}$ are required before the modification but not after the modification. Due to this, an amount of $q_k = q_l \cdot \mu_{kl}$ units of resource $k \in \mathcal{R}^{\text{sa}}$ is no longer



(a) Before modification.

(b) After modification.

Figure 6.52: If $\Delta_{ijl} > 0$ and $\Delta_{ujl} > 0$ hold, the same units of resource $k \in \mathcal{R}^{\text{sa}}$ can be used both before as well as after the modification.

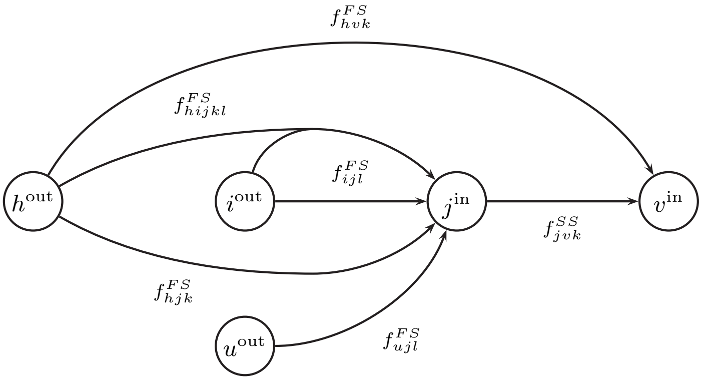
required to support the transfer of resource l to activity j. Now, if an amount of $\tilde{q}_k$ units of resource k is transferred from activities $h \in V_0$ to activity j as pure second-tier resource (i.e. these resource units are not required to process activity j and can be transferred from activity j to activities $v \in V_*$ by start-to-start transfer), an amount of $q^1_k = \min\{q_k, \tilde{q}_k\}$ units of resource k is no longer required by activity j. Instead, these resource units can be transferred directly from activities h to activities $v \in V_*$ where activities v receive resource k from activity j by start-to-start transfer before the modification.

In this case, we calculate a set $\tilde{U}^1_k$ as well as a set $\tilde{V}_k$ for each resource $k \in \mathcal{R}^{\text{sa}}$ with $q^1_k > 0$ such that the set $\tilde{U}^1_k$ contains a sufficient amount of second-tier resource transfers from activities $h \in V_0$ that support the transfer of resource l from activity i to activity j and the set $\tilde{V}_k$ contains a sufficient amount of first-tier or second-tier resource transfers from the start of activity j to activities $v \in V_*$. Then, resource transfers of resource k from the set $\tilde{U}_k^1$ are redirected from the corresponding activities $h \in V_0$ to activities $v \in V_*$ as denoted by the set $\tilde{V}_k$. Here, depending on whether resource k is transferred from the start of activity j to the start of activity v as a first-tier or as a second-tier resource before the modification, resource k is transferred from activity h to activity v as either a first-tier or a second-tier resource after the modification, respectively.

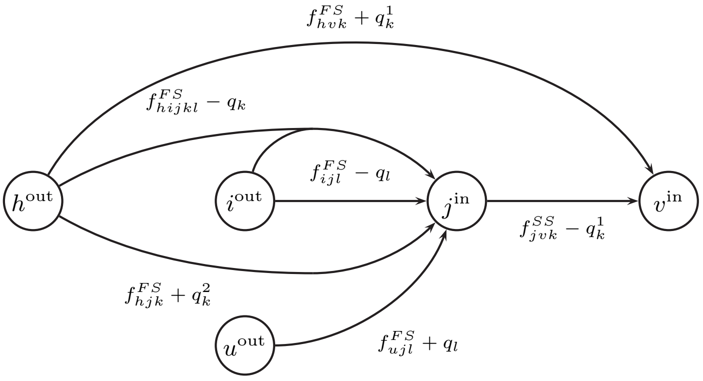
Finally, if $q_k^1 < q_k$ holds, an amount of $q_k^2 = \max\{0, q_k - q_k^1\}$ units of resource k can be transferred directly from activities h to activity j as first-tier resources. For this, another set $\tilde{U}_k^2$ is calculated for each resource $k \in \mathcal{R}^{\text{sa}}$ with $q_k^2 > 0$ such that this set contains a sufficient amount of second-tier resource transfers from activities $h \in V_0$ that support the transfer of resource l from activity i to activity j. These resource transfers are then redirected such that an amount of q_k^2 units of resource k is transferred from activities $h \in V_0$ to activity j as first-tier resources. It should be noted that all sets $\tilde{U}_k^1$, $\tilde{U}_k^2$, and $\tilde{V}_k$ are again calculated based on priority rules.

This situation is visualized in Figure 6.53. We again assume that a sufficient amount of $q_k = q_l \cdot \mu_{kl}$ units of resource k from the end of activity $h \in V_0$ supports the transfer of q_l units of resource l from activity i to activity j before the modification (i.e. the sets $\tilde{U}_k^1 = \tilde{U}_k^2 = \{f_{hijkl}^{FS}\}$ can be calculated based on the selected priority rule). Furthermore, we assume that all $\tilde{q}_k$ units of pure second-tier resource k transferred to activity j are transferred from the start of activity j to an activity $v \in V_*$ as first-tier resources (i.e. the set $\tilde{V}_k = \{f_{jvk}^{SS}\}$ can be calculated based on the selected priority rule). Then, because q_k units of resource k are no longer required to support the transfer of q_l units of resource l from activity u to activity j after the modification, an amount of $q_k^1 = \min\{q_k, \tilde{q}_k\}$ units of resource k is instead transferred from the end of activity h to the start of activity v (based on the sets $\tilde{U}_k^1$ and $\tilde{V}_k$) while an amount of $q_k^2 = \max\{0, q_k - q_k^1\}$ units of resource k is transferred from the end of activity h to the start of activity j as first-tier resources (based on the set $\tilde{U}_k^2$).

- If $\Delta_{ijl} = 0$ and $\Delta_{ujl} > 0$ hold, no supporting resources $k \in \mathcal{R}^{\text{sa}}$ are required before the modification but are required after the modification.



(a) Before modification.



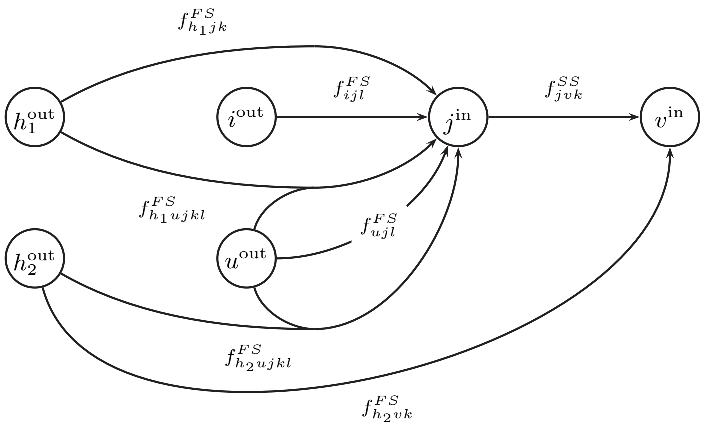
(b) After modification.

Figure 6.53: If $\varDelta_{ijl} > 0$ and $\varDelta_{ujl} = 0$ hold, the supporting second-tier resources $k \in \mathcal{R}^{\mathrm{sa}}$ are no longer required and can either be transferred to other activities $v \in V_*$ (if they have been transferred to activity j as pure second-tier resources) or they can be transferred to activity j as first-tier resources.

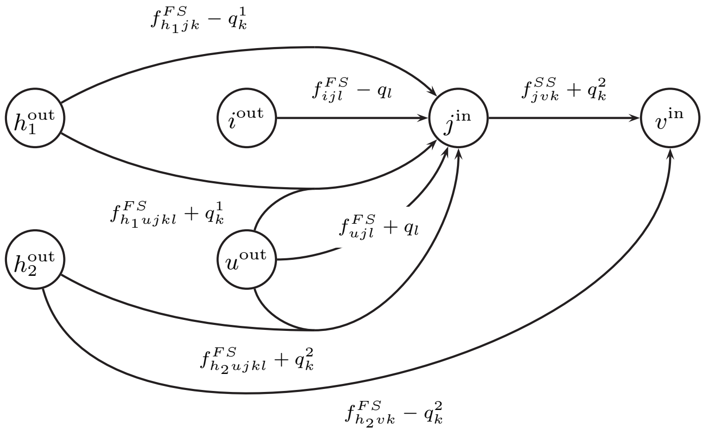
In this case, an amount of $q_k = q_l \cdot \mu_{kl}$ units of resource $k \in \mathcal{R}^{\mathrm{sa}}$ has to be assigned to support the transfer of resource l from activity u to activity j. Now, if an amount of $\tilde{q}_k$ units of resource k is transferred from activities $h \in V_0$ to activity j as first-tier resources (i.e. these resource units are not required to support the transfer of resources $l \in \mathcal{R}^{\mathrm{ru}}$ to activity j), an amount of $q_k^1 = \min\{q_k, \tilde{q}_k\}$ of these resource units can be used to support the transfer of resource l from activity u to activity j. For this, a set $\tilde{U}_k^1$ consisting of a sufficient amount of first-tier resource transfers from activities $h \in V_0$ to activity j is calculated. These resource transfers are then redirected to support the transfer of resource l from activity u to activity i.

If this amount of resource units of resource k is not sufficient (i.e. if $q_k^1 < q_k$ holds), an additional amount of $q_k^2 = \max\{0, q_k - q_k^1\}$ units of resource k has to be redirected to activity j as pure second-tier resources to support the transfer of resource l from activity u to activity j. For this, a set $\tilde{U}_k^2$ consisting of a sufficient amount of either first- or second-tier resource transfers of resource k from activities $h \in V_0$ to activities $v \in V_*$ are selected such that activity h may not be a (direct or indirect) successor of activities u and j and activity v may not be a (direct or indirect) predecessor of activities u and j. These resource transfers are then redirected such that an amount of q_k^2 units of resource k from activities $h \in V_0$ supports the transfer of resource l from activity u to activity j as pure second-tier resources. From there, these resource units of resource k are then transferred to the corresponding activities v by start-to-start transfer. Depending on whether a modified resource transfer between activities h and v has been a first-tier or a second-tier resource transfer before the modification, the resource units of resource k are transferred from the start of activity j to activity v by either first-tier or second-tier resource transfer after the modification, respectively. As before, the sets $\tilde{U}_k^1$ and $\tilde{U}_k^2$ are calculated based on the selected priority rule.

Below, this situation is shown in Figure 6.54. Here, an amount of q_l units of resource l are transferred from activity i to activity j without the aid of supporting resources $k \in \mathcal{R}^{\mathrm{sa}}$ before the modification. After the modification, however, an amount of $q_k = q_l \cdot \mu_{kl}$ units of resource k is required to support the transfer of resource l from activity u to activity j. In order to supply these resource units, we assume that an amount of $\tilde{q}_k$ units of first-tier resource k is transferred from the end of activity $h_1 \in V_0$ to activity j such that no further pure first-tier resource units of resource k are transferred to activity j (i.e. the set $\tilde{U}_k^1 = \{f_{h_1jk}^{FS}\}$ can be calculated based on the selected priority rule). Additionally, we assume that an amount of at least $q_k^2 = \max\{0, q_k - \min\{q_k, \tilde{q}_k\}\}$ units of resource k is transferred from the end of activity



(a) Before modification.



(b) After modification.

Figure 6.54: If $\mathit{\Delta}_{ijl} = 0$ and $\mathit{\Delta}_{ujl} > 0$ hold, supporting second-tier resources $k \in \mathcal{R}^{\text{sa}}$ are required to support the transfer of resource l from activity u to activity j after the modification. These second-tier resource units can either be redirected to activity j based on existing first-tier resource transfers of resource k to activity j or as additional pure second-tier resources transfers.

$h_2 \in V_0$ to activity v by first-tier resource transfer such that no directed path exists from either activity j or activity u to activity h_2 and no directed path exists from activity v to either activity j or activity u (i.e. the set $\tilde{U}_k^2 = \{f_{h_2vk}^{FS}\}$ can be calculated).

Then, after the modification, an amount of $q_k^1 = \min\{q_k, \tilde{q}_k\}$ units of resource k from activity h_1 is used to support the transfer of resource l from activity u to activity j. Additionally, an amount of $q_k^2 = \max\{0, q_k - q_k^1\}$ units of resource k from activity h_2 is redirected to support the transfer of resource l from activity u to activity j. As these latter resource units are transferred as pure second-tier resource units to the start of activity j, they can then immediately be transferred from the start of activity j to the start of activity v.

These four cases can be applied for all resource transfers of resources $l \in \mathcal{R}^{\mathrm{ru}}$ that are modified based on a modification in either neighborhood $\tilde{\mathcal{N}}_{\mathrm{reroute}}$ or $\tilde{\mathcal{N}}_{\mathrm{reverse}}$. As a modification in neighborhood $\tilde{\mathcal{N}}_{\mathrm{reverse}}$ might modify both, resource transfers of resources $k \in \mathcal{R}^{\mathrm{sa}}$ as well as resource transfers of resources $l \in \mathcal{R}^{\mathrm{ru}}$ in a single move, resource transfers of resources $k \in \mathcal{R}^{\mathrm{sa}}$ between the two selected activities $i \in V$ and $j \in V$ are adapted first. Afterward, all resource transfers of resources $l \in \mathcal{R}^{\mathrm{ru}}$ between the two activities i and j are adapted based on the resulting intermediate resource flow.

Example 6.14 We again consider the project from Example 6.12. Now, we modify the feasible resource flow $\mathcal{F}^{(1)}$ displayed in Figure 6.47 by rerouting one unit of resource 2 on the arcs $(2^{\mathrm{out}}, 4^{\mathrm{in}})_2$ and $(0^{\mathrm{out}}, 5^{\mathrm{in}})_2$ based on a modification in the neighborhood $\tilde{\mathcal{N}}_{\mathrm{reroute}}$. The resulting resource flow $\mathcal{F}^{(3)}$ is displayed is Figure 6.55 while the corresponding earliest start schedule with the makespan $C_{max}^{(3)} = 5$ is shown in Figure 6.56.

Here, because $\Delta_{242} > 0$ and $\Delta_{042} > 0$ hold for the transfer time of resource 2 to activity 4, the same resource unit of resource 1 can be used to support the transfer of resource 2 in the resulting resource flow $\mathcal{F}^{(3)}$ as in resource flow $\mathcal{F}^{(1)}$. Thus, one unit of resource 1 from the end of activity 3 supports the transfer of resource 2 from activity 0 to activity 4. Similarly, because $\Delta_{452} = 0$ and $\Delta_{252} = 0$ hold for the transfer time of resource 2 to dummy activity 5, no supporting resource units of resource 1 are required either before or after the modification.

Finally, we modify resource flow $\mathcal{F}^{(1)}$ from Figure 6.47 by reversing the arc $(1^{\mathrm{out}}, 2^{\mathrm{in}})_2$ between activities 1 and 2 based on a modification in the neighborhood $\tilde{\mathcal{N}}_{\mathrm{reverse}}$. The resulting resource flow $\mathcal{F}^{(4)}$ is displayed in Figure 6.57

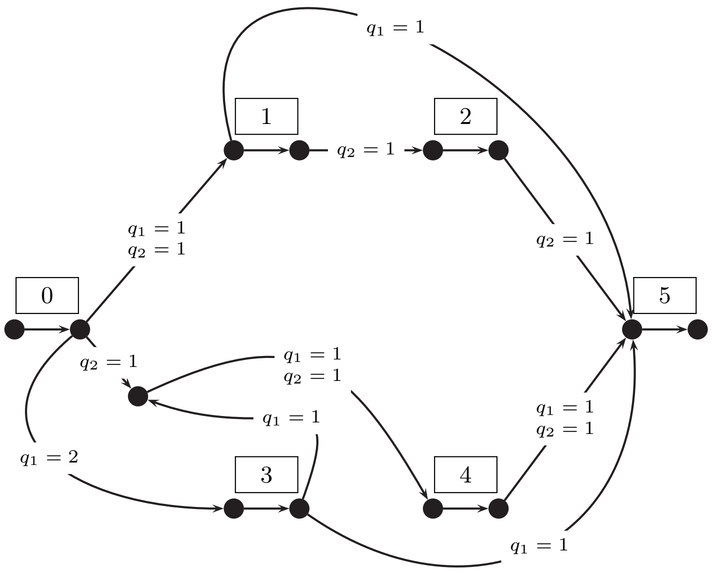


Figure 6.55: Resulting resource flow $\mathcal{F}^{(2)}$ after one unit of resource 2 has been rerouted from activity 2 to activity 5 as well as from activity 0 to activity 4.

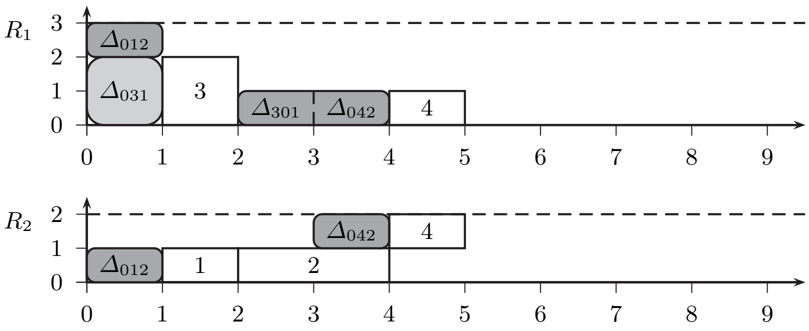


Figure 6.56: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}^{(3)}$ displayed in Figure 6.55.

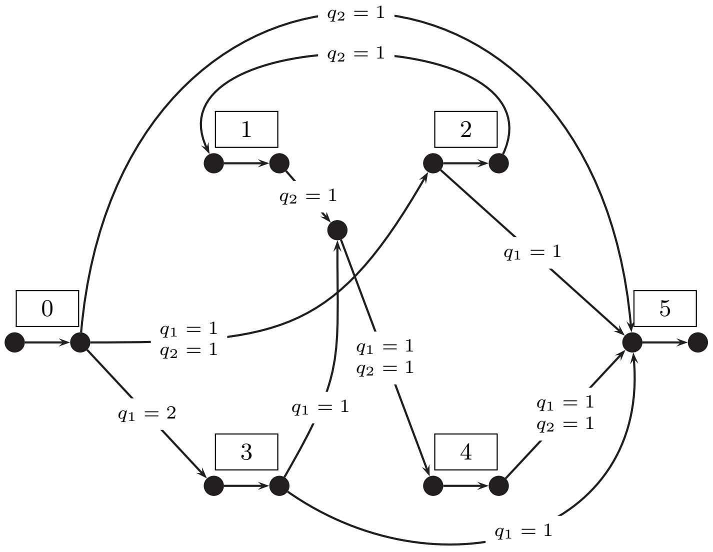


Figure 6.57: Resulting resource flow $\mathcal{F}^{(4)}$ after the arc $(1^{\text{out}}, 2^{\text{in}})_2$ between activities 1 and 2 has been reversed.

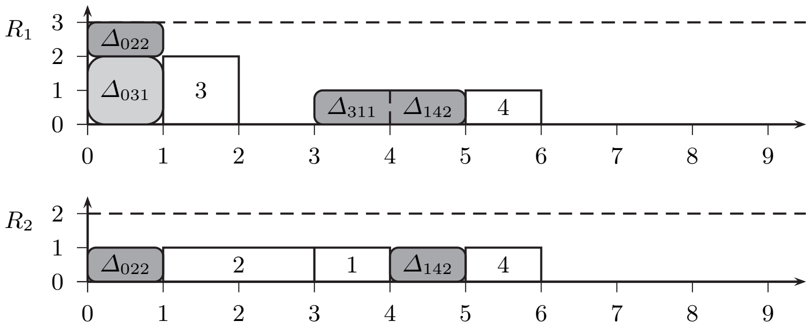


Figure 6.58: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}^{(4)}$ displayed in Figure 6.57.

while the corresponding earliest start schedule with the makespan $C_{max}^{(4)} = 6$ is shown in Figure 6.58.

Here, multiple steps are necessary in order to apply this modification. First of all, the incoming resource transfers to activity 2 have to be adapted. For this, one unit of resource 2 on the arc $(0^{\text{out}}, 1^{\text{in}})_2$ is redirected from activity 0 to activity 2. In this case, because no supporting resources have been required to support the transfer of resource 2 from activity 1 to activity 2 before the modification (i.e. due to $\Delta_{122} = 0$) but one unit of resource 1 is required to support the transfer of resource 2 from activity 0 to activity 2 after the modification (i.e. due to $\Delta_{022} = 1$), one unit of resource 1 has to be redirected to activity 2 as a pure second-tier resource. For this, the arc $(0^{\text{out}}, 0^{\text{out}}, 1^{\text{in}})_{12}$ (which is represented by an arc $(0^{\text{out}}, 1^{\text{in}})_{12}$ in the actual graph) is selected based on a priority rule that selects resource transfers according to increasing numbers (here, the activities from which the resource transfer of resource 1 originates are considered) and one unit of resource 1 is transferred from activity 0 to activity 2 to support the transfer of resource 2. Furthermore, in an intermediate resource flow, this pure second-tier resource unit is transferred from the start of activity 2 to the start of activity 1 on the temporary arc $(2^{\text{in}}, 0^{\text{out}}, 1^{\text{in}})_{12}$.

Next, the arc $(1^{\text{out}}, 2^{\text{in}})_2$ is reversed such that one unit of resource 2 is transferred from activity 2 to activity 1. In this case, one unit of resource 1 has been required to support the transfer of resource 2 from activity 0 to activity 1 before the modification (i.e. due to $\Delta_{012} = 1$) but no supporting resources are required after the modification (i.e. due to $\Delta_{212} = 0$). Thus, one unit of resource 1 that has been transferred to activity 1 as a pure second-tier resource from the start of activity 2 on the temporary arc $(2^{\text{in}}, 0^{\text{out}}, 1^{\text{in}})_{12}$ in the intermediate resource flow is no longer required. Instead, it can immediately be transferred from the start of activity 2 to activity 5.

Finally, the outgoing resource transfers from activity 1 have to be adapted. Here, one unit of resource 2 on the arc $(2^{\text{out}}, 4^{\text{in}})_2$ is redirected from activity 1 to activity 4. In this case, because one unit of resource 1 is required to support the transfer of resource 2 to activity 4 both before as well as after the modification (i.e. due to $\Delta_{242} > 0$ and $\Delta_{142} > 0$), the same resource unit of resource 1 can be used both before as well as after the modification. For this reason, one unit of resource 1 from activity 3 is used to support the transfer of resource 2 from activity 1 to activity 4. □

As before, the two neighborhoods $\tilde{\mathcal{N}}_{\text{reroute}}$ and $\tilde{\mathcal{N}}_{\text{reverse}}$ are neither connected nor opt-connected on their own. Next, regarding the neighborhood $\tilde{\mathcal{N}}_1 = \tilde{\mathcal{N}}_{\text{reroute}} \cup \tilde{\mathcal{N}}_{\text{reverse}}$, we have been unable to either prove or disprove that the neighborhood is connected or opt-connected, i.e. this question remains an open problem. Now, it is again possible to introduce reduced neighborhoods $\tilde{\mathcal{N}}^{\text{max}}_{\text{reroute}}$, $\tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}}$, and $\tilde{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$ based on the definitions of the corresponding neighborhoods from Section 6.2.2. For this, reroute modifications in the neighborhoods $\tilde{\mathcal{N}}^{\text{max}}_{\text{reroute}}$ and $\tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}}$ are again restricted to always reroute the maximal amount of resource units between the two selected arcs. Furthermore, both neighborhoods $\tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}}$ and $\tilde{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$ are limited to modifications between critical activities.

Additionally, if a hyperarc $(h^{\text{out}}, i^{\text{out}}, j^{\text{in}})_{kl}$ or a hyperarc $(h^{\text{in}}, i^{\text{out}}, j^{\text{in}})_{kl}$ lies on a critical path, both, the resource transfer of resource $k \in \mathcal{R}^{\text{sa}}$ as denoted by this hyperarc as well as the resource transfer of the supported resource $l \in \mathcal{R}^{\text{ru}}$ as denoted by the arc $(i^{\text{out}}, j^{\text{in}})_l$ are considered for modification in the neighborhoods $\tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}}$ and $\tilde{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$ even if the arc $(i^{\text{out}}, j^{\text{in}})_l$ is not a critical arc itself. Otherwise, situations might arise in which a feasible resource flow $\mathcal{F}$ can not be transformed into an optimal resource flow $\mathcal{F}^*$ by modifications in the resulting neighborhood $\tilde{\mathcal{N}}_2 = \tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}} \cup \tilde{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$.

Example 6.15 We consider a project consisting of $n = 3$ activities and $r = 2$ renewable resources with capacities $R_1 = 1$ and $R_2 = 2$ such that an amount of $\mu_{12} = 1$ unit of resource 1 is required to support the transfer of one unit of resource 2. The processing times p_i and resource requirements r_{ik} of the activities as well as the transfer times Δ_{ijk} with $\Delta_{ijk} = \Delta_{ij}$ for both resources $k = 1{,}2$ are given in Table 6.12. Finally, only the precedence constraints $1 \to 3$ and $2 \to 3$ are given between the real activities.

i	0	1	2	3	4
p_i	0	1	2	1	0
r_{i1}	1	0	1	1	1
r_{i2}	2	1	0	1	2

(a) Activity parameters.

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	0
4	0	0	0	0	0

(b) Transfer times.

Table 6.12: The processing times p_i and resource requirements r_{ik} of the activities are displayed in (a) while the transfer times Δ_{ijk} with $\Delta_{ijk} = \Delta_{ij}$ for $k = 1{,}2$ are displayed in (b).

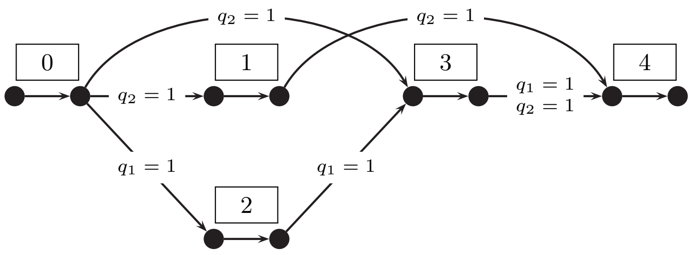


Figure 6.59: Optimal resource flow $\mathcal{F}^*$ for the project from Example 6.15.

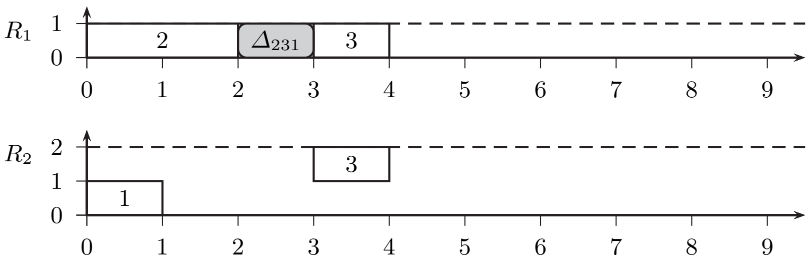


Figure 6.60: The earliest start schedule corresponding to the optimal resource flow $\mathcal{F}^*$ displayed in Figure 6.59.

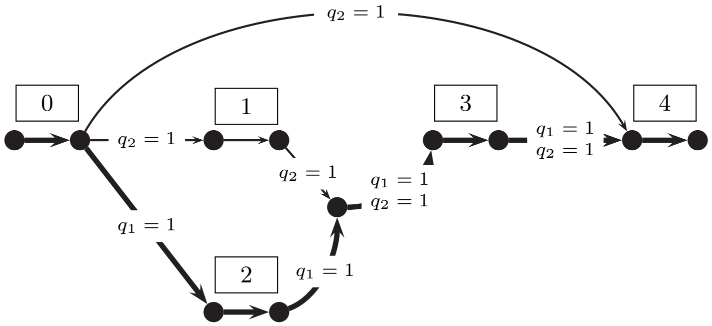


Figure 6.61: Feasible resource flow $\mathcal{F}$ for the project from Example 6.15. The unique critical path is highlighted in this graph.

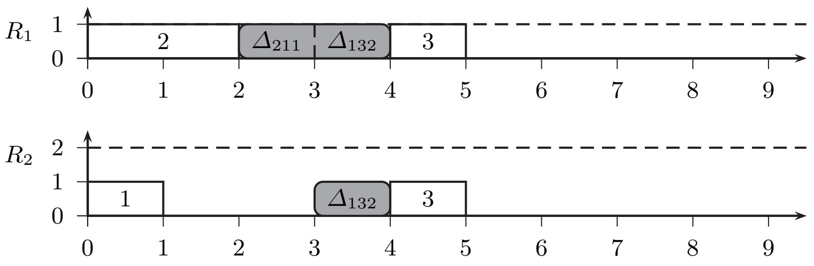


Figure 6.62: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}$ displayed in Figure 6.61.

The unique optimal resource flow $\mathcal{F}^*$ for this project is displayed in Figure 6.59 while the earliest start schedule corresponding to this resource flow with the makespan $C^*_{max} = 4$ is shown in Figure 6.60. Another feasible resource flow $\mathcal{F}$ for this project as well as the corresponding earliest start schedule with the makespan $C_{max} = 5$ are given in Figures 6.61 and 6.62, respectively.

Here, in resource flow $\mathcal{F}$, the arcs $(0^{\text{out}}, 2^{\text{in}})_1$, $(2^{\text{out}}, 1^{\text{out}}, 3^{\text{in}})_{12}$, $(3^{\text{out}}, 4^{\text{in}})_1$, and $(3^{\text{out}}, 4^{\text{in}})_2$ constitute the unique critical path. None of these arcs can be modified using either a reroute or a reverse move, i.e. it is not possible to transform this resource flow into the optimal resource flow $\mathcal{F}^*$ if only the actual critical arcs are considered. Instead, it is necessary to reroute the two non-critical arcs $(1^{\text{out}}, 3^{\text{in}})_2$ and $(0^{\text{out}}, 4^{\text{in}})_2$ in order to transform resource flow $\mathcal{F}$ into resource flow $\mathcal{F}^*$. $\square$

Now, similar to the RCPSP with first-tier resource transfers, Theorem 6.6 does not hold for the neighborhood $\tilde{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$ introduced above. Thus, neighborhood $\tilde{\mathcal{N}}_2$ is not opt-connected (and hence also not connected) for the RCPSP with first- and second-tier resource transfers. Unlike for the RCPSP with first-tier resource transfers, however, this result also holds if all resources $k \in \mathcal{R}$ have the same transfer times $\Delta_{hjk} = \Delta_{hj}$ between activities $h \in V_0$ and $j \in V_*$ and the triangle inequality $\Delta_{hik} + \Delta_{ijk} \geq \Delta_{hjk}$ holds for these transfer times. This follows from the fact that two critical activities $h \in V$ and $j \in V$ may be connected by a critical arc from the end of activity h to the start of activity j as well as by a non-critical directed path from the start of activity h to the start of activity j via at least one other activity $i \in V$. Here, the directed path from the start of activity h may be shorter than the critical arc from the end of activity h to the start of activity j because the processing time p_h of activity h does not contribute to the length of the path.

Example 6.16 We consider a project consisting of $n = 4$ real activities as well as $r = 4$ renewable resources with capacities $R_k = 1$ for $k = 1,2,3,4$ such that an amount of $\mu_{12} = 1$ unit of resource 1 is required in order to support the transfer of one unit of resource 2. The processing times p_i and resource requirements r_{ik} of the activities as well as the transfer times Δ_{ijk} with $\Delta_{ijk} = \Delta_{ij}$ for $k = 1,2,3,4$ are given in Table 6.13. Finally, only the precedence constraint $3 \to 4$ is given between the real activities.

i	0	1	2	3	4	5
p_i	0	4	1	1	1	0
r_{i1}	1	0	1	1	0	1
r_{i2}	1	1	0	0	0	1
r_{i3}	1	1	0	1	0	1
r_{i4}	1	0	0	1	1	1

(a) Activity parameters.

	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	1	1	1	0
2	1	1	0	1	1	0
3	1	1	1	0	1	0
4	1	1	1	1	0	0
5	∞	∞	∞	∞	∞	0

(b) Transfer times.

Table 6.13: The processing times p_i and resource requirements r_{ik} of the activities are displayed in (a) while the transfer times Δ_{ijk} with $\Delta_{ijk} = \Delta_{ij}$ for $k = 1,2,3,4$ are displayed in (b).

The unique optimal resource flow $\mathcal{F}^*$ for this project is displayed in Figure 6.63 while the earliest start schedule corresponding to this resource flow with the makespan $C^*_{max} = 8$ is shown in Figure 6.64. Another feasible resource flow $\mathcal{F}$ for this project as well as the corresponding earliest start schedule with the makespan $C_{max} = 9$ are given in Figures 6.65 and 6.66, respectively.

In this resource flow $\mathcal{F}$, only the arc $(1^{\text{out}}, 3^{\text{in}})_3$ is eligible for a modification in the neighborhood $\tilde{\mathcal{N}}_2$. This is due to the fact that all resources $k = 1,2,3,4$ have unit capacity such that no modification in the neighborhood $\tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}}$ is possible and only the order of activities 1 and 3 on the critical path is not predetermined by precedence constraints. The arc $(1^{\text{out}}, 3^{\text{in}})_3$ can not be reversed, however, because another path from the start of activity 1 to the start of activity 3 via activity 2 exists, i.e. a cycle $(1^{\text{in}}, 2^{\text{in}}, 2^{\text{out}}, 3^{\text{in}}, 3^{\text{out}}, 1^{\text{in}})$ would result from a reverse modification. Thus, resource flow $\mathcal{F}$ can not be

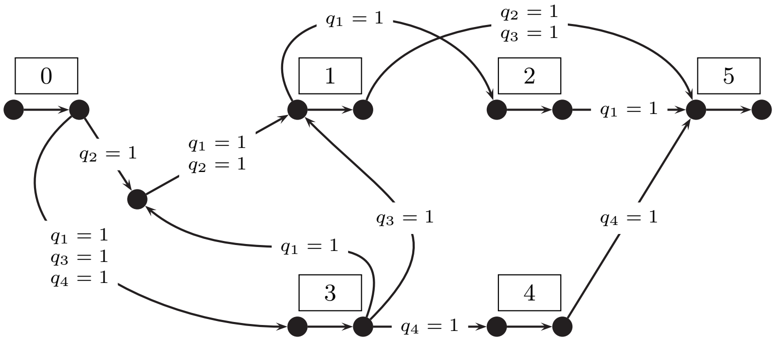


Figure 6.63: Optimal resource flow $\mathcal{F}^*$ for the project from Example 6.16.

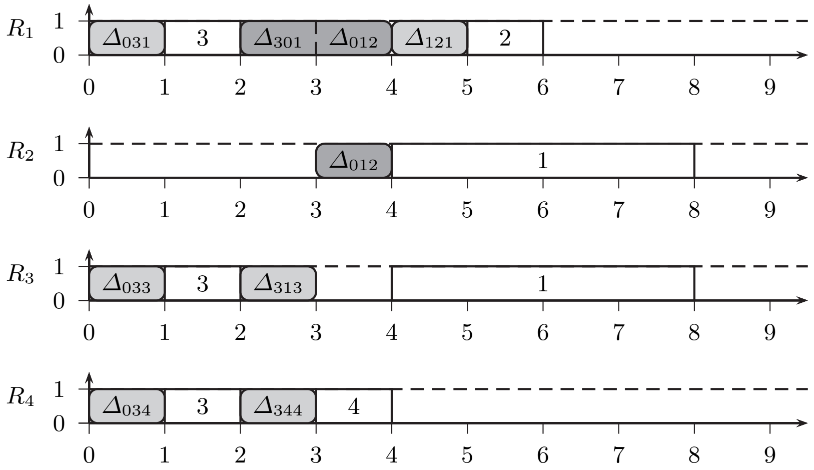


Figure 6.64: The earliest start schedule corresponding to the optimal resource flow $\mathcal{F}^*$ displayed in Figure 6.63.

transformed into the optimal resource flow $\mathcal{F}^*$ based on modifications in the neighborhood $\tilde{\mathcal{N}}_2$. Instead, it would be necessary to first reverse the arc $(2^{\text{out}}, 3^{\text{in}})_1$ based on a modification in the neighborhood $\tilde{\mathcal{N}}_{\text{reverse}}$ and then to reverse both resulting arcs between activities 1 and 3. □

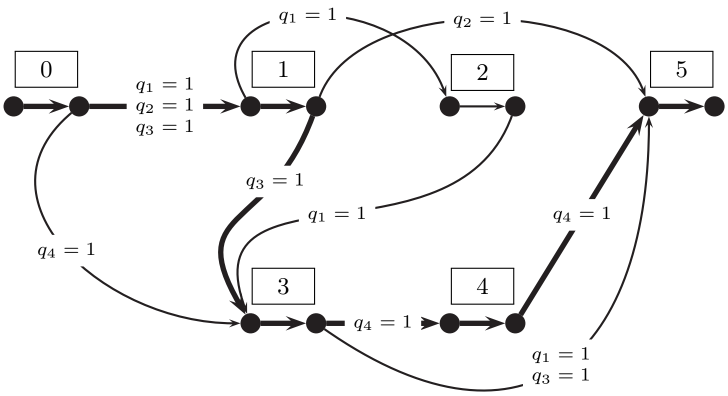


Figure 6.65: Feasible resource flow $\mathcal{F}$ for the project from Example 6.16. The critical arcs are highlighted in this graph.

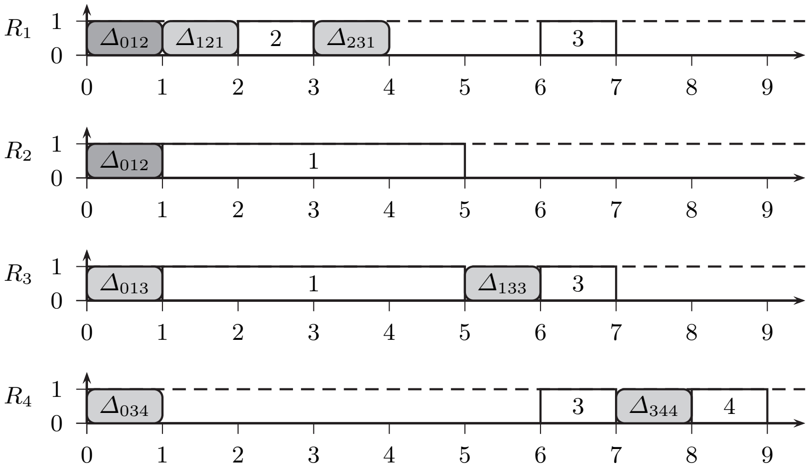


Figure 6.66: The earliest start schedule corresponding to the feasible resource flow $\mathcal{F}'$ displayed in Figure 6.65.

Neighborhood	connected	opt-connected
$\tilde{\mathcal{N}}_1 = \tilde{\mathcal{N}}_{\text{reroute}} \cup \tilde{\mathcal{N}}_{\text{reverse}}$	open	open
$\tilde{\mathcal{N}}_2 = \tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}} \cup \tilde{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$	$\times$	$\times$
$\tilde{\mathcal{N}}_3 = \tilde{\mathcal{N}}^{\text{max}}_{\text{reroute}} \cup \tilde{\mathcal{N}}_{\text{reverse}}$	$\times$	open
$\tilde{\mathcal{N}}_4 = \tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}} \cup \tilde{\mathcal{N}}_{\text{reverse}}$	$\times$	open

Table 6.14: Results concerning the connectivity of the four neighborhoods $\tilde{\mathcal{N}}_1$, $\tilde{\mathcal{N}}_2$, $\tilde{\mathcal{N}}_3$ and $\tilde{\mathcal{N}}_4$ considered in Section 6.2.4.

Finally, the two neighborhoods $\tilde{\mathcal{N}}_3 = \tilde{\mathcal{N}}^{\text{max}}_{\text{reroute}} \cup \tilde{\mathcal{N}}_{\text{reverse}}$ as well as $\tilde{\mathcal{N}}_4 = \tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}} \cup \tilde{\mathcal{N}}_{\text{reverse}}$ can be considered. While both neighborhoods are not connected (this again follows from Example 6.10), it remains an open question whether these two neighborhoods are opt-connected for the RCPSP with first- and second-tier resource transfers. The results concerning the connectivity of the four neighborhoods $\tilde{\mathcal{N}}_1$, $\tilde{\mathcal{N}}_2$, $\tilde{\mathcal{N}}_3$, and $\tilde{\mathcal{N}}_4$ considered in this section are summarized in Table 6.14.

6.3 A Tabu Search Algorithm

Finally, a tabu search algorithm for the RCPSP with first- and second-tier resource transfers is introduced in this section. An outline of this algorithm is given in Algorithm 6.3. In the following, this tabu search algorithm is described in more detail.

In this algorithm, solutions are represented by resource flows as they have been introduced in Section 6.1. Here, in line 1, an initial solution $s \in S$ (i.e. an initial resource flow) is calculated by a parallel schedule generation scheme based on a selected activity rule as well as the transfer rule ES. The schedule generation scheme used in this algorithm differs from the parallel schedule generation scheme outlined in Algorithm 4.2 (cf. Section 4.2.1) in the following aspects.

First of all, activities $i \in V_{\text{all}}$ have to be scheduled instead of jobs consisting of multiple operations. Then, it is important to note that no blocking resources exist in the model considered here. For this reason, the loop in line 7 of Algorithm 4.2 can be omitted as each activity $i \in V_{\text{all}}$ is completed p_i time units after it has been started and can be removed from the set

Algorithm 6.3: Tabu Search

```
Generate an initial solution $s \in S$;
$best := c(s)$; $s^* := s$; $TL := \emptyset$; $S^* := \emptyset$;
repeat
    Choose a solution $s'$ from the selected neighborhood that is not tabu;
    if no solution $s'$ could be generated then
        if $S^* \neq \emptyset$ then
            Restart with an elite solution $(s, TL) \in S^*$;
        else
            Generate a new solution $s \in S$;
            $TL := \emptyset$
        end
    else
        Update the tabu list $TL$;
        $s := s'$;
        if $c(s') < best$ then
            $best := c(s')$; $s^* := s'$; $TL := \emptyset$; $S^* := S^* \cup \{(s', TL)\}$;
            if $|S^*| > l_{max}$ then
                Remove the oldest elite solution from $S^*$;
            end
        end
        if an intensification condition is satisfied then
            if $S^* \neq \emptyset$ then
                Restart with an elite solution $(s, TL) \in S^*$;
            end
        else if a diversification condition is satisfied then
            Generate a new solution $s \in S$;
            $TL := \emptyset$
        end
    end
until a stopping condition is satisfied;
```

A_λ of active activities at time point $t_\lambda = S_i + p_i$ (i.e. lines 16 to 19 of Algorithm 4.2 can also be omitted). Finally, for the RCPSP with first- and second-tier resource transfers, pure second-tier resource transfers are possible. Here, the Procedure CalculateResourceTransfers has to be be modified such that resource transfers of resources $k \in \mathcal{R}^{sa}$ can start from either the start or the end of an activity $i \in V$. Now, an initial resource flow $\mathcal{F}$ can be computed by storing all first- and second-tier resource transfers selected by the modified Procedure CalculateResourceTransfers. It should be noted that, alternatively, the serial schedule generation scheme can be modified accordingly and also be used to generate an initial resource flow.

After an initial solution has been generated, new resource flows are generated based on modifications in the selected neighborhood in line 4 of the algorithm. Here, the various neighborhoods introduced in Section 6.2.4 can be used, e.g. the neighborhood $\tilde{\mathcal{N}}_4 = \tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}} \cup \tilde{\mathcal{N}}_{\text{reverse}}$. In any of these neighborhoods, whenever resource transfers have to be adapted based on a priority rule, sets $\tilde{U}_k$ (or U_k) consisting of incoming resource transfers are selected based on the transfer rule ES (i.e. based on the earliest arrival time of the resource units at the receiving activity) while sets $\tilde{V}_k$ (or V_k) of outgoing resource transfers are selected based on the transfer rule TT. The latter transfer rule TT selects resource transfers in the order of non-decreasing transfer times between the activities.

In order to evaluate a reroute move of two arcs between activities $i \in V_0$ and $j \in V_*$ as well as between $u \in V_0$ and $v \in V_*$, it is possible to calculate the maximal possible improvement of the makespan by regarding the latest finish times LF_j and LF_v of activities j and v in the current schedule as well as the possible earliest finish times EF_j and EF_v of these activities after the modification if only the redirected arcs are considered. Then, if $LF_j - EF_j \leq \delta$ or $LF_v - EF_v \leq \delta$ hold, the selected reroute move can not improve the makespan of the current resource flow by more than δ time units and the actual makespan does not have to be calculated. Here, we select δ as the improvement of the makespan for the best move evaluated so far. This preliminary evaluation can be performed in $\mathcal{O}(1)$ time based on the longest path lengths between the activities for the current resource flow. In the case of reverse moves, these are always evaluated by actually performing the move and evaluating the resulting resource flow.

Now, only moves based on modifications in the selected neighborhood that are not tabu are evaluated in the tabu search algorithm, i.e. no aspiration criteria are used due to the size of the neighborhoods. Additionally, if a resource flow is generated with a smaller objective function value than the objective function value of the current solution s, this solution s' is accepted immediately (first-fit). Otherwise, the best solution s' is selected for the next iteration, if such a solution exists.

After a solution s' has been chosen during an iteration, the tabu list TL is updated based on the selected move. Here, two tabu lists TL_{add} and

TL_{drop} are used as described by Glover and Laguna (1998). While the tabu list TL_{drop} stores information about arcs that have been added to the resource flow (or on which additional resource units are transferred), the tabu list TL_{add} stores information about arcs that have been removed from the resource flow (or on which fewer resource units are transferred). In particular, if a reroute modification has been applied to reroute resource units of resource $k \in \mathcal{R}$ on two selected arcs between activities $i \in V_0$ and $j \in V_*$ as well as between activities $u \in V_0$ and $v \in V_*$, the tabu list TL_{drop} stores the triples (i, v, k) and (u, j, k) (i.e additional units of resource k are transferred from activity i to activity v as well as from activity u to activity j) while the tabu list TL_{add} stores the triples (i, j, k) and (u, v, k) (i.e. fewer units of resource k are transferred from activity i to activity j as well as from activity u to activity v). Similarly, if a reverse modification has been applied to reverse all arcs between two activities $i \in V$ and $j \in V$, the tabu list TL_{drop} stores all triples (j, i, k) corresponding to resources $k \in \mathcal{R}$ for which at least one arc has been reversed while the tabu list TL_{add} stores all triples (i, j, k).

Now, a reroute move for two selected arcs of a resource $k' \in \mathcal{R}$ between activities $i' \in V_0$ and $j' \in V_*$ as well as between activities $u' \in V_0$ and $v' \in V_*$ is tabu if either the triple (i', j', k') or the triple (u', v', k') is contained in the tabu list TL_{drop} (i.e. the corresponding arcs may not be removed from the resource flow if they have only recently been added) or if either the triple (i', v', k') or the triple (u', j', k') is contained in the tabu list TL_{add} (i.e. the corresponding arcs may not be added to the resource flow if they have only recently been removed). Similarly, a reverse move between two selected activities $i' \in V$ and $j' \in V$ is tabu if at least one triple (i', j', k') is contained in the tabu list TL_{drop} or if at least one triple (j', i', k') is contained in the tabu list TL_{add} for all resources $k' \in \mathcal{R}$ for which at least one arc exists in the resource flow.

It should be noted that each of these two tabu lists TL_{add} and TL_{drop} is able to avoid cycling (i.e. the revisiting of a solution that has previously been visited) by itself. In particular, the tabu list TL_{drop} prevents the tabu search from removing specific arcs from a resource flow that have only recently been inserted while the tabu list TL_{add} prevents the tabu search from inserting specific arcs to a resource flow that have only recently been removed. In the tabu search algorithm, it is possible to use either only one of the two tabu lists or both depending on the selected tabu tenures. If both tabu lists are used, this generally results in a stronger tabu restriction because the tabu condition is satisfied more frequently for similar tabu tenures.

The tabu tenures t_{add} and t_{drop} (i.e. the number of iterations for which moves are considered tabu) are chosen independently for both tabu lists TL_{add} and TL_{drop} based on the size of the selected neighborhood $\mathcal{N}$:

$$t_{add} := rand(a) + \alpha \cdot |\mathcal{N}| \qquad \text{and} \qquad t_{drop} := rand(b) + \beta \cdot |\mathcal{N}|$$

Here, a and b are two positive integer values such that the function $rand$ randomly selects an integer number from the interval $[0, a[$ and $[0, b[$, respectively. For this, a random number generator with a fixed seed is used. Furthermore, α and β are selected with $\alpha, \beta \in [0, 1[$.

If no new solution could be generated during an iteration (e.g. if all moves are tabu) or if either an intensification or a diversification condition is satisfied, the tabu search either restarts from an elite solution if such a solution exists (cf. Section 4.2) or a new solution is generated by the parallel schedule generation scheme. Here, in order to diversify the search, we calculate the amount of iterations that activities $i \in V$ have been critical activities and use this information in order to select these activities with a higher priority (i.e. the priority values based on the selected activity rule are modified accordingly).

7 A Solution Approach for the HEP based on Resource Flows

In Chapter 4, a solution approach based on priority rules for the problem of hospital evacuations has been described. A drawback of this solution approach is that resource transfers of aids are selected by priority rules that do not necessarily ensure that a job is started as early as possible, i.e. no optimal solution might exist in the solution space generated by the schedule generation schemes. For this reason, an alternative solution approach for the HEP is described in this chapter. This solution approach is based on the resource flow representation introduced in Chapter 6. In contrast to the RCPSP with first- and second-tier resource transfers for which this solution representation has been introduced, however, the following differences have to be taken into account.

First of all, the problem of hospital evacuations is modeled as a multi-mode RCPSP such that the selected modes influence the required assistants and aids as well as the operations that actually have to be scheduled. For this reason, changing either of the two modes may make it necessary to repair the current resource flow, e.g. if different amounts of assistants or aids are required to process a job $j \in J$ after its equipment mode $m_1 \in \mathcal{M}_{j1}$ has been changed, or different operations of a job have to be scheduled after its route mode $m_2 \in \mathcal{M}_{j2}$ has been changed.

Next, while both first-tier as well as second-tier resource transfers can occur in the problem of hospital evacuations (i.e. assistants can be transferred as first-tier or second-tier resources), it is assumed that all assistants that are required to transport aids to the initial location of a patient are also used to evacuate the patient. As a result of this, no pure second-tier resource transfers can occur. Finally, blockings between operations of different jobs that require the same building section have to be incorporated into the resource flow representation.

In the following, the problem of hospital evacuations is decomposed into two subproblems. These subproblems are referred to as evacuation subproblem as well as routing subproblem. An outline of this solution approach is visualized in Figure 7.1.

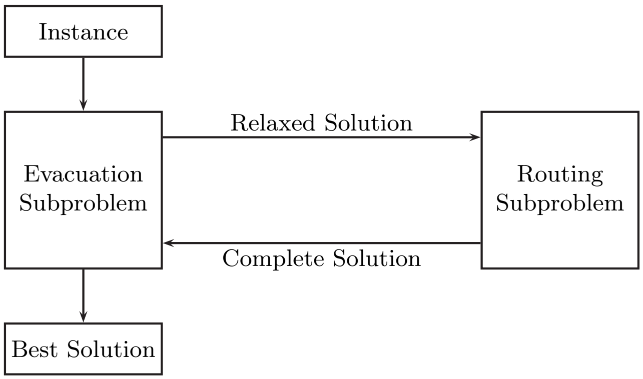


Figure 7.1: Outline of the solution approach described in Chapter 7.

As can be seen in this figure, the evacuation subproblem is solved first. For this, the problem of hospital evacuations is relaxed by assuming that all building sections have unlimited capacity (i.e. it is assumed that no blockings can occur). As a result of this, the usage of the building sections can be neglected for the evacuation subproblem. For this subproblem, every job $j \in J$ is assigned a route mode $m_2 \in \mathcal{M}_{j2}$ that can not be changed by the solution approach for this subproblem itself but relies on feedback from the solution approach for the routing subproblem. Then, the relaxed problem consists of selecting and modifying the equipment modes $m_1 \in \mathcal{M}_{j1}$ of the jobs $j \in J$ (i.e. regarding the required types of assistant and aids for the evacuation of the patient) as well as deciding in which order the patients are evacuated to the safety zones by the the available assistants and aids. This problem is solved by a tabu search algorithm in which solutions are represented by resource flows between jobs (cf. Section 7.1).

Now, whenever a given stopping condition $SC1$ is satisfied, the best resource flow for the evacuation subproblem that has been found since the last time the stopping condition $SC1$ has been satisfied is handed to the routing subproblem. For the routing subproblem, we no longer consider the relaxation of unlimited capacity in the building sections. Instead, the routing subproblem consists of modifying the route modes $m_2 \in \mathcal{M}_{j2}$ of the jobs $j \in J$ as well as deciding in which order the patients are evacuated through the corresponding building sections. For this, the resource flow for the evacuation subproblem that has been handed to the routing subproblem is converted into additional precedence constraints between the operations, i.e. the usage of assistants and aids is assumed to be fixed. Then, the routing subproblem is solved by a tabu search algorithm in which solutions are represented by resource flows between operations (cf. Section 7.2).

The resource flows generated during this second stage correspond to complete solutions that satisfy all constraints of the hospital evacuation problem. After a given stopping condition $SC2$ has been satisfied, the best resource flow generated during this stage is returned to the first stage. For the evacuation subproblem, we then replace the route modes $m_2 \in \mathcal{M}_{j2}$ of the jobs by the route modes of this solution for the routing subproblem in order to influence the further search. In particular, changing the evacuation routes for the patients such that these are evacuated to different safety zones might influence the transfer times between the jobs. Finally, the best complete solution is returned as output at the end of the algorithm after another stopping condition $SC3$ has been satisfied.

The solution approach outlined here as well as the decomposition of the problem into an evacuation as well as a routing subproblem has been chosen in order to reduce the size of the solution space. In particular, the solution space for the HEP can be very large due to the large number of operations that might have to be considered (i.e. depending on the number of building sections required to model the evacuation routes for the patients). By using the solution approach outlined above, the evacuation subproblem is solved first such that only the jobs have to be scheduled. As the number of jobs is generally smaller than the number of operations, the solution space for the evacuation subproblem is reduced compared to a one-stage algorithm. Additionally, the resource flow for the evacuation subproblem is translated into additional precedence constraints for the routing subproblem such that the solution space for the routing subproblem can be further constrained. Thus, in each subproblem, it is possible to consider a reduced solution space.

7.1 Evacuation Subproblem

In this section, we present a tabu search algorithm for the evacuation problem that represents solutions as resource flows. As described above, the evacuation subproblem deals with a relaxed version of the hospital evacuation problem that is concerned with scheduling all jobs with respect to the available assistants and aids while neglecting the usage of the building sections. Instead, it is assumed that all patients can be evacuated from their initial locations to the safety zones without interruptions (i.e. without blockings). Here, because jobs instead of operations have to be scheduled, we use an adapted version of the model introduced in Section 4.1. In this adapted model, each job $j \in J$ is assigned processing times p_{jm} as well as resource requirements r_{jmk} for all resources $k \in \mathcal{R}^{\text{trf}}$ based on the possible mode combinations $m = (m_1, m_2) \in \mathcal{M}_j$ of job j (with $m_1 \in \mathcal{M}_{j1}$ and $m_2 \in \mathcal{M}_{j2}$).

For this, the processing times p_{jm} of jobs $j \in J$ are calculated as the minimum time required to evacuate the patient corresponding to job j from his or her initial location to a safety zone based on the evacuation route denoted by route mode $m_2 \in \mathcal{M}_{j2}$ as well as the required assistants and aids denoted by equipment mode $m_1 \in \mathcal{M}_{j1}$. On the other hand, the resource requirements r_{jmk} for resources $k \in \mathcal{R}^{\text{trf}}$ of $j \in J$ correspond to the required assistants and aids for equipment mode $m_1 \in \mathcal{M}_{j1}$. Here, as stated above, the usage of building sections $k \in \mathcal{R}^{\text{sect}}$ is neglected for this subproblem. Finally, transfer time $\Delta_{\varphi(i)\psi(j)k}$ denotes the time required to transfer resource $k \in \mathcal{R}^{\text{trf}}$ from the end of job $i \in J$ (where $\varphi(i)$ denotes the safety zone to which the patient corresponding to job i is evacuated to in route mode $m_2 \in \mathcal{M}_{i2}$) to the start of job $j \in J$ (where $\psi(j)$ denotes the initial location of the patient corresponding to job j).

In the following, a route mode $m_2 \in \mathcal{M}_{j2}$ is selected for each job $j \in J$ based on some priority rule. Together with a solution for the evacuation subproblem (i.e. an assignment of an equipment mode $m_1 \in \mathcal{M}_{j1}$ to each job $j \in J$ as well as a feasible resource flow $\mathcal{F}$), it is then possible to evaluate a given solution for this subproblem. For this, an earliest start schedule is calculated based on the modified processing times p_{jm} and transfer times $\Delta_{\varphi(i)\psi(j)k}$ as they have been discussed above. As before, an earliest start schedule is calculated based on the lengths of the longest paths in the AON-flow network corresponding to a given solution (i.e. the mode assignments $m = (m_1, m_2)$ of the jobs $j \in J$ as well as the resource flow $\mathcal{F}$).

Example 7.1 We consider the problem from Example 4.4 where $N = 3$ patients have to be evacuated from the hospital displayed in Figure 4.1 with the help of two assistants (referred to as resource 1) as well as one stretcher (referred to as resource 2). Here, each job can be performed in only one mode combination $m = (m_1, m_2) = (1,1) = 1$ such that patient 1 has to be evacuated from room 5 to exit 1 by one assistant, patient 2 has to be evacuated from room 3 to exit 2 by two assistants as well as one stretcher, and patient 3 has to be evacuated from room 3 to exit 1 by one assistant. The processing times of the three jobs can then be calculated as $p_{11} = 6$, $p_{21} = 5$, and $p_{31} = 5$ based on the data from Example 4.4. The transfer times between the different locations are given in Table 4.1.

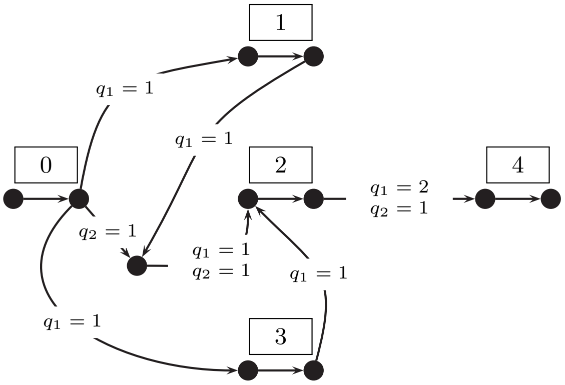


Figure 7.2: Solution for the evacuation subproblem displaying the resource transfers of assistants (resource 1) and aids (resource 2) between the jobs of the problem instance considered in Example 7.1.

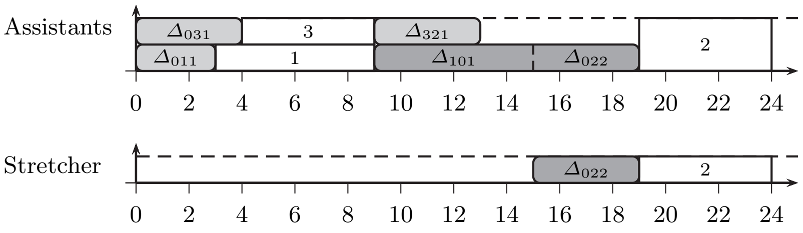


Figure 7.3: The earliest start schedule corresponding to the feasible resource flow displayed in Figure 7.2.

A feasible resource flow for the evacuation subproblem of this instance is displayed in Figure 7.2. The earliest start schedule with the makespan $C_{max} = 24$ corresponding to this resource flow is shown in Figure 7.3. $\square$

Now, the tabu search algorithm for the evacuation subproblem is described. An outline of this algorithm is given in Algorithm 7.1. In line 1 of this algorithm, an initial solution s_{relax} for the relaxed problem is generated by assigning an equipment mode $m_1 \in \mathcal{M}_{j1}$ as well as a route mode $m_2 \in \mathcal{M}_{j2}$ to each job $j \in J$ as described in Section 4.2. Based on these mode assignments, an initial resource flow is then generated by a parallel schedule generation scheme. Here, we have adapted the parallel schedule generation from Section 4.2.1 to schedule the jobs (instead of the operations) with respect to the available assistants and aids. It should be noted that this parallel schedule generation scheme is always able to generate a feasible resource flow because no blockings or deadlocks occur in the relaxed problem. After an initial solution s_{relax} has been generated, a complete solution s^*_{compl} based on this solution is calculated in line 2 of the algorithm. This is done by solving the corresponding routing subproblem (cf. Section 7.2).

Then, in the while-loop between lines 4 to 28, the tabu search algorithm chooses a solution s'_{relax} based on the current solution s_{relax} in line 5. For this, reroute and reverse moves in the neighborhoods described in Section 6.2 can be used. Here, in particular, we use the neighborhoods $\tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}}$ and $\tilde{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$. Additionally, we have implemented a neighborhood $\mathcal{N}^{\text{eqpt}}_{\text{mode}}$ that defines modifications to change the equipment mode $m_1 \in \mathcal{M}_{i1}$ of one job $i \in J$. As pointed out above, changing the equipment mode m_1 of a job $i \in J$ generally invalidates the current resource flow because some resource $k \in \mathcal{R}^{\text{trf}}$ may no longer be required by job i while other resources $k \in \mathcal{R}^{\text{trf}}$ may become required. In this case, the resource flow is repaired by redirecting resource transfers of resources $k \in \mathcal{R}^{\text{trf}}$ that are affected by the modification. Thus, if resource units of a resource $k \in \mathcal{R}^{\text{trf}}$ are no longer required by job i, they can immediately be transferred from jobs $h \in J$ that have sent these resource units to job i to jobs $j \in J$ that have received these resource units from job i. Similarly, if additional resource units of a resource $k \in \mathcal{R}^{\text{trf}}$ are required after the modification, resource transfers of resource k between two jobs $h \in J$ and $j \in J$ are redirected such that resource k is first transferred from job h to job i and then from job i to job j such that no cycles occur in the resulting resource flow.

After a new solution s'_{relax} has been generated, the corresponding move is stored in the tabu list in line 14 of the algorithm and the previous solution s_{relax} is replaced by solution s'_{relax} in line 15. Here, three tabu lists TL_{add},

TL_{drop}, and TL_{mode} are used as described in Sections 4.2 and 6.3. Additionally, if this solution s'_{relax} is better than the currently best solution s^*_{relax} (i.e. if $c(s'_{\text{relax}}) < c(s^*_{\text{relax}})$ holds), the currently best solution is replaced by this solution in line 17 of the algorithm.

Algorithm 7.1: Tabu Search for the Evacuation Subproblem

```
1  Generate an initial solution $s_{\text{relax}} \in S$;
2  Calculate $s^*_{\text{compl}} := RSP(s_{\text{relax}})$; $c^*_{\text{compl}} := c(s^*_{\text{compl}})$;
3  $c^*_{\text{relax}} := c(s_{\text{relax}})$; $s^*_{\text{relax}} := s_{\text{relax}}$; $TL := \emptyset$;
4  repeat
5      Choose a solution $s'_{\text{relax}}$ from the neighborhood of the current solution
       $s_{\text{relax}}$ that is not tabu;
6      if no solution $s'_{relax}$ could be generated then
7          Calculate $s_{\text{compl}} := RSP(s^*_{\text{relax}})$ with $c_{\text{compl}} := c(s_{\text{compl}})$;
8          if $c_{compl} < c^*_{compl}$ then
9              $c^*_{\text{compl}} := c_{\text{compl}}$; $s^*_{\text{compl}} := s_{\text{compl}}$;
10         end
11         Change the route modes $m_2 \in \mathcal{M}_{j2}$ of jobs $j \in J$ in solution $s_{\text{relax}}$
           to the route modes $\tilde{m}_2 \in \mathcal{M}_{j2}$ of the jobs in solution $s_{\text{compl}}$;
12         $c^*_{\text{relax}} := c(s_{\text{relax}})$; $s^*_{\text{relax}} := s_{\text{relax}}$; $TL := \emptyset$;
13     else
14         Update the tabu list $TL$;
15         $s_{\text{relax}} := s'_{\text{relax}}$;
16         if $c(s'_{relax}) < c^*_{relax}$ then
17             $c^*_{\text{relax}} := c(s'_{\text{relax}})$; $s^*_{\text{relax}} := s'_{\text{relax}}$; $TL := \emptyset$;
18         end
19     end
20     if stopping condition $SC1$ is satisfied then
21         Calculate $s_{\text{compl}} := RSP(s^*_{\text{relax}})$ with $c_{\text{compl}} := c(s_{\text{compl}})$;
22         if $c_{compl} < c^*_{compl}$ then
23             $c^*_{\text{compl}} := c_{\text{compl}}$; $s^*_{\text{compl}} := s_{\text{compl}}$;
24         end
25         Change the route modes $m_2 \in \mathcal{M}_{j2}$ of jobs $j \in J$ in solution $s_{\text{relax}}$
           to the route modes $\tilde{m}_2 \in \mathcal{M}_{j2}$ of the jobs in solution $s_{\text{compl}}$;
26         $c^*_{\text{relax}} := c(s_{\text{relax}})$; $s^*_{\text{relax}} := s_{\text{relax}}$; $TL := \emptyset$;
27     end
28 until stopping condition $SC3$ is satisfied;
```

If either no new solution s'_{relax} could be generated during an iteration or if stopping condition $SC1$ has been satisfied, the routing subproblem (RSP) is solved for the best solution s^*_{relax} found since the last restart of the algorithm (cf. lines 7 and 21). Then, if $c(s_{\text{compl}}) < c(s^*_{\text{compl}})$ holds for the objective function value of the resulting complete solution s_{compl}, this solution s_{compl} replaces the currently best complete solution s^*_{compl} found so far (cf. lines 9 and 23). Additionally, the route modes $m_2 \in \mathcal{M}_{j2}$ of jobs $j \in J$ that have been used since the last restart of the algorithm are changed to the route modes $\tilde{m}_2 \in \mathcal{M}_{j2}$ of the jobs in solution s_{compl} (cf. lines 11 and 25). Finally, the solution s^*_{relax} is replaced by this new solution and the tabu lists TL_{add}, TL_{drop}, and TL_{mode} are reset (cf. lines 12 and 26).

Above, changing the fixed route modes $m_2 \in \mathcal{M}_{j2}$ of jobs $j \in J$ based on a complete solution s_{compl} for the routing subproblem can be regarded as feedback to influence the further search. In particular, the transfer times between the jobs depend on the safety zones to which the corresponding patients are evacuated to. Thus, changing the route modes $m_2 \in \mathcal{M}_{j2}$ of jobs $j \in J$ (and hence the safety zones to which the corresponding patients are evacuated to) also influences these transfer times and the tabu search algorithm might be able to visit other regions of the solution space. This is visualized in Example 7.2.

Example 7.2 We consider a problem consisting of $N = 2$ jobs $i = 1{,}2$ that both require one assistant (referred to as resource 1) for the evacuation. It is assumed that patient 1 is initially located in a room $R1$ and patient 2 is initially located in a room $R2$. Additionally, both patients can be evacuated to either a safety zone $E1$ or a safety zone $E2$. Thus, both jobs can be performed in two possible mode combinations $m = (m_1, m_2) = (1, 1) = 1$ and $m = (m_1, m_2) = (1, 2) = 2$. Here, the processing times p_{im} of the jobs are given by $p_{11} = 1$, $p_{12} = 2$, $p_{21} = 3$, and $p_{22} = 1$. The transfer times between the different locations are given in Table 7.1. Finally, one assistant is available for the evacuation. This assistant is assumed to be initially located at safety zone $E1$.

	$R1$	$R2$	$E1$	$E2$
$R1$	0	3	1	2
$R2$	3	0	3	1
$E1$	1	3	0	3
$E2$	2	1	3	0

Table 7.1: The transfer times between the locations from Example 7.2.

Now, we assume that the tabu search algorithm for the evacuation subproblem initially selects mode combination 1 for job 1 (i.e. this patient is evacuated to safety zone $E1$) as well as mode combination 2 for job 2 (i.e. this patient is evacuated to safety zone $E2$). A feasible resource flow for the evacuation problem is displayed in Figure 7.4 while the earliest start schedule with the makespan $C_{max} = 6$ corresponding to this solution (i.e. the resource flow as well as the mode assignments) is shown in Figure 7.5.

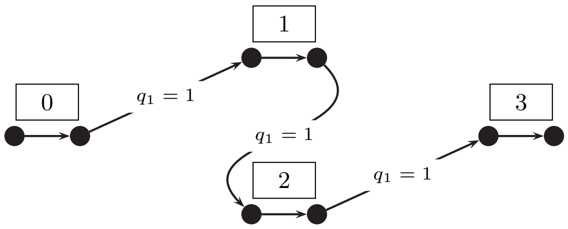


Figure 7.4: Solution for the evacuation subproblem displaying the resource transfers of the available assistant (resource 1) between the jobs of the problem instance considered in Example 7.2.

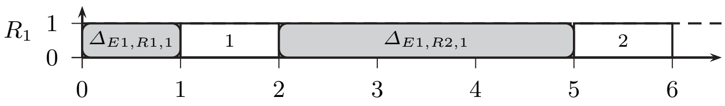


Figure 7.5: The earliest start schedule corresponding to the feasible resource flow displayed in Figure 7.4 if patient 1 is evacuated to safety zone $E1$ before patient 2 is evacuated to safety zone $E2$.

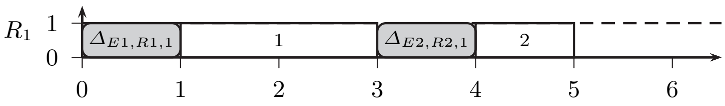


Figure 7.6: The earliest start schedule corresponding to the feasible resource flow displayed in Figure 7.4 if patient 1 is evacuated to safety zone $E2$ before patient 2 is evacuated to safety zone $E2$.

Alternatively, if patient 1 is evacuated to safety zone $E1$ instead (i.e. if mode combination 2 is selected for job 1), the earliest start schedule with the makespan $C_{max} = 5$ displayed in Figure 7.6 can be calculated based on the resource flow displayed in Figure 7.4. It should be noted that this schedule corresponds to the unique optimal solution for this example. Thus, changing the route modes of the jobs might influence the solutions that can be found by the tabu search algorithm for the evacuation subproblem. □

Finally, the tabu search algorithm terminates after stopping condition $SC3$ is satisfied. It should be noted that the constraints regarding the transfer of resources $k \in \mathcal{R}^{\text{trf}}$ for the evacuation subproblem differ slightly from those for the RCPSP with first- and second-tier resource transfers described in Section 5.2. In particular, no pure second-tier resource transfers can occur in the problem of hospital evacuations, i.e. the amount of assistants required to evacuate a patient is always sufficient to support the transfer of the required aids to the initial location of the patient. As a result of this, no start-to-start transfers have to be taken into account but only finish-to-start transfers of resources $k \in \mathcal{R}^{\text{trf}}$.

7.2 Routing Subproblem

In this section, the tabu search algorithm for the routing subproblem is introduced. As described above, this tabu search algorithm is handed the best solution s^*_{relax} for the relaxed problem that has been obtained since the last time the routing subproblem has been solved. This solution s^*_{relax} is then transformed and extended into an initial solution s_{compl} for the complete problem. Here, similar to the evacuation subproblem, solutions for the routing subproblem are represented as resource flows. After an initial solution s_{compl} has been generated, the tabu search algorithm tries to improve this solution s_{compl} by modifying the selected route modes m_2 of the jobs as well as by changing the order in which the patients are evacuated through the building sections.

In the following, the solution representation for this problem is outlined in Section 7.2.1 while the neighborhoods that have been defined on this solution representation are described in Section 7.2.2. In both of these sections, we primarily focus on the differences of the solution representation as well as the neighborhoods that arise from the blocking constraint as compared to the solution representation and the neighborhoods introduced in Chapter 6. Finally, the tabu search algorithm itself is presented in Section 7.2.3.

7.2.1 Solution Representation

The solution representation for the routing subproblem is outlined in this section. As described above, solutions for this subproblem are represented as resource flows. Here, first of all, it is important to note that all resources $k \in \mathcal{R}^{\text{sect}}$ can be transferred from an operation $u \in V_0$ to another operation $v \in V_*$ as first-tier resources by finish-to-start transfers without an associated transfer time (i.e. $\Delta_{uvk} = 0$ holds for all $u \in V_0$, $v \in V_*$, and $k \in \mathcal{R}^{\text{sect}}$). In this, the routing subproblem corresponds to a multi-mode RCPSP without transfer times as opposed to the multi-mode RCPSP with first- and second-tier transfers that has to be solved in the evacuation subproblem. On the other hand, however, blockings have to be taken into account for the routing subproblem (cf. Section 4.1.1).

In the following, a solution for the routing subproblem is represented as a resource flow as described in 6.1.1. Thus, if a resource $k \in \mathcal{R}^{\text{sect}}$ is transferred from an operation $u \in V_0$ to another operation $w \in V_*$, this transfer is represented by a resource transfer $f_{uwk} > 0$ denoting the amount of resource units (or space) transferred from operation u to operation w. Then, the blocking constraint is incorporated into the AON-flow network representing a given resource flow. This is done in order to ensure that blockings are taken into account when calculating the earliest start schedule based on the lengths of the longest paths between the operations. In particular, if an amount of $f_{uwk} > 0$ units of resource $k \in \mathcal{R}^{\text{sect}}$ is transferred from a blocking operation $u \in V$ to another operation $w \in V_*$, it has to be ensured that operation w can only start at time $S_w \geq S_v$ after the successor $v \in V$ of operation u (with $(u, v) \in B$) has been started.

Now, in order to integrate the blocking constraint into the AON-flow network representing a given resource flow, we adapt the following idea used by Mascis and Pacciarelli (2000, 2002) in the alternative graph representation for the job-shop problem with blockings. If an operation i is blocking (i.e. if the machine required by this operation only becomes available again after the next operation j of the same job has been started), a directed arc is inserted from the start of operation j to the start of the next operation u that requires the machine instead of from operation i to operation u directly. As a result of this, the starting time S_u of operation u as it is calculated based on the lengths of the longest paths can not be earlier than the starting time S_j of operation j (i.e. $S_u \geq S_j \geq S_i + p_i$ has to hold for the starting time of operation u).

In the following, we use this idea in order to represent blockings in an AON-flow network corresponding to a given resource flow. At the same time, however, it should be noted that the adapted AON-flow network representing a given resource flow as it is used here differs from the alternative graph model introduced by Mascis and Pacciarelli (2002). In particular, an alternative graph for a given problem instance represents the solution space consisting of all semi-active schedules for the problem instance. On the other hand, an AON-flow network corresponding to a given resource flow represents exactly one earliest start schedule.

Now, we describe the adapted AON-flow network representation for the routing subproblem in more detail. Here, first of all, each operation $u \in V_{\text{all}}$ is represented by two nodes u^{in} and u^{out} that correspond to the start as well as the end of the operation. Then, if an amount of $f_{uwk} > 0$ units of resource $k \in \mathcal{R}^{\text{sect}}$ are transferred from a blocking operation $u \in V$ to another operation $w \in V_*$ in resource flow $\mathcal{F}$ (i.e. the space in building section k required by operation u only becomes available after the patient has arrived in the next building section), an arc $(v^{\text{in}}, w^{\text{in}})_k$ from the start of operation $v \in V$ to the start of operation w is inserted into the AON-flow network where operation v is the direct successor of the blocking operation u (i.e. $(u, v) \in B$ holds). This situation is visualized in Figure 7.7.

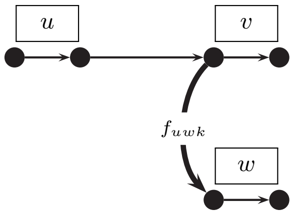


Figure 7.7: The resource transfer $f_{uwk} > 0$ of resource $k \in \mathcal{R}^{\text{sect}}$ from a blocking operation $u \in V$ to another operation $w \in V_*$ is represented by an arc $(v^{\text{in}}, w^{\text{in}})_k$ from the start of operation v to the start of operation w where operation v is the direct successor of operation u (i.e. $(u, v) \in B$ holds). Here, the resource transfer $f_{uwk} > 0$ is represented by a thick arc while the precedence constraint $u \rightarrow v$ is represented by a thin arc.

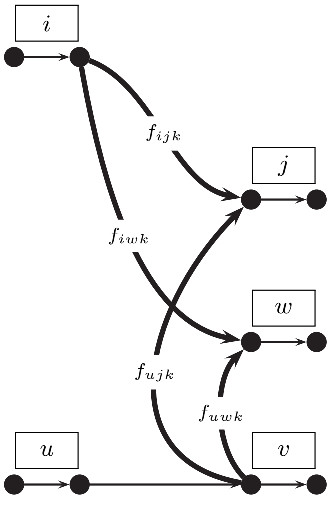
In this graph, the arc $(v^{\text{in}}, w^{\text{in}})_k$ from the start of operation v to the start of operation w is assigned the label f_{uwk} denoting the resource transfer of resource k from operation u to operation w it represents. In the following, we use this representation in order to clarify which resource transfers are modeled by the arcs in an AON-flow network. Now, the arc $(v, w)_k$ is assigned a weight equal to zero. Then, when calculating the starting time of operation w based on the lengths of the longest paths between the operations in the AON-flow network, operation w can only start in building section k after operation v has been started (i.e. $S_w \geq S_v$ has to hold) and the building section is no longer occupied by operation u.

Finally, if an amount of $f_{uwk} > 0$ units of space in a building section $k \in \mathcal{R}^{\text{sect}}$ are transferred from a non-blocking operation $u \in V_0$ (e.g. the last operation of the corresponding job $\sigma(u) \in J$) to another operation $w \in V_*$, the transfer is represented by an arc $(u^{\text{out}}, w^{\text{in}})_k$ with a weight equal to zero in the AON-flow network. In this case, resource k immediately becomes available after operation u has been completed and operation w can start at time $S_w \geq S_u + p_{um}$ based on the length of the longest paths.

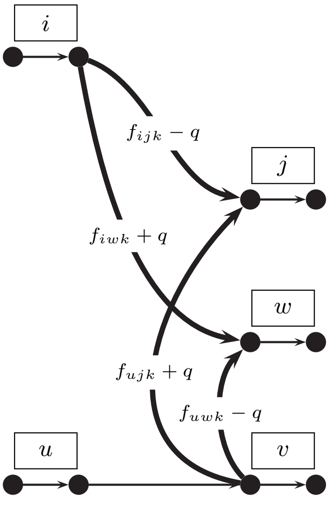
7.2.2 Neighborhoods

Next, the neighborhoods based on the solution representation introduced above that can be used for the routing subproblem are described. In particular, adapted neighborhoods $\hat{\mathcal{N}}_{\text{reroute}}$ and $\hat{\mathcal{N}}_{\text{reverse}}$ are outlined based on the neighborhoods $\mathcal{N}_{\text{reroute}}$ and $\mathcal{N}_{\text{reverse}}$. Here, while resource transfers between non-blocking operations can be redirected based on modifications in the neighborhoods $\hat{\mathcal{N}}_{\text{reroute}}$ and $\hat{\mathcal{N}}_{\text{reverse}}$ as described in Section 6.2, additionally, resource transfers between blocking operations (or pairs of one blocking and one non-blocking operation) have to be considered.

Now, we first consider reroute modifications in the neighborhood $\hat{\mathcal{N}}_{\text{reroute}}$. In this case, the two selected resource transfers can be rerouted as described in Section 6.2 regardless of whether they originate from blocking or non-blocking operations. For example, we assume that two resource transfers $f_{ijk} > 0$ and $f_{uwk} > 0$ for a resource $k \in \mathcal{R}^{\text{sect}}$ have been selected for a modification in the neighborhood $\hat{\mathcal{N}}_{\text{reroute}}$ such that operation $i \in V_0$ is non-blocking (i.e. the corresponding resource transfer is represented by an arc $(i^{\text{out}}, j^{\text{in}})_k$ from the end of operation i to the start of operation $j \in V_*$) and operation $u \in V$ is blocking (i.e. the corresponding resource transfer is represented by an arc $(v^{\text{in}}, w^{\text{in}})_k$ from the start of operation $v \in V$ to the start of operation $w \in V_*$ where operation v is the successor of operation u with $(u, v) \in B$). Furthermore, we assume that no directed path exists from either operation j to operation v or from operation w to operation i in the AON-flow network. In this case, an amount of $q = \{1, \ldots, \min\{f_{ijk}, f_{uwk}\}\}$ units of resource k is rerouted from operation u to operation j as well as from operation i to operation w in the resulting resource flow. In the corresponding graph, these two resource transfers $f'_{ujk} = f_{ujk} + q$ and $f'_{iwk} = f_{iwk} + q$ are represented by the arcs $(v^{\text{in}}, j^{\text{in}})_k$ and $(i^{\text{out}}, w^{\text{in}})_k$. This situation is visualized in Figure 7.8.



(a) Before modification.



(b) After modification.

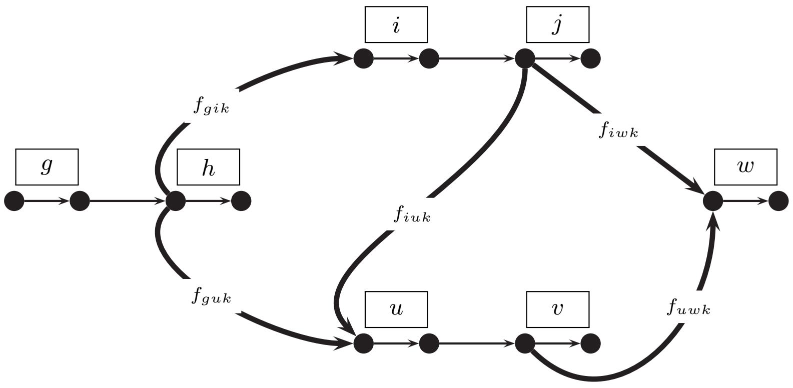
Figure 7.8: The resource transfers before a reroute modification has been used to redirect the resource transfers $f_{uwk} > 0$ and $f_{ijk} > 0$ are shown in (a) while the resource transfers after an amount of $q = \{1, \ldots, \min\{f_{ijk}, f_{uwk}\}\}$ units of resource $k \in \mathcal{R}^{\text{sect}}$ have been rerouted are displayed in (b).

Next, we consider the adapted neighborhood $\hat{\mathcal{N}}_{\text{reverse}}$. Here, for a reverse modification in this neighborhood, two operations $i \in V$ and $u \in V$ are selected such that operation i may not be a direct or indirect predecessor of operation u and no other directed path may exist from operation i to operation u. Then, in order to reverse all resource transfers between these operations, it is important to consider whether operation u is blocking or non-blocking. Depending on this, the arcs representing the resulting resource transfers from operation u either start from the operation itself (i.e. if operation u is non-blocking) or from the successor of operation u (i.e. if operation u is blocking). Finally, the incoming resource transfers to operation i as well as the outgoing resource transfers from operation u are redirected as before (cf. Section 6.2).

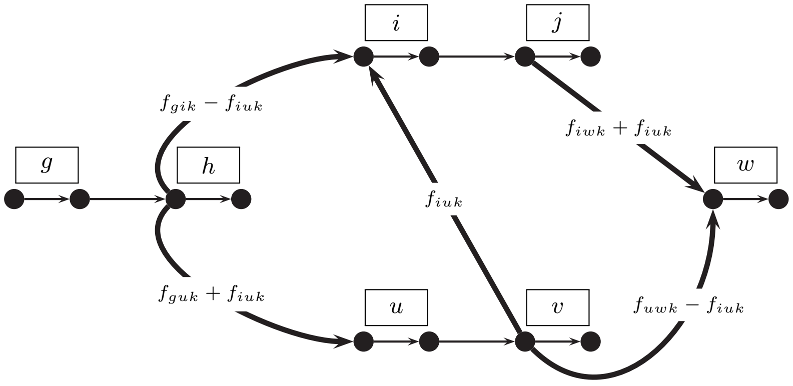
For example, we assume that a resource transfer $f_{iuk} > 0$ of a resource $k \in \mathcal{R}^{\text{sect}}$ from operation $i \in V$ to operation $u \in V$ exists such that operation i is no direct or indirect predecessor of operation u and no other directed path exists from operation i to operation u. Furthermore, we assume that both i and u are blocking operations. Then, this resource transfer is represented by the arc $(j^{\text{in}}, u^{\text{in}})_k$ in the AON-flow network where operation $j \in V$ is the successor of operation i (i.e. $(i, j) \in B$ holds). Additionally, we assume that operation i receives a sufficient amount of $f_{gik} \geq f_{iuk}$ units of resource k from the blocking operation $g \in V$ (i.e. the set $U_k = \{g\}$ can be calculated) and operation u sends a sufficient amount of $f_{uwk} \geq f_{iuk}$ units of resource k to operation $w \in V_*$ (i.e. the set $V_k = \{w\}$ can be calculated). These two resource transfers are represented by arcs $(h^{\text{in}}, i^{\text{in}})_k$ and $(v^{\text{in}}, w^{\text{in}})_k$ where operation $h \in V$ is the successor of operation g (i.e. $(g, h) \in B$ holds) and operation $v \in V$ is the successor of operation u (i.e. $(u, v) \in B$ holds). It should be noted that the sets U_k and V_k are again computed based on priority rules as described in Section 6.2 and might contain more operations.

Now, if the resource transfer f_{iuk} is reversed based on a modification in the neighborhood $\hat{\mathcal{N}}_{\text{reverse}}$, an amount of $f'_{uik} = f_{iuk}$ units of resource k is transferred from operation u to operation i in the resulting resource flow $\mathcal{F}'$. Here, because operation u is a blocking operation, this resource transfer is represented by the arc $(v^{\text{in}}, i^{\text{in}})_k$ from the start of operation v to the start of operation i. Additionally, an amount of f_{iuk} units of resource k are redirected from operation g to operation u (represented by the arc $(h^{\text{in}}, u^{\text{in}})_k$) as well as from operation i to operation w (represented by the arc $(j^{\text{in}}, w^{\text{in}})_k$). This situation is visualized in Figure 7.9.

Similar to Section 6.2, it is again possible to introduce reduced neighborhoods $\hat{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}}$ and $\hat{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$. Here, for neighborhood $\hat{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}}$, at least one of the two selected arcs representing resource transfers $f_{ijk} > 0$ and $f_{uvk} > 0$ has to be a critical arc. Furthermore, always the maximal amount of $q = \min\{f_{ijk}, f_{uvk}\}$ units of resource $k \in \mathcal{R}^{\text{sect}}$ are redirected by a modification in the neighborhood $\hat{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}}$. On the other hand, the neighborhood $\hat{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$ is again limited to modifications between critical activities $i \in V$ and



(a) Before modification.



(b) After modification.

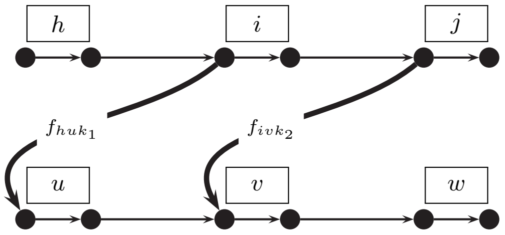
Figure 7.9: The resource transfers of a resource $k \in \mathcal{R}^{\text{sect}}$ before the resource transfer $f_{iuk} > 0$ between operations $i \in V$ and $u \in V$ is reversed are shown in (a) while the resource transfers after the modification are displayed in (b). Apart from reversing the corresponding arc, incoming as well as outgoing resource transfers between the operations have been redirected accordingly.

$j \in V$ such that at least one critical arc representing the resource transfer of a resource $k \in \mathcal{R}^{\text{sect}}$ exists between these two activities.

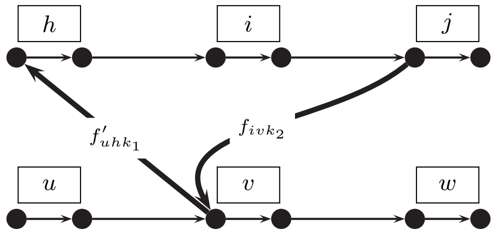
It should be noted that a problem occurs if resource transfers from two or more consecutive operations of one job exist to two or more consecutive operations of another job. In this case, reversing any of these arcs based on a modification in either the neighborhood $\hat{\mathcal{N}}_{\text{reverse}}$ or $\hat{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$ always results in a cycle. This situation is visualized in Figure 7.10 where resource transfers $f_{huk_1} > 0$ and $f_{ivk_2} > 0$ exist from the two consecutive blocking operations $h \in V$ and $i \in V$ of one job to the two consecutive blocking operations $u \in V$ and $v \in V$ of another job. In this case, we assume that the precedence constraints $(h, i) \in B$, $(i, j) \in B$, $(u, v) \in B$, and $(v, w) \in B$ are given such that the two resource transfers are represented by the arcs $(i^{\text{in}}, u^{\text{in}})_{k_1}$ and $(j^{\text{in}}, v^{\text{in}})_{k_2}$, respectively. Then, reversing the resource transfer f_{huk_1} results in a cycle between operations h, i, j, and v while reversing the resource transfer f_{ivk_2} results in a cycle between operations u, v, w, and i. Here, incoming resource transfers as well as outgoing resource transfers that are redirected as part of a reverse modification do not have to be considered because the order in which the corresponding operations are processed in relation to the selected operations does not change.

It is easy to see that this limitation might lead to situations in which a given resource flow $\mathcal{F}$ can not be transformed into an optimal resource flow $\mathcal{F}^*$ by a finite number of modifications in either the neighborhood $\hat{\mathcal{N}}_1 = \hat{\mathcal{N}}_{\text{reroute}} \cup \hat{\mathcal{N}}_{\text{reverse}}$ or the neighborhood $\hat{\mathcal{N}}_2 = \hat{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}} \cup \hat{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$. As a result of this, neither of these neighborhoods is opt-connected (and hence also not connected) for the problem considered here.

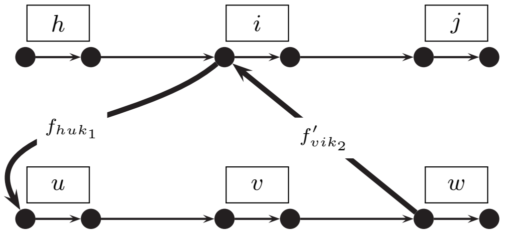
For this reason, we now introduce modified neighborhoods $\hat{\mathcal{N}}^{\text{cons}}_{\text{reverse}}$ and $\hat{\mathcal{N}}^{\text{cons,ca}}_{\text{reverse}}$ such that both of these neighborhoods reverse all resource transfers between consecutive operations of the same jobs. Here, for both of these neighborhoods, two maximal sets P and Q are computed such that the set P contains a total of λ consecutive operations $i_1 \to \ldots \to i_\lambda$ of one job while the set Q contains a total of λ consecutive operations $v_1 \to \ldots \to v_\lambda$ of another job. Furthermore, resource transfers $f_{i_1 v_1 k_1} > 0, \ldots, f_{i_\lambda v_\lambda k_\lambda} > 0$ for resources $k_1, \ldots, k_\lambda \in \mathcal{R}^{\text{sect}}$ have to exist between the operations contained in these sets. It should be noted that these sets contain the same number of operations due to the fact that each evacuation route is represented as a chain of operations such that each operation of the same job requires a different building section. These sets are called maximal if no pair of operations $h \in V$ and $u \in V$ (with $h \to i_1$ and $u \to v_1$ being direct predecessors of



(a) Before modification.



(b) After modification of resource transfer f_{huk_1}.



(c) After modification of resource transfer f_{ivk_2}.

Figure 7.10: If resource transfers from two or more consecutive operations of one job to two or more consecutive operations of another job exist (cf. (a)), reversing either of these resource transfers always results in a cycle (cf. (b) and (c)).

operations i_1 and v_1 of the corresponding jobs) or j and w (with $i_\lambda \to j$ and $v_\lambda \to w$ being direct successors of operations i_λ and v_λ of the corresponding jobs) with resource transfers $f_{huk} > 0$ or $f_{jwk} > 0$ exist for some resource $k \in \mathcal{R}^{\text{sect}}$. Then, a modification in the neighborhood $\hat{\mathcal{N}}^{\text{cons}}_{\text{reverse}}$ reverses all resource transfers between the operations contained in these sets. Finally, for a modification in the neighborhood $\hat{\mathcal{N}}^{\text{cons,ca}}_{\text{reverse}}$, at least one of the arcs representing the resource transfers between operations from the sets P and Q has to be a critical arc.

It should be noted that the size of these neighborhoods is again bounded by $\mathcal{O}(n^2)$ for the neighborhood $\hat{\mathcal{N}}^{\text{cons}}_{\text{reverse}}$ (i.e. each resource transfer between two activities can belong to at most one pair of sets P and Q) and $\mathcal{O}(n)$ for the neighborhood $\hat{\mathcal{N}}^{\text{cons,ca}}_{\text{reverse}}$ (i.e. for a given critical path, the sets P and Q have to be calculated for each pair of critical activities at most once). Finally, we did not further study the connectivity of the different combinations of neighborhoods that are possible. Based on the results from Section 6.2, however, all neighborhoods except for the neighborhood $\hat{\mathcal{N}}_3 = \hat{\mathcal{N}}_{\text{reroute}} \cup \hat{\mathcal{N}}_{\text{reverse}} \cup \hat{\mathcal{N}}^{\text{cons}}_{\text{reverse}}$ can be opt-connected at most.

7.2.3 A Tabu Search Algorithm

In the following, a tabu search algorithm for the routing subproblem is presented. As described above, this algorithm is called by the tabu search algorithm for the evacuation subproblem in order to generate a complete solution s_{compl} for a given problem instance based on the best solution s^*_{relax} for the relaxed problem that has been found since the last time the routing subproblem has been solved. For this, the given solution s^*_{relax} is first transformed and extended into an initial solution s_{compl} for the routing subproblem. This is done by replacing each job $j \in V$ by the operations $u \in V_j(m_2)$ representing the evacuation route corresponding to the route mode $m_2 \in \mathcal{M}_{j2}$ of this job that has been used in the evacuation subproblem.

Next, the resource transfers $f_{ijk} > 0$ of resources $k \in \mathcal{R}^{\text{trf}}$ between jobs $i \in J$ and $j \in J$ are replaced by additional precedence constraints between the operations. Here, if a resource transfer $f_{ijk} > 0$ of some resource $k \in \mathcal{R}^{\text{trf}}$ exists between jobs $i \in J$ and $j \in J$, a precedence constraint $u \rightarrow v$ between the last operation $u \in V'_{\text{out}}$ of the selected evacuation route of job i and the first operation $v \in V'_{\text{in}}$ of the selected evacuation route of job j is introduced. In the AON-flow network, the arcs representing these additional precedence constraints are weighted with a value corresponding to the longest transfer time of any resource $k \in \mathcal{R}^{\text{trf}}$ transferred between the two jobs as it is calculated based on inequalities (4.2) to (4.4). These weights are later used in order to calculate the starting times of the operations based on the lengths of the longest paths in the AON-flow network.

Finally, the initial resource transfers of resources $k \in \mathcal{R}^{\text{sect}}$ between the operations have to be calculated. For this, all jobs $j \in J$ are sorted with respect to the precedence constraints according to a given priority rule. Then, each job $j \in J$ is considered in this order and resource transfers from already completed operations to each operation $u \in V_j(m_2)$ of the evacuation route corresponding to the selected route mode $m_2 \in \mathcal{M}_{j2}$ are calculated such that no cycles occur in the resulting graph. Here, because all operations of a job are inserted before operations of another job are considered and because operations can only be inserted between already completed operations as well as the dummy end operation $n+1$, it is always possible to generate a feasible resource flow (i.e. no deadlocks occur).

Example 7.3 We again consider the problem from Example 4.4 as well as the resource flow for the evacuation subproblem displayed in Figure 7.2 (cf. Example 7.1). Based on this resource flow, we now generate an initial solution for the routing subproblem. For this, in a first step, the jobs 1, 2, and 3 are replaced by the corresponding chains of operations representing the respective evacuation routes. Additionally, the resource transfers of assistants and aids between the jobs are transformed into additional precedence constraints between the operations. For example, the resource transfer of resource 1 from job 3 to job 2 is replaced by a precedence constraint $16 \rightarrow 7$ between the last operation of job 3 and the first operation of job 2.

Next, the initial resource transfers of resources $k \in \mathcal{R}^{\text{sect}}$ between the operations have to be calculated. For this, the three jobs are sorted according to a selected priority rule with respect to the precedence constraints. For example, we assume that the order $1, 3, 2$ is calculated. In this case, resource transfers to all operations of job 1 are selected for the required building sections based on a priority rule. Afterward, resource transfers to all operations of job 3 and finally to all operations of job 2 are selected. As a result of this, all resource transfers to operations of job 1 originate at the dummy source operation 0, all resource transfers to operations of job 3 originate at either the dummy source operation 0 or an operation of job 1 that requires the same building section, and all resource transfers to operations of job 2 originate at either the dummy source operation 0 or an operation of either job 1 or 3 that requires the same building section.

A partial AON-flow network displaying both, the precedence constraints between the operations as well as the resource transfers of resources R3 (room 3), C4 (corridor section 4), and E1 (exit 1) is displayed in Figure 7.11. For the sake of clarity, the remaining resource transfers have been

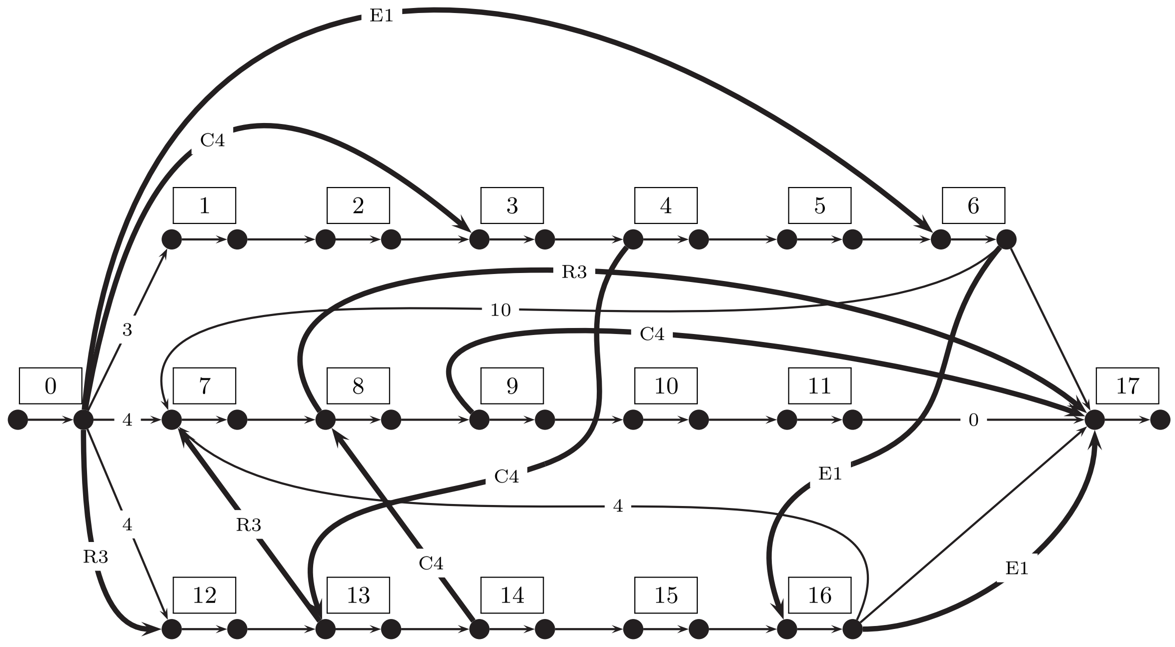


Figure 7.11: Partial AON-flow network for the problem instance considered in Example 7.3. The thin arcs between the operations represent precedence constraints while the thick arcs represent resource transfers of the building sections R3 (room 3), C4 (corridor section 4), as well as E1 (exit 1) between the operations.

omitted from this figure. In this AON-flow network, the precedence constraints $0 \rightarrow 1$, $0 \rightarrow 7$, $0 \rightarrow 12$, $6 \rightarrow 7$, $16 \rightarrow 7$, and $11 \rightarrow 17$ represent the additional precedence constraints that have been calculated based on the resource flow for the evacuation subproblem displayed in Figure 7.2. These arcs are weighted with the time required to transfer the respective resources $k \in \mathcal{R}^{\text{trf}}$ between the operations. The arcs visualizing the resource transfers of resources $k \in \mathcal{R}^{\text{sect}}$ are labeled with the name of the corresponding resource. These arcs are assigned a weight equal to zero in order to calculate the starting times of the operations based on the lengths of the longest paths between the operations.

Finally, it should be noted that this AON-flow network represents a complete solution for the problem instance considered in this example. The corresponding earliest start schedule with the makespan $C_{max} = 24$ is displayed in Figure 4.8. $\square$

After an initial solution s_{compl} has been generated, the tabu search algorithm tries to generate and evaluate a new solution s'_{compl} in each iteration until either no new solution s'_{compl} could be generated during an iteration or until a given stopping condition is satisfied. For this algorithm, we use the adapted neighborhoods $\hat{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}}$, $\hat{\mathcal{N}}^{\text{ca}}_{\text{reverse}}$, and $\hat{\mathcal{N}}^{\text{cons,ca}}_{\text{reverse}}$ in order to generate new solutions s'_{compl}. Additionally, a neighborhood $\mathcal{N}^{\text{route}}_{\text{mode}}$ is used such that a modification in this neighborhood changes the route mode $m_2 \in \mathcal{M}_{j2}$ of exactly one job $j \in J$. It should be noted that changing the route mode $m_2 \in \mathcal{M}_{j2}$ of a job $j \in J$ makes it necessary to replace the operations representing the previous evacuation route of the job by the operations representing the new evacuation route.

In order to insert operations $u \in V_j(m_2)$ corresponding to the new route mode $m_2 \in \mathcal{M}_{j2}$ of a job $j \in J$, it is first tested if the same building section $k \in \mathcal{R}^{\text{sect}}$ has been used by an operation of the same job before the modification. If this is the case, it is always possible to redirect all incoming as well as outgoing resource transfers of resource k from this operation to operation u. This is due to the fact that each building section $k \in \mathcal{R}^{\text{sect}}$ is required by at most one operation of each job (i.e. a patient is not moved through the same building section more than once during the evacuation) and the required space is the same both before as well as after the modification (i.e. the required assistants and aids do not change).

Otherwise, if building section k has not been required by an operation of the same job before the modification, resource transfers between other operations have to be redirected to operation u as described in Section 6.2.4. Additionally, resource transfers of building sections $k \in \mathcal{R}^{\text{sect}}$ from operations corresponding to the old evacuation route that are no longer required also have to be redirected, i.e. they can immediately be transferred from operations that send resource units of resource k to this operation to operations that receive resource units of resource k from this operation. It should be noted that the tabu search algorithm may not always be able to repair the graph after the route mode m_2 of a job has been changed due to deadlocks. In this case, the tabu search algorithm has to choose a different modification based on the current solution s_{compl}.

After a new solution s'_{compl} has been generated, the tabu lists TL_{add}, TL_{drop}, and TL_{mode} are updated based on the modification that has resulted in this new solution s'_{compl}. As before, these tabu lists are used as described in Sections 4.2 and 6.3. Additionally, if this solution s'_{compl} is better than the currently best solution s^*_{compl} (i.e. if $c(s'_{\text{compl}}) < c(s^*_{\text{compl}})$ holds), the currently best solution is replaced by this solution. Finally, the tabu search terminates after either no new solution s'_{compl} could be generated in an iteration or after a given stopping condition has been satisfied. In this case, the best solution s^*_{compl} is returned to the evacuation subproblem and the algorithm continues as described in Section 7.1.

8 Computational Results

In this chapter, we evaluate the performance of the different algorithms introduced in this thesis. In particular, we test the tabu search algorithm described in Section 6.3 that has been implemented to solve the RCPSP with first- and second-tier resource transfers as well as the two solution approaches for the problem of hospital evacuations outlined in Section 4.2 as well as in Chapter 7, respectively. All of these algorithms have been implemented in Java and the evaluation has been performed on a computer with an Intel Core 2 Quad Q6600 (2.4 GHz) processor and 4 GB RAM.

In the following, results for the tabu search algorithm that has been introduced in Section 6.3 are reported in Section 8.1. Here, we primarily report results for the classical RCPSP as well as the RCPSP with first-tier resource transfers because problem instances for both of these problems can be obtained and computational results for other heuristics have been published. On the other hand, for the RCPSP with first- and second-tier resource transfers, neither problem instances nor prior results exist in literature to the best of our knowledge. For this reason, we have generated our own test data and compare the results obtained by our algorithm with those obtained by IBM ILOG CPLEX 12.1 (by the IBM Corporation) based on the mixed-integer linear programming formulation introduced in Section 5.2.2. Finally, in Section 8.2, we report results for the solution approaches that have been implemented in order to solve the problem of hospital evacuations.

8.1 Results for the RCPSP with and without Transfer Times

In this section, we evaluate the performance of the tabu search algorithm introduced in Section 6.3. Below, computational results for the classical RCPSP are reported in Section 8.1.1, results for the RCPSP with first-tier resource transfers are reported in Section 8.1.2, and results for the RCPSP with first- and second-tier resource transfers are reported in Section 8.1.3.

8.1.1 Classical RCPSP

In order to evaluate the tabu search algorithm, we solve the four sets of problem instances for the classical RCPSP consisting of 30, 60, 90, and 120 activities, respectively, as they have been introduced by Kolisch and Sprecher (1997) and Kolisch et al. (1999). For each of these problem instances, a total of $r = 4$ renewable resources are available. Additionally, the network complexity NC, the resource factor RF, as well as the resource strength RS have been varied in order to generate these problem instances. These parameters are described in detail by Kolisch et al. (1995) and determine the complexity of the problem instances (i.e. they determine how hard individual instances are to solve). In total, each of the sets of problem instances consisting of 30, 60, and 90 activities contains 480 instances while the set of problem instances consisting of 120 activities contains a total of 600 instances. These problem instances are available online at the website of the project scheduling problem library (PSPLIB, `http://www.om-db.wi.tum.de/psplib/main.html`).

The tabu search algorithm used to solve these problem instances has been set up as follows: First of all, we use the neighborhood $\mathcal{N}_2 = \mathcal{N}^{\text{max,ca}}_{\text{reroute}} \cup \mathcal{N}^{\text{ca}}_{\text{reverse}}$ described in Section 6.2.2 in order to find new solutions. Then, the stopping condition for the tabu search algorithm is set to 10 000 iterations (i.e. the tabu search terminates after at most 10 000 iterations have been performed). It should be noted that this stopping condition differs from the stopping condition used by Hartmann and Kolisch (2000) and Kolisch and Hartmann (2006) in their extensive evaluation of heuristics for the RCPSP. Instead, they use a stopping condition based on the number of partial or complete schedules generated during the search.

We have opted to use a different stopping condition because individual modifications in our tabu search algorithm may often have only a limited effect on the resulting schedule. For example, if resource transfers for all $r = 4$ resources exist between two activities $i \in V$ and $j \in V$, at least four reroute modifications would be required in order to reroute all of these resource transfers such that the two activities can be processed in parallel. Additionally, in order to perform these four modifications, a total of four iterations are required because only one modification per iteration can be selected. At the same time, the number of modifications that are evaluated in each iteration might be very large due to the size of the neighborhoods (in particular due to the size of the neighborhood $\mathcal{N}^{\text{max,ca}}_{\text{reroute}}$). Then, because each of these modifications is counted as one schedule, the overall number of iterations that can be performed before a stopping condition based on the number of schedules generated during the search is satisfied can be very small. For example, already for some of the problem instances consisting of 30 operations, less than 100 iterations are performed for a stopping condition of 5 000 schedules depending on how the schedules are counted.

Next, $l_{max} = 1$ elite solutions can be stored at any time during the search. The algorithm restarts from this elite solution if no improving solution (i.e. a solution with a smaller makespan than the makespan of the currently best solution) could be found in the last 750 iterations. If no improving solution could be found in the last 1 500 iterations, the algorithms restarts from a new solution as described in Section 6.3. These parameters have been selected based on some preliminary tests and are used to solve all problem instances. Finally, the parameters a, b, α, and β used to calculate t_{add} and t_{drop} have been selected independently for each set of problem instances based on preliminary tests such that two passes are performed over all problem instances (cf. Table 8.1). Here, only the tabu list TL_{add} is used for the first pass while both tabu lists TL_{add} and TL_{drop} are used for the second pass.

n	a	α	b	β
30	5	0.3	0	0
60	10	0.1	0	0
90	10	0.4	0	0
120	15	0.1	0	0

(a) Parameter settings 1.

n	a	α	b	β
30	4	0.3	3	0.005
60	5	0.1	3	0.01
90	5	0.3	3	0.01
120	10	0.1	3	0.01

(b) Parameter settings 2.

Table 8.1: Parameter settings used to evaluate the problem instances. The parameters for the first pass if only the tabu list TL_{add} is used are displayed in (a) while the parameters for the second pass if both tabu lists TL_{add} and TL_{drop} are used are displayed in (b).

Based on these parameters, we now evaluate the problem instances. The obtained results are shown in Table 8.2. Similar to the computational study by Kolisch and Hartmann (2006), we report the average deviation Δ (in percent) from the optimal solutions for the problem instances consisting of 30 activities as well as the average deviation Δ (in percent) from the critical path lower bounds LB_0 for the problem instances consisting of 60, 90, and 120 activities. Furthermore, the average time t (in seconds) required by the

Iter.	0		100		500		1 000		5 0000		10 000	
n	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]
30	4.28	0.0	2.31	0.1	1.75	0.5	1.53	1.0	0.86	4.3	0.66	7.8
60	17.33	0.0	15.13	0.9	14.52	4.3	14.20	8.2	13.33	37.6	13.02	73.1
90	15.70	0.0	14.44	3.3	14.01	14.6	13.79	28.4	13.00	128.5	12.79	250.2
120	43.76	0.0	41.98	14.9	41.14	69.4	40.76	137.9	38.93	653.7	38.32	1297.9

(a) Results for the first pass (parameter settings 1).

Iter.	0		100		500		1 000		5 0000		10 000	
n	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]
30	4.28	0.0	2.41	0.1	1.85	0.5	1.65	1.0	0.91	4.4	0.74	8.2
60	17.33	0.0	15.32	0.9	14.68	4.1	14.41	8.0	13.51	37.2	13.19	73.2
90	15.70	0.0	14.51	3.1	14.13	14.1	13.92	27.5	13.07	128.7	12.83	253.0
120	43.76	0.0	42.01	14.3	41.18	66.2	40.82	131.6	39.06	633.8	38.51	1254.7

(b) Results for the second pass (parameter settings 2).

Table 8.2: Results obtained by the tabu search algorithm for the problem instances from the PSPLIB consisting of 30, 60, 90, and 120 activities. The results for the first pass based on the parameter settings from Table 8.1(a) are reported in (a) while the results for the second pass based on the parameter settings from Table 8.1(b) are reported in (b).

tabu search algorithm is displayed. These results are reported after 0, 100, 500, 1 000, 5 000, as well as 10 000 iterations.

In Figure 8.1, we display the average deviation from the optimal solutions after 0, 100, 500, 1 000, 5 000, as well as 10 000 iterations for the problem instances consisting of 30 activities during the first pass. It can be seen that the largest improvements have been achieved during the first 500 iterations. Also, between 1 000 and 5 000 iterations a large improvement can be observed. Apart from the greater number of iterations since the last measurement, this is also due to the intensification and diversification strategies used during the tabu search.

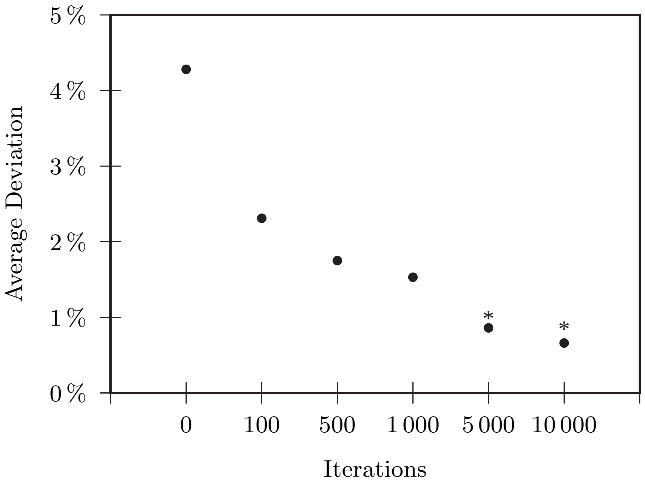


Figure 8.1: Average deviation from the optimal solutions after 0, 100, 500, 1 000, 5 000, and 10 000 iterations for the problem instances consisting of 30 activities obtained during the first pass based on the parameter settings from Table 8.1(a). The asterisks mark the average deviation for the same parameter settings if no intensification and diversification strategies are used.

For comparison, we have solved the problem instances consisting of 30 activities without intensification and diversification strategies using the same parameters as displayed in Table 8.1(a). In this case, the average deviation after 0, 100, 500, and 1 000 remains the same while it is $\varDelta = 1.03\,\%$ after

5 000 iterations and only $\varDelta = 0.89\,\%$ after 10 000 iterations. These results are also displayed in Figure 8.1. It should be noted that similar results can be observed for the other sets of problem instances (cf. Table 8.2).

Now, we compare our results to other available results. For this, we report our best results in Table 8.3 and compare them to the best results obtained by heuristics as they have been reported on the website of the PSPLIB. In particular, we calculate the average deviation $\varDelta_{best}$ (in percent) of the best solutions obtained in either the first or the second pass as described above and compare it to the average deviation $\varDelta_{heur}$ (in percent) of the best heuristic solutions reported on the website of the PSPLIB. As before, the average deviation is calculated from the optimal solutions for the problem instances consisting of 30 activities as well as from the critical path lower bounds LB_0 for the problem instances consisting of 60, 90, and 120 activities. Apart from these results, we also report the number *opt* of problem instances for which an optimal solution could be found. Here, a solution is referred to as being optimal if its makespan is either equal to the makespan of the optimal solution for problem instances consisting of 30 activities or if the makespan is equal to the best known lower bound (as reported on the website of the PSPLIB) for the remaining sets of problem instances.

n	$\varDelta_{best}$ [%]	*opt*	$\varDelta_{heur}$ [%]	*opt*
30	0.58	401	0.00	480
60	12.85	345	10.37	431
90	12.54	340	9.48	402
120	37.85	171	29.18	291

Table 8.3: Average deviation $\varDelta_{best}$ of the best solution obtained in either the first or the second pass. These values are compared to the average deviation $\varDelta_{heur}$ of the best solutions obtained by heuristic algorithms as reported on the website of the PSPLIB. The number *opt* denotes the number of problem instances for which the solution could be proved to be optimal.

It should be noted that here as well as in the following tables, we will generally not compare the computational times required by different algorithms because these algorithms have been used on different systems using different stopping conditions. It is likely, however, that our algorithm requires a similar or longer computational time in order to solve the problem instances. On the one hand, this is due to the size of the neighborhood $\mathcal{N}_{\text{reroute}}^{\text{max,ca}}$ which depends on the number of resource transfers between activities that can be adapted. On the other hand, evaluating a single modification in the neighborhood $\mathcal{N}^{\text{ca}}_{\text{reverse}}$ is bounded by $\mathcal{O}(n^3)$ because the lengths of the longest paths between the activities have to be calculated. In contrast to this, the time complexity for generating a schedule for a given activity list based on a schedule generation scheme is bounded by $\mathcal{O}(n^2 r)$ for the classical RCPSP (cf. Section 3.2.3). As described above, this is the approach employed in the majority of heuristics that have been developed for the classical RCPSP.

In can be seen in Table 8.3 that our results are worse (up to almost 9 % for the instances consisting of 120 activities) than the best results reported on the website of the PSPLIB for all sets of problem instances. It should be noted, however, that these results have been obtained by a multitude of different heuristics. For this reason, we now compare our results to those of individual algorithms as they are listed by Kolisch and Hartmann (2006) in their computational study. A selection of these results for the problem instances consisting of 30, 60, and 120 activities is given in Table 8.4.

As this comparison reveals, the results obtained by our algorithm are worse than those obtained by state-of-the-art algorithms specifically designed for the classical RCPSP. Indeed, even the sampling method used by Tormos and Lova (2003) with a stopping condition of 1 000 schedules is able to obtain better results than our algorithm. This algorithm generates schedules using either the parallel or the serial schedule generation scheme and then improves the solution by either forward-backward improvement passes or backward-forward improvement passes. On the other hand, some of the earlier heuristics that do not employ forward-backward improvement (e.g. the tabu search algorithm introduced by Baar et al. (1998) or the sampling methods evaluated by Kolisch (1996a,b)) obtain similar results.

Next, we compare the results obtained by our algorithm to those obtained by the tabu search algorithm introduced by Artigues et al. (2003). As described in Section 3.2.3, solutions in this tabu search algorithm are also represented by resource flows. Based on this solution representation, they use modifications that remove individual activities from a resource flow and then reinsert them into the resource flow. As described by Artigues et al. (2003), the algorithm terminates after at most 5 000 iterations without an improvement of the best solution for the problem instances consisting of 60 activities as well as after at most 11 000 iterations without an improvement of the best solution for the problem instances consisting of 120 activities. Based on these settings, they report an average deviation of $\Delta = 12.05\,\%$

Approach	Reference	1 000	5 000	50 000
GA, TS	Kochetov and Stolyar (2003)	0.10	0.04	0.00
SS	Debels et al. (2006)	0.27	0.11	0.01
GA	Valls et al. (2008)	0.27	0.06	0.02
Sampling	Tormos and Lova (2003)	0.25	0.13	0.05
TS	Nonobe and Ibaraki (2002)	0.46	0.16	0.05
TS	Baar et al. (1998)	0.86	0.44	-

(a) Results for the problem instances consisting of 30 activities.

Approach	Reference	1 000	5 000	50 000
SS	Debels et al. (2006)	11.73	11.10	10.71
GA	Valls et al. (2008)	11.56	11.10	10.73
GA, TS	Kochetov and Stolyar (2003)	11.71	11.17	10.74
Sampling	Tormos and Lova (2003)	11.88	11.62	11.36
TS	Nonobe and Ibaraki (2002)	12.97	12.18	11.58
TS	Baar et al. (1998)	13.80	13.48	-

(b) Results for the problem instances consisting of 60 activities.

Approach	Reference	1 000	5 000	50 000
GA	Valls et al. (2008)	34.07	32.54	31.24
SS	Debels et al. (2006)	35.22	33.10	31.57
GA, TS	Kochetov and Stolyar (2003)	34.74	33.36	32.06
Sampling	Tormos and Lova (2003)	35.01	34.41	33.71
TS	Nonobe and Ibaraki (2002)	40.86	37.88	35.85

(c) Results for the problem instances consisting of 120 activities.

Table 8.4: A selection of results for the problem instances as they have been obtained by other heuristic algorithms (cf. Kolisch and Hartmann (2006)). Here, the average deviation (in percent) from the optimal solutions for the problem instances consisting of 30 activities is reported in (a), the average deviation (in percent) from the critical path lower bounds LB_0 for the problem instances consisting of 60 activities is reported in (b), and the average deviation (in percent) from the critical path lower bounds LB_0 for the problem instances consisting of 120 activities is reported in (c). These results are reported after a maximum of 1 000, 5 000, and 50 000 schedules have been generated.

from the critical path lower bounds for the problem instances consisting of 60 activities as well as an average deviation of $\Delta = 36.16\,\%$ from the critical path lower bounds for the problem instances consisting of 120 activities.

Thus, also this algorithm obtains better results than our algorithm. As before, however, it should be noted that this algorithm has been specifically designed for the classical RCPSP. In particular, activities are reinserted in optimal positions in relation to the current resource flow. As described in Section 6.2.2, the same property could not be ensured in polynomial time for the RCPSP with first- and second-tier resource transfers because the problem of selecting incoming (first- and second-tier) resource transfers to an activity is NP-hard.

Finally, we take a closer look at problem instances that are hard to solve. Here, problem instances with a large resource factor RF and a small resource strength RS are particularly hard to solve (cf. Hartmann and Kolisch (2000)). For these problem instances, a large resource factor RF indicates that the real activities require a large portion of the available resource types (i.e. a resource factor $RF = 1$ denotes that each real activity $i \in V$ has resource requirements $r_{ik} > 0$ for every resource $k \in \mathcal{R}$) while a small resource strength RS indicates that the available resources are scarce in relation to the resource requirements of the activities (i.e. a resource strength $RS = 0$ for a resource $k \in \mathcal{R}$ denotes that $r_{ik} = R_k$ holds for the resource requirements of a real activity $i \in V$).

In Table 8.5, we report the average deviation Δ_{best} (in percent) of the best solution obtained during either the first or the second pass of our algorithm for the problem instances with a resource factor $RF < 1.0$ and a resource strength $RS > 0.3$ as well as for the problem instances with a resource factor $RF < 0.75$ and a resource strength $RS > 0.3$, respectively. In the former case, the sets of problem instances consisting of 30, 60, and 90 activities contain a total of 450 instances while the set of problem instances consisting of 120 activities contains a total of 510 instances. In the latter case, all four sets of problem instances contain a total of 420 instances. As before, we compare our results to the average deviation Δ_{heur} (in percent) of the best solutions obtained by heuristic algorithms as reported on the website of the PSPLIB. In both cases, the average deviation is calculated from the optimal solutions for the problem instances consisting of 30 activities as well as from the critical path lower bounds LB_0 for the problem instances consisting of 60, 90, and 120 activities. Finally, we again report the number *opt* of problem instances for which the solution could be proved to be optimal.

As can be seen in this table, the results for problem instances with a smaller resource factor RF and a larger resource strength RS are better than those reported in Table 8.3 where all problem instances have been considered. This is due to the considerable influence these two parameters have on the solution space. On the one hand, the resource factor RF influences the number of resource types required by the real activities. Here, for a large resource factor RF, the activities require multiple types of resources in order to be processed. As a result of this, more resource transfers exist between the activities and the size of the solution space is increased.

n	Δ_{best} [%]	*opt*	Δ_{heur} [%]	*opt*
30	0.32	398	0.00	450
60	8.22	345	6.60	431
90	8.34	340	6.05	402
120	28.00	171	20.68	290

(a) $RF < 1.0$ and $RS > 0.3$.

n	Δ_{best} [%]	*opt*	Δ_{heur} [%]	*opt*
30	0.15	390	0.00	420
60	4.57	345	3.72	419
90	4.32	340	2.97	402
120	17.49	171	12.28	289

(b) $RF < 0.75$ and $RS > 0.3$.

Table 8.5: The results obtained by our algorithm as well as the best results obtained by heuristic algorithms as reported on the website of the PSPLIB for the problem instances with a resource factor $RF < 1.0$ and a resource strength $RS > 0.3$ are given in (a) while the results for a resource factor $RF < 0.75$ and a resource strength $RS > 0.3$ are given in (b).

On the other hand, the resource strength RS influences the tightness of the resource constraints. Here, for a large resource strength RS, it is more likely that activities can be started as early as possible with respect to the given precedence constraints (i.e. a schedule generation scheme is generally already able to obtain good solutions) while a small resource strength RS makes this less likely. In particular, the order in which activities are processed by the available resource units is very important in order to obtain a good solution for problem instances with a small resource strength RS.

As a result of this, the problem instances with a large resource factor RF as well as a small resource strength RS are particularly hard to solve for our algorithm because the number of resource transfers between the activities in the resource flow is generally larger than for a smaller resource factor RF (i.e. the size of the solution space is increased) and the order in which activities are processed by the available resource units is more important in order to obtain a good solution (i.e. very specific modifications are required in order to obtain a good solution). While the same problem also applies for other heuristic algorithms (cf. Hartmann and Kolisch (2000)), it does not increase the size of the solution space for heuristics that represent solutions as activity lists.

8.1.2 RCPSP with First-Tier Resource Transfers

In this section, we evaluate the performance of the tabu search algorithm for problem instances of the RCPSP with first-tier resource transfers. For this, we use the extended sets of problem instances consisting of 30 as well as 60 activities as they have been generated by Krüger and Scholl (2009). These sets of problem instances have been generated as follows.

First of all, Krüger and Scholl (2009) have tried to solve as many problem instances as possible from the sets of problem instances consisting of 30 as well as 60 activities that can be obtained from the website of the PSPLIB for the classical RCPSP (i.e. the same problem instances that have been used above). For this, they have used a branch-and-bound algorithm with a time limit of 3 600 s. Here, optimal solutions could be obtained for all problem instances consisting of 30 activities as well as for 400 problem instances consisting of 60 activities. These problem instances have then been extended by symmetric transfer times $\Delta_{ijk} = \Delta_{jik}$ between all pairs of activities $i, j \in V$ as well as for all resources $k \in \mathcal{R}$ such that the triangle inequality $\Delta_{hik} + \Delta_{ijk} \geq \Delta_{hjk}$ holds for these transfer times and the optimality of the solution found by the branch-and-bound algorithm as well as the makespan of this solution are retained in the extended problem instance.

Now, in order to evaluate these problem instances, we use the same settings for our tabu search algorithm as they have been described in Section 8.1.1. In particular, we again perform two passes over the problem instances based on the parameters given in Table 8.1. The results obtained by our algorithm are reported in Table 8.6. As before, we report the average deviation Δ (in percent) from the optimal solutions as well as the average time t (in seconds)

Iter.	0		100		500		1 000		5 0000		10 000	
n	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]
30	8.27	0.0	4.08	0.1	3.00	0.6	2.65	1.1	1.52	4.8	1.27	8.8
60	5.79	0.0	2.72	0.5	1.94	2.3	1.66	4.3	1.14	17.3	0.98	31.8

(a) Results for the first pass (parameter settings 1).

Iter.	0		100		500		1 000		5 0000		10 000	
n	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]	Δ [%]	t [s]
30	8.27	0.0	3.95	0.1	3.07	0.6	2.66	1.1	1.59	4.9	1.30	9.1
60	5.79	0.0	2.75	0.5	2.09	2.3	1.89	4.4	1.35	19.8	1.21	37.4

(b) Results for the second pass (parameter settings 2).

Table 8.6: Results obtained by the tabu search algorithm for the problem instances consisting of 30 as well as 60 activities that have been extended by Krüger and Scholl (2009) for the RCPSP with first-tier resource transfers. The results for the first pass based on the parameter settings from Table 8.1(a) are reported in (a) while the results for the second pass based on the parameter settings from Table 8.1(b) are reported in (b). It should be noted that the average deviation Δ is calculated from the optimal solutions for both sets of problem instances.

required by the tabu search algorithm after 0, 100, 500, 1 000, 5 000, and 10 000 iterations. It should be noted that here, the optimal solutions are known for both sets of problem instances.

In Table 8.7, we now compare the average deviation Δ_{best} (in percent) of the best solutions obtained by our algorithm during either the first or the second pass to the results obtained by other heuristic solution approaches. Here, we can only compare our results to those reported by Krüger and Scholl (2009) as well as Krüger (2009) because no further results for these problem instances have been reported to the best of our knowledge. In particular, we compare our results to the average deviation Δ_{SGS} (in percent) obtained by a multi-pass heuristic as well as to the average deviation Δ_{GA} (in percent) obtained by a genetic algorithm as they have been reported by Krüger (2009). Here, for the multi-pass heuristic, she uses both the serial as well as the parallel schedule generation scheme with forward-backward improvement as well as all possible combinations of priority rules (both, priority rules to select activities as well as priority rules to select resource transfers) in order to solve the problem instances. The best solutions obtained by any of these combinations are then used in order to calculate the average deviation Δ_{SGS}. Finally, the genetic algorithm is a one-gene self adapting algorithm for which both, the activity list as well as the schedule generation scheme used to transform an activity list into a schedule can be modified. For this algorithm, the results are reported for a stopping condition of 5 000 schedules.

n	Δ_{best} [%]	*opt*	Δ_{SGS} [%]	*opt*	Δ_{GA} [%]	*opt*
30	1.10	382	1.63	322	0.16	459
60	0.82	339	1.61	284	0.38	355

Table 8.7: Average deviation Δ_{best} of the best solution obtained by our algorithm during either the first or the second pass. These results are compared to the average deviation Δ_{SGS} obtained by a multi-pass heuristic as well as the average deviation Δ_{GA} obtained by a genetic algorithm as they have been reported by Krüger (2009).

As can be seen here, the results obtained by our algorithm are again worse than the results obtained by the genetic algorithm introduced by Krüger (2009). The difference is particularly large for the problem instances consisting of 30 activities for which our algorithm is on average 1 % worse than the genetic algorithm. As before, this is due to the fact that problem instances with a large resource factor RF as well as a small resource strength RS are particularly hard to solve for our algorithm. Here, in Table 8.8, we report the average deviation Δ_{best} (in percent) of the best solution obtained by our algorithm during either the first or the second pass over the 30 hard problem instances with transfer times consisting of 30 activities with a resource factor $RF = 1.0$ and a resource strength $RS = 0.2$. We compare these results to the average deviation Δ_{sampl} (in percent) as well as the average deviation Δ_{GA} (in percent) obtained by either a random sampling procedure or by a one-gene self adapting genetic algorithm as they have been reported by Krüger (2009) for a stopping condition of 5 000 schedules.

Δ_{best} [%]	*opt*	Δ_{sampl} [%]	*opt*	Δ_{GA} [%]	*opt*
6.3	3	5.41	1	1.03	22

Table 8.8: The results obtained by our algorithm as well as the results obtained by a random sampling procedure and a genetic algorithm as they have been reported by Krüger (2009) for the 30 hard problem instances consisting of 30 activities with a resource factor $RF = 1.0$ and a resource strength $RS = 0.2$.

These results highlight the problem of our algorithm to solve hard problem instances with a large resource factor RF and a small resource strength RS. In particular, even the random sampling method obtains a better average deviation than our algorithm (with a difference of almost 1 %). At the same time, it can be seen in Table 8.7 that the results obtained by our algorithm for the problem instances consisting of 60 activities are better than those obtained by the multi-pass heuristic and less than 0.5 % worse than those obtained by the genetic algorithm. This is due to the fact that a lot of the hard problem instances are not contained in this set of problem instances. Thus, as already stated in Section 8.1.1, our algorithm is able to obtain good results for problem instances with a smaller resource factor RF and a larger resource strength RS.

8.1.3 RCPSP with First- and Second-Tier Resource Transfers

In order to evaluate the performance of the tabu search algorithm for the RCPSP with first- and second-tier resource transfers, we first have to generate problem instances for this problem because no problem instances exist in literature to the best of our knowledge. For this, we have first generated problem instances for the classical RCPSP using the project scheduling problem instance generator (ProGen) introduced by Kolisch et al. (1995) and available on the website of the PSPLIB. These problem instances have then been extended into problem instances for the RCPSP with first- and second-tier resource transfers by generating support requirements and transfer times for these instances as described below.

First of all, we have generated a total of 24 problem instances for the classical RCPSP using the instance generator ProGen. For this, we have used similar base parameter settings as those used by Kolisch and Sprecher (1997) for the single-mode problem instances for the classical RCPSP consisting of 30 and 60 activities. In particular, $r = 4$ renewable resources are available such that each activity requires between $U^{min} = 1$ and $U^{max} = 10$ units of between $Q^{min} = 1$ and $Q^{max} = 4$ of these resources. Furthermore, each activity $i \in V$ is assigned a processing time between $p_i^{min} = 1$ and $p_i^{max} = 10$. We have generated two sets of problem instances consisting of 10 as well as 20 real activities using the combinations of network complexity NC, resource factor RF, and resource strength RS given in Table 8.9.

	NC	RF	RS
1	1.5	0.5	0.2
2	1.5	0.5	0.5
3	1.5	0.5	0.7
4	1.5	1.0	0.2
5	1.5	1.0	0.5
6	1.5	1.0	0.7

Table 8.9: Combinations of network complexity NC, resource factor RF, and resource strength RS used to generate the problem instances for the RCPSP with first- and second-tier resource transfers.

For each of these combinations of parameters, we have generated two problem instances. In the following, we use the same naming conventions as those used by Kolisch and Drexl (1997) for these problem instances. Then, the first problem instance for parameter combination 3 consisting of 20 activities is referred to as problem instance $j203\_1$ while the second problem instance for this parameter combination consisting of 20 activities is referred to as $j203\_2$.

Next, we extended these problem instances by support requirements. For this, we first define the set $\mathcal{R}^{\mathrm{sa}} = \{1, 2\}$ of resources that can be used to support the transfer of other resources as well as the set $\mathcal{R}^{\mathrm{ru}} = \{3, 4\}$ of resources that require supporting resources. Now, we scale down the resource availabilities and resource requirements for resources $l \in \mathcal{R}^{\mathrm{ru}}$ by a factor $\sigma = 3$ to $R_l = \lceil {}^{R_l}/_{\sigma} \rceil$ and $r_{il} = \lceil {}^{r_{il}}/_{\sigma} \rceil$ for all real activities $i \in V$. Afterward, we randomly choose support requirements μ_{kl} for $k = 1, 2$ and $l = 3, 4$ such that the following conditions are fulfilled. First of all, it has to be ensured that $\mu_{kl} > 0$ holds for at least one resource $k \in \mathcal{R}^{\mathrm{sa}}$ for each resource $l \in \mathcal{R}^{\mathrm{ru}}$. Otherwise, no supporting resources would be required for the transfer of resource l. Additionally, it has to be ensured that

$$\sum_{l \in \mathcal{R}^{\mathrm{ru}}} \mu_{kl} \cdot R_l \leq R_k$$

holds for each resource $k \in \mathcal{R}$. In particular, this latter condition ensures that it is still possible to satisfy all support requirements of resources $l \in \mathcal{R}^{\mathrm{ru}}$ if all units of these resources are required by an activity.

Finally, we extended the problem instances by transfer times. Here, we set $\Delta_{0jk} = 0$ for the transfer time of resource $k \in \mathcal{R}$ from dummy source activity 0 to all activities $j \in V_{\mathrm{all}}$ and $\Delta_{i,n+1,k} = 0$ for the transfer time of resource $k \in \mathcal{R}$ from all activities $i \in V_{\mathrm{all}}$ to dummy sink activity $n+1$. The remaining transfer times Δ_{ijk} for resources $k \in \mathcal{R}$ between real activities $i \in V$ and $j \in V$ have to be chosen such as to fulfill the following conditions. First of all, $\Delta_{iik} = 0$ has to hold for the transfer time of resource $k \in \mathcal{R}$ from each activity $i \in V$ to itself. Then, the triangle inequality $\Delta_{hik} + \Delta_{ijk} \geq \Delta_{hjk}$ has to hold for all transfer times of resource $k \in \mathcal{R}$ between activities $h, i, j \in V$. Finally, $\Delta_{ijk} \leq \Delta_{ijl}$ has to hold for the transfer of resources $k \in \mathcal{R}^{\mathrm{sa}}$ and $l \in \mathcal{R}^{\mathrm{ru}}$ between activities $i \in V$ and $j \in V$ if $\mu_{kl} > 0$ holds for the amount of resource units of resource k required to support the transfer of resource l.

In order to generate these transfer times, we have generated random pairs of values (x_i, y_i) with $0 \leq x_i, y_i \leq 4$ for each activity $i \in V$. Additionally, we have generated four random values $\delta_k \in]0, 1]$ (with k = 1,2,3,4) and sorted these according to increasing numbers. Then we have calculated transfer times

$$\Delta_{ijk} = \min \left\{ 5, \left\lceil \delta_k \cdot \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \right\rceil \right\}$$

for all resources $k \in \mathcal{R}$ between all pairs of activities $i \in V$ and $j \in V$. This ensures that all conditions listed above are satisfied for the generated transfer times. All transfer times generated here take values between $\Delta^{min} = 0$ and $\Delta^{max} = 5$. At the same time, however, only a small portion of these transfer times are equal to zero because values are generally rounded up to the next integer value.

Now, we evaluate the performance of our tabu search algorithm based on these problem instances. For this, we use the neighborhood $\tilde{\mathcal{N}}_4 = \tilde{\mathcal{N}}^{\text{max,ca}}_{\text{reroute}} \cup \tilde{\mathcal{N}}_{\text{reverse}}$ as it has been introduced in Section 6.2.4. The base parameter settings (i.e. the stopping condition as well as the parameters related to the intensification and diversification strategies) are the same as described in Section 8.1.1. Finally, we perform only one pass using the parameters a, b, α, and β given in Table 8.10. These parameters have again been selected based on some preliminary tests.

n	a	α	b	β
10	5	0.2	2	0.0
20	7	0.2	3	0.0

Table 8.10: Parameter settings for parameters a, b, α, and β used to evaluate the problem instances.

In order to compare the results obtained by our algorithm, we have modeled all problem instances as mixed-integer linear programs (cf. Section 5.2.2 for the mixed-integer linear programming formulation) using the Zimpl modeling language (cf. Koch (2004), made available by the Zuse Institute Berlin) and solve them using IBM ILOG CPLEX 12.1 with a time limit of 3 600 s. The results obtained by both our algorithm as well as CPLEX for the problem instances consisting of 10 activities are reported in Table 8.11 while the results for the problem instances consisting of 20 activities are reported in Table 8.12. Here, apart from reporting the makespan *obj* obtained for each problem instance as well as the required time t (in seconds), we also report the deviation Δ (in percent) from the best lower bound LB that has been found by CPLEX.

Now, we first consider the results for the problem instances consisting of 10 activities. Here, CPLEX has been able to obtain optimal solutions for 9 problem instances and feasible solutions for 2 more problem instances (i.e. only for one problem instance, no feasible solution could be generated within the time limit). In comparison to this, our algorithm could only find optimal solutions for 6 problem instances within 10 000 iterations. It should

		CPLEX			0			1 000			10 000		
Inst.	*LB*	*obj*	*Δ* [%]	*t* [*s*]	*obj*	*Δ* [%]	*t* [*s*]	*obj*	*Δ* [%]	*t* [*s*]	*obj*	*Δ* [%]	*t* [*s*]
*j*101_1	54*	54	0.00	113.4	68	25.93	0.0	59	9.26	2.2	54	0.00	10.0
*j*101_2	44*	44	0.00	4.9	51	15.91	0.0	49	11.36	0.5	49	11.36	5.1
*j*102_1	29*	29	0.00	15.0	33	13.79	0.0	29	0.00	0.8	29	0.00	8.7
*j*102_2	38*	38	0.00	3.9	47	23.68	0.0	38	0.00	0.6	38	0.00	6.5
*j*103_1	36*	36	0.00	2.7	43	19.44	0.0	36	0.00	0.4	36	0.00	4.7
*j*103_2	27*	27	0.00	1.6	34	25.93	0.0	27	0.00	0.6	27	0.00	6.0
*j*104_1	40	–	–	3600.0	67	67.50	0.0	65	62.50	1.4	65	62.50	13.0
*j*104_2	41	64	56.10	3600.0	68	65.85	0.0	67	63.41	1.3	61	48.78	13.4
*j*105_1	31*	31	0.00	1245.2	38	22.58	0.0	32	3.23	1.7	31	0.00	16.3
*j*105_2	39	44	12.82	3600.0	46	17.95	0.0	44	12.82	1.4	44	12.82	15.0
*j*106_1	34*	34	0.00	354.2	35	2.94	0.0	35	2.94	1.5	35	2.94	15.9
*j*106_2	32*	32	0.00	1136.4	36	12.50	0.0	33	3.13	1.8	33	3.13	17.0
						26.17	0.0		14.04	1.2		11.79	11.0

Table 8.11: Results for the problem instances consisting of 10 activities as they have been obtained by both CPLEX 12.1 within a time limit of 3 600 *s* as well as by our tabu search algorithm after 0, 1 000, and 10 000 iterations. Apart from the makespan *obj* and the required time *t* (in seconds), we report the deviation *Δ* (in percent) from the best lower bound *LB* that has been found by CPLEX. If the lower bound *LB* for a problem instance could be proved to be optimal, it has been marked with an asterisk.

		CPLEX			0			1 000			10 000		
Inst.	*LB*	*obj*	*Δ* [%]	*t* [*s*]	*obj*	*Δ* [%]	*t* [*s*]	*obj*	*Δ* [%]	*t* [*s*]	*obj*	*Δ* [%]	*t* [*s*]
*j*201_1	37	77	108.11	3600.0	71	91.89	0.0	64	72.97	7.1	62	67.57	64.4
*j*201_2	46	–	–	3600.0	84	82.61	0.0	77	67.39	5.6	67	45.65	55.6
*j*202_1	45	–	–	3600.0	76	68.89	0.0	67	48.89	7.4	63	40.00	72.0
*j*202_2	44	61	38.64	3600.0	62	40.91	0.0	58	31.82	6.9	56	27.27	68.3
*j*203_1	44	55	25.00	3600.0	62	40.91	0.0	55	25.00	8.5	54	22.73	85.5
*j*203_2	44	49	11.36	3600.0	60	36.36	0.0	50	13.64	7.6	49	11.36	72.5
*j*204_1	44	–	–	3600.0	111	152.27	0.0	109	147.73	9.0	109	147.73	89.7
*j*204_2	35	–	–	3600.0	72	105.71	0.0	70	100.00	12.5	68	94.29	119.4
*j*205_1	45	–	–	3600.0	80	77.78	0.0	76	68.89	9.4	72	60.00	98.7
*j*205_2	41	–	–	3600.0	68	65.85	0.0	66	60.98	11.2	64	56.10	120.2
*j*206_1	31	–	–	3600.0	47	51.61	0.0	44	41.94	11.8	44	41.94	121.4
*j*206_2	33	–	–	3600.0	52	57.58	0.0	48	45.45	11.4	46	39.39	110.4
						72.70	0.0		60.39	9.0		54.50	89.8

Table 8.12: Results for the problem instances consisting of 20 activities as they have been obtained by both CPLEX 12.1 within a time limit of 3 600 *s* as well as by our tabu search algorithm after 0, 1 000, and 10 000 iterations. Apart from the makespan *obj* and the required time *t* (in seconds), we report the deviation *Δ* (in percent) from the best lower bound *LB* that has been found by CPLEX. Here, none of the solutions (and hence none of the lower bounds) could be proved to be optimal.

be noted, however, that the deviation of the makespan obtained by our algorithm for problem instances $j106\_1$ and $j106\_2$ from the makespan of the optimal solution is only one time unit. At the same time, our algorithm could generate better solutions for the problem instances that CPLEX has been unable to solve to optimality. As before, it can be noted that problem instances with a large resource factor RF and a small resource strength RS are especially hard to solve for both CPLEX as well as our algorithm. In particular, this can be seen for problem instances with a resource factor $RF = 1.0$ (i.e. if $r_{ik} > 0$ for all $k \in \mathcal{R}$ holds for all real activities $i \in V$).

Next, we consider the results for the problem instances consisting of 20 activities. Here, CPLEX has been unable to find an optimal solution for any problem instance. Moreover, CPLEX could only obtain feasible solutions for 4 of the problem instances within the time limit. On the other hand, our algorithm could obtain and improve feasible solutions for all problem instances. In particular, it should be noted that the solution obtained by our algorithm is always better or at least as good as the solution obtained by CPLEX.

Based on these results, we can see that our algorithm is able to obtain good results for the RCPSP with first- and second-tier resource transfers also for larger problem instances. Furthermore, the tabu search algorithm has been able to improve the average deviation of the makespan of the initial solutions generated by the parallel schedule generation from the lower bounds LB by more than 15 % within 10 000 iterations for both sets of problem instances. This is a much larger improvement than for any of the sets of problem instances considered in Sections 8.1.1 and 8.1.2 and implies that the schedule generation scheme is not able to generate good solutions for the RCPSP with first- and second-tier resource transfers based on priority rules.

Finally, we consider the computational time required by our algorithm. It can be seen that the average time required to perform 10 000 iterations for the problem instances consisting of 10 activities is already very large with $t_{avg} = 11\,s$. The reason for this is twofold. On the one hand, redirecting resource transfers of resources $l \in \mathcal{R}^{\text{ru}}$ that require supporting resources requires more computational time than for resources that do not require supporting resources because apart from modifying the corresponding resource transfers, it is also necessary to redirect supporting resource transfers and possibly to repair the graph as described in Section 6.2.4. More importantly, however, the number of resource transfers between the activities can be much larger for the RCPSP with first- and second-tier resource transfers than for either the classical RCPSP or the RCPSP with first-tier resource transfers. In particular, while at most one resource transfer of a resource $k \in \mathcal{R}$ can exist between two activities $h \in V_0$ and $j \in V_*$ for the latter two problems, up to $2 + 2n \cdot |\mathcal{R}^{\text{ru}}|$ resource transfers of a resource $k \in \mathcal{R}^{\text{sa}}$ can exist between two activities $h \in V_0$ and $j \in V_*$ for the RCPSP with first- and second-tier resource transfers (i.e. two first-tier resource transfers from either the start or the end of activity h to activity j as well as $2n \cdot |\mathcal{R}^{\text{ru}}|$ second-tier resource transfers from either the start or the end of activity h that support the transfer of a resource $l \in \mathcal{R}^{\text{ru}}$ from activity $i \in V_0$ to activity j). This has a large impact on the number of reroute modifications in the neighborhood $\mathcal{N}^{\text{max,ca}}_{\text{reroute}}$ that have to be evaluated in each iteration.

8.2 Results for the Problem of Hospital Evacuations

In this section, we evaluate the performance of the two solution approaches for the problem of hospital evacuations that have been introduced in Section 4.2 as well as in Chapter 7, respectively. For this, in Section 8.2.1, we first describe how test data for this problem has been generated. Afterward, in Section 8.2.2, the computational results obtained for this test data are reported.

8.2.1 Generation of Test Data

In this section, we describe how the test data for the problem of hospital evacuations has been generated. First of all, we consider the infrastructure of the hospital itself. Here, we have been able to obtain floor plans for the Asklepios Harzklinik Goslar, a hospital for acute care services in the city of Goslar that is providing a total of 331 beds. Due to the size of the hospital, we have opted to only model a part of it for the tests performed in this section. In particular, we have modeled the third floor of the hospital which is consisting of a total of 34 sick rooms. In the following, between one and four patients can be accommodated in each sick room such that a maximum of 73 patients can be placed in these sick rooms at any time. Here, the actual number of beds per sick room could also be obtained from the floor plan. Finally, two elevators as well as five stairs have been modeled while another two elevators as well as another stair have been omitted from the model. A rough sketch of this floor is displayed in Figure 8.2.

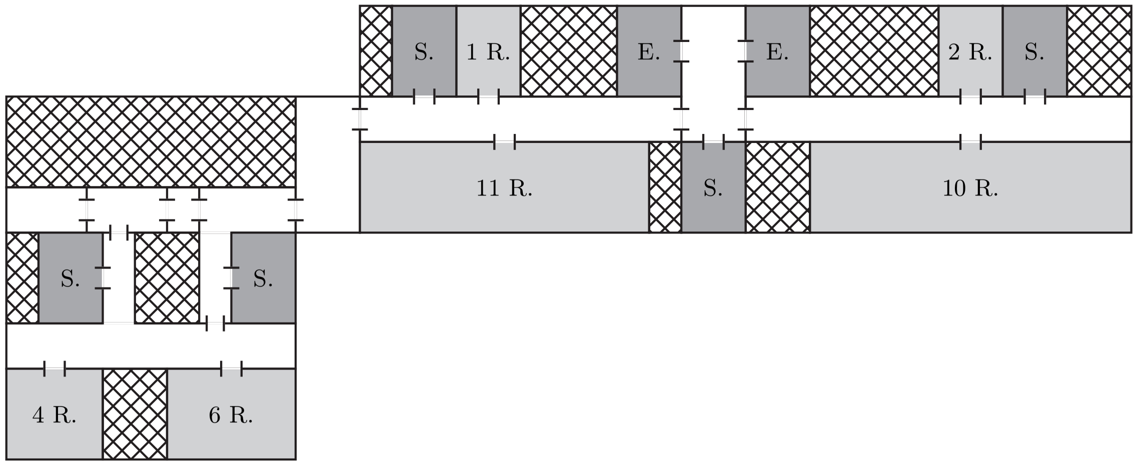


Figure 8.2: A rough sketch of a part of the third floor of the Asklepios Harzklinik Goslar as it is used for the evaluation of the solution approaches. The third floor considered here consists of a total of 34 sick rooms (abbreviated as R.) as well as two elevators (abbreviated as E.) and five stairs (abbreviated as S.). Two further elevators as well as one stair have been omitted. In each sick room, between one and four patients can be accommodated. The actual numbers have been obtained from the floor plan but are omitted in this sketch. Furthermore, this sketch displays where doors have to be taken into account (i.e. either doors connecting a room or a staircase to a corridor section or doors connecting two corridor sections). Finally, crosshatched areas in this figure denote other rooms that do not have to be considered in the model.

Now, the corridors displayed in this Figure are divided into a total of 38 corridor sections that connect the various locations considered in this model. The length of the individual corridor sections has been obtained from the floor plan and is later used in order to calculate the time required to evacuate a patient through the corridor section by the required assistants and aids. Finally, a building section such as a sick room or a staircase is always connected to the corresponding corridor section by a door while additional doors might exist between two consecutive corridor sections. These doors are also visualized in Figure 8.2.

In the following, we assume that all stairs as well as the two elevators correspond to safety zones. Thus, as soon as a patient has been evacuated through the corresponding building section, he is assumed to be safe and the assistants and aids used for the evacuation of the patient are available again. Finally, the capacity of a building section (i.e. the available space inside this building section) is selected as follows. First of all, each sick room is assumed to have a sufficient amount of space such that all patients inside this sick room can simultaneously be prepared for the evacuation regardless of the required aids. Next, corridor sections as well as staircases have a capacity proportional to their length. Here, it should be noted that corridors generally have a width of at least $2.5\,m$ such that two hospital beds can pass each other (cf. Wolf (2001)). Finally, it is assumed that each elevator can be used for the evacuation of at most one patient at any time regardless of the aid used for the evacuation of the patient.

Next, we consider the assistants and aids available for the evacuation of the patients. Here, we assume that only one type of assistants is used for the evacuation of the patients (i.e. we neglect the case that some patients might require the supervision of assistants with special skills). Furthermore, we assume that a limited amount of wheelchairs and stretchers is available for the evacuation of the patients. In the following, we assume that one assistant is required in order to support the transfer of one stretcher or one wheelchair. For another test performed in Section 8.2.2, we additionally assume that some patients can also be evacuated in their hospital beds. In this case, a hospital bed can be assumed to be available for each patient in his or her sick room such that hospital beds do not have to be modeled as scarce resources.

In order to evacuate a patient from his or her initial location to a safety zone, the following combinations of assistants and aids are possible depending on the severity of the condition of the patient. First of all, if a patient can walk, it is sufficient if the evacuation of the patient is aided by one assistant. Otherwise, patients can either be evacuated in a sitting position (i.e. in a wheelchair with the help of one assistant) or in a lying position (i.e. on a stretcher with the help of three assistants or on a bed with the help of two assistants). Various parameters associated with these combinations are given in Table 8.13. These values are approximations of values as they have been reported by Wolf (2001) and are later used in order to calculate resource requirements as well as processing and transfer times.

	r_{space}	t_{prep}	t_{door}	v_{empty}	v_{evac}
Assistant	1	60	4	1.4	1.0
Wheelchair	2	80	7	1.2	1.2
Stretcher	3	100	4	1.2	1.3
Bed	3	60	6	–	1.0

Table 8.13: Parameters related to the possible combinations of assistants and aids that can be used for the evacuation of a patient. Here, r_{space} (in square meters) denotes the space required in a building section for the evacuation of a patient, t_{prep} (in seconds) denotes the time required to prepare the patient before the evacuation, t_{door} (in seconds) denotes the time required to pass through a door, v_{empty} (in $^{\text{meters}}/_{\text{second}}$) denotes the speed with which the assistants and aids can be transferred between the evacuation of two patients, and v_{evac} (in $^{\text{meters}}/_{\text{second}}$) denotes the speed with which the required assistants and aids can evacuate a patient.

Here, the space required to evacuate a patient through a building section using either of these combinations of assistants and aids is denoted by r_{space} (in square meters). As described above, this amount of space is required in any building section the patient is evacuated through with the exception of an elevator (which can only be used to evacuate one patient at a time independent of the required space).

Next, the time required to evacuate a patient through a specific building section can be calculated for each of these combinations. These times correspond to the processing times of the operations and are calculated based on equation (4.1). For this, first of all, the time (in seconds) required to evacuate a patient through a corridor section is calculated by dividing the length of the corridor section (in meters) by the speed v_{evac} (in $^{\text{meters}}/_{\text{second}}$). In the following, this result is always rounded up to the next integer value.

Additionally, the time t_{door} (in seconds) has to be taken into account if a door exists between two consecutive building sections while the time t_{prep} (in seconds) has to be included at the beginning of the evacuation of a patient in order to prepare the patient for the evacuation. Finally, we add a fixed time of $60\,s$ if an elevator has to be used for the evacuation of a patient as well as a fixed time of $90\,s$ if stairs have to be used. These latter values have been selected due to the problem that we have been unable to obtain more realistic values for these two types of building sections.

Finally, the transfer times for assistants and aids between the different locations in the hospital can be computed. For this, we first calculate a shortest path between each pair of locations (as denoted by the length of the building sections). Then, we calculate the sum of the lengths of the corridor sections of the shortest path between two locations (in meters) and divide this value by the speed v_{empty} (in $^{\text{meters}}/_{\text{second}}$). As before, the resulting time (in seconds) is rounded up to the next integer value. To this, we add the corresponding time t_{door} (in seconds) for each door along the route as well as a fixed time of $60\,s$ for each time an elevator or stairs have to be used for the transfer of either assistants or aids.

In the following, all of this data remains the same for all problem instances. As a result of this, only the number of patients (as well as their initial location and the severity of their condition) and the amounts of available assistants and aids can be varied in order to generate problem instances. Now, we created three sets of problem instances such that the first set consists of problem instances in which 30 patients have to be evacuated, the second set consists of problem instances in which 50 patients have to be evacuated, and the third set consists of problem instances in which 70 patients have to be evacuated. The initial location for each patient (i.e. his or her sick room) has been selected randomly such that in each room, no more than the corresponding maximal amount of patients can be located.

Additionally, approximately 60 % of the patients can walk, 20 % of the patients have to be evacuated while sitting, and 20 % of the patients have to be evacuated while lying. These values have again been selected based on the values reported by Wolf (2001). Here, patients that can walk always have to be evacuated by one assistant (i.e. no aids are required for the evacuation of these patients). Thus, only one equipment mode m_1 exists for the corresponding jobs. Additionally, these patients can be evacuated via both stairs as well as elevators. For this reason, we computed the shortest evacuation route from the initial location of each patient to each safety zone and modeled these evacuation routes as chains of operations with the resource requirements and processing times described above. Thus, a total of seven route modes m_2 (i.e. one for each safety zone) are available for each of these patients. Next, patients that can be evacuated while sitting can either be evacuated in a wheelchair, on a stretcher, or in a hospital bed (if hospital beds are considered) while patients that have to be evacuated while lying can only be evacuated on either a stretcher or in a hospital bed. In both cases, these patients can only be evacuated by elevator, i.e. there are only two possible safety zones to which these patients can be evacuated.

Finally, the amounts of available assistants and aids have been selected. Here, we considered the following three cases. In the first case, a small amount of assistants and aids is available (i.e. between 5 to 10 assistants as well as between 1 to 3 wheelchairs and stretchers). Then, in the second case, a medium amount of assistants and aids is available (i.e. between 10 to 20 assistants as well as between 3 to 6 wheelchairs and stretchers). Finally, in the third case, a large amount of assistants and aids is available (i.e. between 20 to 40 assistants as well as between 6 to 10 wheelchairs and stretchers). The actual values for all three cases have been selected randomly for each problem instance.

Based on these settings regarding the number of patients as well as the available amounts of assistants and aids, a total of 9 combinations are possible. For each of these combinations, we have generated two problem instances such that a total of 18 problem instances have been created. In the following, we use a similar naming convention for these problem instances as for the problem instances for the classical RCPSP from the PSPLIB. In particular, the first problem instance consisting of 50 jobs for the second case (i.e. if a medium amount of assistants and aids is available) is referred to as $h502\_1$ while the second problem instance consisting of 50 jobs for the second case is referred to as $h502\_2$.

It should be noted that a lot of the parameters used in this model are only approximations of real values. Also, various simplifications and assumptions have been made that do not necessarily have to hold in reality. For example, the time required to evacuate a patient through specific building sections is generally more complex and depends on specific properties of the hospital, e.g. the width of the corridor and doors. Furthermore, additional regulations (e.g. public building emergency regulations) might have to be taken into account for some hospitals regarding the evacuations of the patients. Due to a lack of such detailed information, we have opted to restrict our evaluation to the base parameters described above. For a more realistic scenario, however, it is easily possible to adapt many of these settings in order to meet the requirements without having to change the model.

8.2.2 Computational Results

Now, we evaluate the performance of the two solution approaches for the problem of hospital evacuations based on the test data that has been generated as described above. Here, first of all, we use the solution approach based on priority rules as it has been described in Section 4.2 in order to solve the problem instances. In particular, we use the tabu search algorithm with either the serial or the parallel schedule generation scheme as well as all five priority rules listed in Table 4.2 (i.e. the priority rules SPT, LPT, LST, LFT, and RAND). As a result of this, we perform a total of 10 tests for this solution approach (i.e. combining each of the five priority rules with each of the two schedule generation schemes).

In order to evaluate the problem instances, we have set up the tabu search algorithm as follows. First of all, the minimal length of the tabu list is set to $TL_{min} = 10$ while the maximal length is set to $TL_{max} = 30$. The actual length of the tabu list is then adapted according to equations (4.34) and (4.35). Next, $l_{max} = 1$ elite solution can be stored. The tabu search algorithm continues from this elite solution if the currently best solution could not be improved in the last $t_{max} = 50$ iterations. Finally, we use a stopping condition of approximately $120\,s$ for the algorithm. We have decided to use this stopping condition because it allows for a more fair comparison of the different solution approaches.

Then, the problem instances have been solved based on these settings. Here, for a first test, we have set up the problem instances such that patients that can not walk can either be evacuated in a wheelchair or on a stretcher, i.e. no hospital beds can be used. Below, the results obtained by the tabu search algorithm using the different combinations of schedule generation schemes and priority rules are reported in Table 8.14. As can be seen from these results, the tabu search algorithm obtains much better results if it is using the parallel schedule generation scheme than if it is using the serial schedule generation scheme. This is due to the fact that the serial schedule generation scheme is not able to insert a selected operation between other operations that have already been scheduled and require the same resources (cf. Section 4.2.2). In particular, this is not possible because the completion time

	Parallel Schedule Generation Scheme					Serial Schedule Generation Scheme				
Instance	SPT	LPT	LST	LFT	RAND	SPT	LPT	LST	LFT	RAND
h301_1	1932	1886	2168	1931	1932	2163	1638	1740	1609	1606*
h301_2	2031	2031	2031	2031	2031	2581	2031	2018*	2043	2024
h302_1	814*	814*	814*	814*	814*	1136	1179	2563	983	1157
h302_2	1112	1139	1087*	1146	1125	1330	1949	1894	1212	1465
h303_1	674*	674*	674*	674*	674*	674*	693	674*	674*	717
h303_2	566*	566*	566*	566*	566*	632	909	1721	798	774
h501_1	3570	3525	3538	3538	3553	5024	3668	3284*	3284*	4243
h501_2	2607	2607	2607	2607	2317*	2867	3602	6841	2553	3329
h502_1	1983	2210	2138	2002	1876*	2316	2840	3362	2098	2774
h502_2	1524	1496	1543	1414*	1447	1742	2144	2990	1749	2239
h503_1	907	869*	869*	888	894	1224	1592	1918	1226	1502
h503_2	974*	974*	1122	1062	974*	1540	1266	1329	1243	1598
h701_1	4413*	4413*	4413*	4413*	4419	6063	5724	6962	4575	5759
h701_2	4208*	4351	4643	4611	4766	5628	5144	10924	4317	5833
h702_1	2578	2578	2578	2578	2568*	3100	5213	5275	2940	3775
h702_2	3020	2858*	2860	2859	2898	3755	3759	5838	3266	3946
h703_1	1414	1414	1389*	1414	1400	2222	1928	2842	1471	2234
h703_2	1597	1597	1597	1597	1592*	2245	2038	2021	1707	2460

Table 8.14: Results obtained by the tabu search algorithm that has been introduced in Section 4.2 for all possible combinations of schedule generation schemes and priority rules. The best solution that has been found by any of these combinations is marked with an asterisk. The problem instances for these tests have been set up such that no hospital beds could be used in order to evacuate the patients.

of the operation is not known at the time when it is being scheduled due to blockings. As a result of this, the performance of the serial schedule generation scheme depends more strongly on the order in which the operations are scheduled than the performance of the parallel schedule generation scheme.

Before we take a closer look at individual results, we now use the solution approach that has been introduced in Chapter 7 in order to solve the problem instances. For this solution approach, we use the following settings. First of all, the stopping condition for the algorithm is again set to $120\,s$ (this corresponds to stopping condition $SC3$). Then, the tabu search algorithm for the evacuation subproblem is set up to call the tabu search algorithm for the routing subproblem if no improving solution could be found in the last 300 iterations (this corresponds to stopping condition $SC1$). Finally, the tabu search algorithm for the routing subproblem terminates after either no improving solution could be generated in the last 10 iterations or after at most $20\,s$ (this corresponds to stopping condition $SC2$).

In the following, we use the parameters $TL_{min}^{evac} = 4$ and $TL_{max}^{evac} = 10$ for the evacuation subproblem as well as the parameters $TL_{min}^{route} = 2$ and $TL_{max}^{route} = 7$ for the routing subproblem in order to determine the tabu tenure for the tabu list TL_{mode}. While these parameters are used to solve all problem instances, the parameters a, b, α, and β used to calculate the tabu tenures for tabu lists TL_{add} and TL_{drop} for both subproblems have been selected separately for each set of problem instances consisting of 30, 50 and 70 jobs (cf. Table 8.15). All of these parameters have again been selected based on some preliminary tests.

	Evacuation				Routing			
N	a	α	b	β	a	α	b	β
30	2	0.20	2	0	5	0.20	0	0
50	4	0.25	2	0	5	0.20	2	0
70	6	0.20	3	0	6	0.20	2	0

Table 8.15: Settings for parameters a, b, α, and β for both subproblems that have been used to evaluate the problem instances.

Below, the results that have been obtained by this solution approach based on these parameter settings are reported in Table 8.16. Here, we compare these results to the best solution obtained by the solution approach based on priority rules for any of the combinations of a schedule generation scheme and a priority rule (cf. Table 8.14). Additionally, in this table, we display the best results obtained for the relaxed problem that have been found by the tabu search algorithm for the evacuation subproblem. As described in Chapter 7, the usage of building sections is neglected for these solutions, i.e. it is assumed that the patients can be evacuated from their sick rooms to the selected safety zones without blockings. It should be noted, however, that these results do not constitute lower bounds for the problem and better solutions might exist in the solution space.

Instance	Priority Rules	Relaxed	Complete
$h301\_1$	1606	1387	1387*
$h301\_2$	2018*	2018	2018*
$h302\_1$	814*	785	841
$h302\_2$	1087*	1058	1092
$h303\_1$	674*	506	674*
$h303\_2$	566	509	556*
$h501\_1$	3284*	3284	3284*
$h501\_2$	2317	2068	2167*
$h502\_1$	1876*	1695	1911
$h502\_2$	1414	1236	1358*
$h503\_1$	869*	786	968
$h503\_2$	974*	687	1011
$h701\_1$	4413	4040	4378*
$h701\_2$	4208	3671	3849*
$h702\_1$	2568	2223	2490*
$h702\_2$	2858	2494	2660*
$h703\_1$	1389	919	1337*
$h703\_2$	1592	815	1273*

Table 8.16: Results obtained by the solution approach based on resource flows. Here, the best solution for the relaxed problem as well as the best solution for the complete problem are reported. These results are compared to the best solution obtained by the solution approach based on priority rules for any combination of a schedule generation scheme and a priority rule. The best solution that has been found by either of the two solution approaches is marked with an asterisk.

As can be seen, the solution approach based on resource flows has been able to find solutions with either a better or at least an equal makespan for

13 out of 18 problem instances. Additionally, for 4 out of the remaining 5 problem instances, the difference of the makespan of the solution found by the solution approach based on resource flows from the makespan of the best solution found by the solution approach based on priority rules amounts to less than 50 units (i.e. seconds). Also, the best solution found by the solution approach based on priority rules has been obtained as a result of 10 runs with different combinations of schedule generation schemes and priority rules. Thus, while the solution approach based on resource flows only had 2 minutes to find a solution, the solution approach based on priority rules had a total of 20 minutes (i.e. 2 minutes for each of the 10 combinations). For this reason, we now compare the results of the individual runs. Here, in Table 8.17, we report the average deviation (in percent) from the best solutions found by either the solution approach based on priority rules or the solution approach based on resource flows (cf. Table 8.16) for all algorithms that have been tested.

Parallel Schedule Generation Scheme					Priority
SPT	LPT	LST	LFT	Rand	Rules
8.54	8.60	10.63	8.89	7.58	4.35

Serial Schedule Generation Scheme					Resource
SPT	LPT	LST	LFT	Rand	Flow
38.24	46.05	96.77	17.88	45.25	1.16

Table 8.17: The average deviation (in percent) from the best solutions found by either the solution approach based on priority rules or the solution approach based on resource flows for all combinations of schedule generation schemes and priority rules based on the results displayed in Table 8.14. Also, the average deviation (in percent) of the best solutions found by all of these combinations and the average deviation (in percent) of the best solutions found by the solution approach based on resource flows are reported.

These results show that the solution approach based on resource flows outperforms the solution approach based on priority rules. In particular, the average deviation for the best combination of a schedule generation scheme and a priority rule is more than 6 % worse than the average deviation for the solution approach based on resource flows. Indeed, even the average deviation of the best solutions found by all combinations of schedule generation schemes and priority rules is still more than 3 % worse. Also, these results show that our initial observation that the parallel schedule generation scheme is able to obtain better solutions than the serial schedule generation scheme is correct.

Here, even for the best priority rule (i.e. the priority rule LFT), the serial schedule generation scheme obtains an average deviation that is more than 7 % worse than the average deviation for the worst priority rule (i.e. the priority rule LST) for the parallel schedule generation scheme. Finally, while the selected priority rule has a large influence on the quality of the solutions obtained by the serial schedule generation scheme, the same is not true for the solutions obtained by the parallel schedule generation scheme. Instead, with the exception of the priority rule LST, the difference between the average deviations obtained for the various priority rules is generally less than or around 1 %.

It should be noted that despite all this we can not compare the overall performance of the algorithms due to a lack of optimal solutions or even lower bounds for the problems. Here, while we have modeled the problem of hospital evacuations as a mixed-integer linear program in Section 4.1.2, it is highly unlikely that the problem instances can be solved (or even a feasible solution can be found) by a mixed-integer linear program solver such as CPLEX. This is due to the fact that, apart from the 30 to 70 jobs that have to be scheduled, each job consists of multiple operations for each evacuation route. For the larger problem instances, this can amount to several thousand operations. At the same time, only a portion of these operations actually have to be scheduled depending on the selected route modes of the jobs.

Next, we have solved the problem instances again. Unlike before, however, patients can now also be evacuated inside their hospital beds. Thus, three equipment modes are available for patients that can be evacuated while sitting and two equipment modes are available for patients that can only be evacuated while lying. Here, the best solution obtained by the solution approach based on priority rules for any combination of a schedule generation scheme and a priority rule as well as the solution obtained by the solution approach based on resource flows are reported in Table 8.18. Furthermore, the average deviations (in percent) from the best solutions found by either of these two solution approaches for all algorithms are reported in Table 8.19. It should be noted that the solutions for the problem instances if no hospital beds can be used for the evacuation of the patients are always contained in the solution space for the instances if hospital beds can be used. This

Instance	Priority Rules	Relaxed	Complete
h301_1	1249	1186	1183*
h301_2	1241	1180	1194*
h302_1	749*	719	760
h302_2	960	916	953*
h303_1	654*	466	654*
h303_2	534*	482	556
h501_1	1866	1639	1745*
h501_2	2274	1687	1830*
h502_1	1756	1274	1493*
h502_2	1205	1014	1192*
h503_1	772*	687	804
h503_2	954*	523	1057
h701_1	4631	2779	3014*
h701_2	4274	3221	3336*
h702_1	2505	1735	1957*
h702_2	2736	1942	2124*
h703_1	1418	712	1150*
h703_2	1603	734	1225*

Table 8.18: Results obtained by the solution approach based on resource flows if beds can be used for the evacuation of the patients. These results are compared to the best result obtained by the solution approach based on priority rules for any combination of a schedule generation scheme and a priority rule.

Parallel Schedule Generation Scheme					Priority
SPT	LPT	LST	LFT	Rand	Rules
18.15	16.76	17.59	18.99	15.67	13.99

Serial Schedule Generation Scheme					Resource
SPT	LPT	LST	LFT	Rand	Flow
53.56	56.12	108.50	28.35	71.16	1.14

Table 8.19: The average deviation (in percent) from the best solutions found by either the solution approach based on priority rules or the solution approach based on resource flows for all algorithms that have been tested. These results are for the case that hospital beds can be used for the evacuation of the patients.

is due to the fact that no other parameters of the problem instances have been changed and these solutions can still be generated if the corresponding modes are selected.

Here, even more than for the first run, these results show that the solution approach based on resource flows is able to obtain better results than the solution approach based on priority rules. In particular, the average deviation for the solution approach based on resource flows is 13 % better than the average deviation of the best solutions obtained by any combination of a schedule generation scheme and a priority rule. This is likely due to the fact that the additional modes for the jobs have a larger influence on the solution approach based on priority rules because the schedule generation schemes always have to schedule all operations corresponding to the selected route modes. On the other hand, for the solution approach based on resource modes, the jobs themselves are taken into account in the evacuation problem such that evaluating a single modification requires less time.

Finally, we consider some observations concerning the evacuation of hospitals that can be made from these results. Here, first of all, it can be seen that allowing hospital beds for the evacuation of the patients has the largest influence on the estimated time required for the evacuation for problem instances in which fewer assistants and aids are available in relation to the patients that have to be evacuated. On the one hand, this is because two instead of three assistants are sufficient to evacuate a patient in a hospital bed instead of on a stretcher (which before had to be used in order to evacuate patients that have to be evacuated while lying). On the other hand, it is possible that the amount of additional resource transfers can be reduced if assistants and aids are not in the same location (i.e. because hospital beds are available in the sick rooms of the patients).

Next, the difference between the best solution for the relaxed problem and the best solution for the complete problem is largest for problem instances in which a lot of assistants and aids are available (cf. Tables 8.16 and 8.18). This is due to the fact that blockings are more likely to occur in this case because more patients can be evacuated at the same time. Here, a closer look at the solutions reveals that the majority of these blockings occur in some key locations of the hospital. In particular, the elevators can be identified as bottlenecks. Here, for a more realistic scenario, it might be feasible to identify these bottleneck resources beforehand and model them with additional care while neglecting other building sections. For example, multiple smaller corridor sections that are less critical could be combined to one larger corridor section. As a side effect, this would also reduce the number of operations that have to be scheduled.

Apart from this, various other tests can be performed for the problem of hospital evacuations. For example, each aid can be assigned to a fixed team of assistants (e.g. if two wheelchairs are available, two teams consisting of one wheelchair as well as one assistant could be formed). On the one hand, this would render second-tier transfers for this problem unnecessary. On the other hand, this would reduce the solution space and might exclude better solutions (e.g. if only a limited number of patients have to be evacuated in a wheelchair, the assistants that are assigned to the corresponding teams can not be used in order to evacuate other patients that require different or no aids). Also, various other parameters that have been used for these tests can be adapted. For example, different aids could be considered for the evacuation of the patients or different times could be associated with the usage of stairs and elevators.

Here, we will not perform any of these additional tests for the following reasons. First of all, many of these parameters and variations depend on the actual hospital that has to be evacuated as well as further requirements that might have to be taken into account. Apart from a lack of such specific data, it is also unlikely that any new insights can be gained regarding the performance of the two solution approaches. The latter is also due to the problem that the results obtained by the algorithms can not be compared to any other results. In particular, no optimal solutions or lower bounds for these problem instances exist.

9 Conclusions

The problem of hospital evacuations that has been tackled in this thesis poses a greater difficulty than typical evacuation problems considered in literature. In particular, patients that have to be evacuated can not always help themselves but rely on the help of assistants and aids. At the same time, this problem has received very little attention in literature despite its relevance. For example, Sternberg et al. (2004) and Lipp et al. (1998) remark that many hospitals are insufficiently prepared for an actual evacuation. Due to this, we have tackled the problem of hospital evacuations in order to estimate the time required to evacuate all patients from their sick rooms inside the hospital to safety zones inside or outside the hospital. Apart from this, because of its significance to the model that has been presented for the problem of hospital evacuations, we have especially dealt with the RCPSP with resource transfers and presented a solution approach based on resource flows for this problem. In the following, we will summarize the key findings of this thesis. Furthermore, we will discuss some limitations of the models and solution approaches and give an outlook on possible research that can be conducted based on these findings.

First of all, we considered the problem of hospital evacuations. We have modeled this problem as a multi-mode RCPSP with resource transfers and blockings. Additionally, we have presented two solution approaches for the problem. Here, the first solution approach uses schedule generation schemes as well as priority rules in order to generate schedules. For this solution approach, we have shown that an optimal solution may not be contained in the solution space considered by the schedule generation schemes due to the priority rules used to select resource transfers to an operation. In particular, we have proved that the problem of selecting first- and second-tier resource transfers to an operation is NP-hard (cf. Section 4.2.3).

Due to this drawback, we have then integrated first- and second-tier resource transfers as well as blockings into a solution representation based on resource flows that is used for the second solution approach. This approach is a two-stage algorithm for the problem such that, in the first stage, the jobs are scheduled with respect to the available assistants and aids while, in the second stage, the operations are scheduled with respect to the available space in the building sections based on the results from the first stage. For both of these solution approaches we have reported computational results. These results have shown that the solution approach based on resource flows outperforms the solution approach based on priority rules. At the same time, however, we have been unable to evaluate the overall performance of the algorithms due to a lack of optimal solutions or lower bounds.

Apart from this, further limitations of both the model as well as the solution approaches that have been presented for the problem of hospital evacuations exist. For instance, the model primarily focuses on the constraints that are given by the available assistants and aids as well as the infrastructure of the hospital. Possible other requirements that may depend on the actual hospital that has to be evacuated are not taken into account. Additionally, it is assumed that the situation within the hospital is static and predictable. In reality, however, many unpredictable situations or stochastic parameters have to be taken into account, e.g. if part of a corridor is blocked by a hospital bed or based on the physical condition, preparedness, and exhaustion of the assistants. All of these limitations contribute to the fact that our solution approaches can only give a rough estimate on how much time is required for an actual evacuation. This problem can further be aggravated by the actual parameters used for the problem instances, e.g. depending on how accurately the infrastructure as well as the parameters related to the different resources could be modeled.

Due to these limitations, various opportunities for further research related to the problem of hospital evacuations exist. For example, it would be interesting to model and evaluate the evacuation of a real hospital and compare the results to values obtained in actual evacuations or evacuation exercises. Also, in order to better be able to evaluate the performance of the algorithms, lower bounds for the problem of hospital evacuations should be developed. Apart from this, two extensions of the model as well as of the solution approach are of particular interest. On the one hand, for evacuations where patients have to be transported to sheltering facilities, an integrated approach can be developed that uses the results obtained by our algorithms in order to better schedule the further transport of the patients from the staging areas to these sheltering facilities. On the other hand, because the evacuation of a hospital is seldom static, stochastic elements can be taken into account and a simulation can be developed. This simulation could be used, for example, to test and improve the reliability of a solution that has been calculated by our algorithms. Also, this latter approach might be of interest in order to evaluate more dynamic situations such as a fire or a hazardous materials spill inside the hospital.

Next, we tackled the RCPSP with first- and second-tier resource transfers. As described in Chapter 5, higher-tier resource transfers have only been considered by Krüger (2009) as well as by Krüger and Scholl (2010). To the best of our knowledge, no solution approach for this problem has previously been presented in literature. Thus, due to the importance of resource transfers to the model for the problem of hospital evacuations, we have dealt with this problem in detail in this thesis.

For this, we have first introduced a model for this problem and compared it to the model presented by Krüger (2009). This resulted in the observation that, in the model by Krüger (2009), the start of an activity might be delayed because an activity can only be completed when all resource units that are transferred to the end of the activity have arrived. Due to this, better solutions might be excluded from the solution space in this model whereas this problem does not occur in our model (cf. Section 5.3). At the same time, a limitation of both models is that resource units that have been transferred to the start of an activity by pure second-tier transfers have to remain at the activity until the start of the activity. This, in turn, might also lead to the exclusion of better solutions from the solution space. For this reason, it might be interesting to adapt the model such that pure second-tier resource units do not have to wait at the activity to which they support the transfer of a first-tier resource until the starting time of this activity.

Afterward, we have introduced a solution approach for the RCPSP with first- and second-tier resource transfers. For this, we have extended the solution representation based on resource flows to this problem. Additionally, we have shown that a feasible resource flow exists for every project that represents an optimal (earliest start) schedule. Based on this solution representation, we have then defined modifications to redirect resource transfers. Here, while it remains an open question whether any of the resulting neighborhoods for the RCPSP with first- and second-tier resource transfers is connected or opt-connected, we have been able to show that at least the neighborhood $\mathcal{N}_1 = \mathcal{N}_{\text{reroute}} \cup \mathcal{N}_{\text{reverse}}$ is connected for both, the classical RCPSP as well as the RCPSP with first-tier resource transfers.

In order to solve the RCPSP with first- and second-tier resource transfers, we have implemented a tabu search algorithm that represents solutions as resource flows. The results that have been obtained by this algorithm for problem instances for the classical RCPSP as well as for the RCPSP with first-tier resource transfers are generally worse than those obtained by other algorithms. In part, this is due to the fact that we did not especially develop the algorithm in order to solve these problems. On the other hand, for the RCPSP with first- and second-tier resource transfers, our algorithm has been shown to reliably obtain results even for larger problem instances. As for the problem of hospital evacuations, however, we have been unable to evaluate the actual performance of the algorithm due to a lack of lower bounds or already existing computational results. A general problem related to this solution approach is that we have been unable as of yet to better guide the search for good solutions.

Regarding this solution approach, the following opportunities for further research might be of interest. First of all, various open questions regarding the connectivity of the different neighborhoods remain. Next, it might be possible to further reduce the size of the existing neighborhoods as well as to use them in different local search algorithms. Also, criteria should be found that can guide the search to regions of the solution space that contain good solutions. Related to these possibilities for further research, the solution approach could also be adapted to the more special situation encountered in either the classical RCPSP or the RCPSP with first-tier resource transfers.

Finally, we considered the blocking constraint in the context of the RCPSP. Similar to the extension of resource transfers, this problem has received little attention until now. Likewise, in this thesis we only considered this constraint to a lesser content because the blocking constraint has a smaller influence on the problem of hospital evacuations considered in this thesis than resource transfers. Also, it can be assumed that the blocking constraint is primarily of interest for some specialized applications whereas resource transfers are more likely to occur in other applications. We have been able, however, to incorporate the blocking constraint into the solution representation based on resource flows as well as into the modifications based on this solution representation. Here, it might be interesting to take a closer look at the RCPSP with blockings and to either build on the initial results reported in this thesis or to develop a more specific approach to deal with blockings. In the former case, for example, the connectivity of the neighborhoods that have been presented for this problem should be considered. Also, it might be possible to further reduce the size of the neighborhoods.

A Notation

Below, some general notation as well as the notation and symbols used for the classical RCPSP, the RCPSP with first- and second-tier resource transfers, and the problem of hospital evacuations is summarized. Further notation used only locally in the different chapters is not listed here.

A.1 General Notation

RCPSP	Resource-constrained project scheduling problem
HEP	Hospital evacuation problem
MILP	Mixed-integer linear program
C_{max}	Makespan of a feasible schedule
LB	Lower bound
UB	Upper bound

A.2 Classical RCPSP

n	Number of (real) activities
r	Number of renewable resources
V	Set of activities $i = 1, \dots, n$
V_{all}	Set of activities $i = 0, 1, \dots, n, n+1$
V_0	Set of activities $i = 0, 1, \dots, n$
V_*	Set of activities $i = 1, \dots, n, n+1$
$\mathcal{R}$	Set of renewable resources $k = 1, \dots, r$
A	Set of precedence constraints $i \rightarrow j$ between activities $i \in V_{\text{all}}$ and $j \in V_{\text{all}}$
R_k	Capacity of resource $k \in \mathcal{R}$
p_i	Processing time of activity $i \in V_{\text{all}}$

r_{ik}	Resource requirements of activity $i \in V_{\text{all}}$ for resource $k \in \mathcal{R}$
S_i	Starting time of activity $i \in V_{\text{all}}$ in a feasible schedule
C_i	Completion time of activity $i \in V_{\text{all}}$ in a feasible schedule

A.3 RCPSP with First- and Second-Tier Resource Transfers

Additional to the notation for the classical RCPSP, the following notation is used for the RCPSP with first- and second-tier resource transfers.

$\mathcal{R}^{\text{sa}}$	Set of resources that can be transferred by stand-alone transfer
$\mathcal{R}^{\text{ru}}$	Set of resources that have to be transferred by resource-using transfer
Δ_{ijk}	Transfer time for resource $k \in \mathcal{R}$ from activity $i \in V_0$ to activity $j \in V_*$
μ_{kl}	Amount of resource units of resource $k \in \mathcal{R}^{\text{sa}}$ required to support the transfer of one unit of resource $l \in \mathcal{R}^{\text{ru}}$

A.4 Problem of Hospital Evacuations

N	Number of jobs
r	Number of all renewable resources
J	Set of jobs $j = 0, 1, \ldots, N, N+1$
$\mathcal{R}$	Set of all renewable resources $k = 1, \ldots, r$
$\mathcal{R}^{\text{asst}}$	Set of all resources representing assistants
$\mathcal{R}^{\text{aid}}$	Set of all resources representing aids
$\mathcal{R}^{\text{trf}}$	Set of all resources associated with a transfer time
$\mathcal{R}^{\text{sect}}$	Set of all resources representing building sections
R_k	Capacity of resource $k \in \mathcal{R}$
m_{j1}	Number of equipment modes for job $j \in J$
m_{j2}	Number of route modes for job $j \in J$
$\mathcal{M}_{j1}$	Set of equipment modes $m_1 = 1, \ldots, m_{j1}$ for job $j \in J$
$\mathcal{M}_{j2}$	Set of route modes $m_2 = 1, \ldots, m_{j2}$ for job $j \in J$
m_j	Number of mode combinations $m = (m_1, m_2)$ with $m_1 \in \mathcal{M}_{j1}$ and $m_2 \in \mathcal{M}_{j2}$ for job $j \in J$

$\mathcal{M}_j$	Set of mode combinations $m = 1, \dots, m_j$ for job $j \in J$
n	Number of operations
n_j	Number of operations associated with job $j \in J$
V	Set of operations $u = 1, \dots, n$
V_{all}	Set of operations $u = 0, 1, \dots, n, n+1$
V_0	Set of operations $u = 0, 1, \dots, n$
V_*	Set of operations $u = 1, \dots, n, n+1$
V_{in}	Set containing the first operation of each evacuation route of jobs $j = 1, \dots, N$
V_{out}	Set containing the last operation of each evacuation route of jobs $j = 1, \dots, N$
V'_{in}	Set containing the first operation of each evacuation route of jobs $j = 1, \dots, N$ as well as dummy operation $n+1$
V'_{out}	Set containing the last operation of each evacuation route of jobs $j = 1, \dots, N$ as well as dummy operation 0
$V_j(m_2)$	Set of operations representing the evacuation route denoted by route mode $m_2 \in \mathcal{M}_{j2}$ of job $j \in J$
$\sigma(u)$	Job $j \in J$ to which operation $u \in V_{\text{all}}$ belongs
$\mathcal{B}$	Set of precedence constraints $u \to v$ between blocking operations $u \in V_{\text{all}}$ and operations $v \in V_{\text{all}}$
$\mathcal{NB}$	Set of precedence constraints $u \to v$ between non-blocking operations $u \in V_{\text{all}}$ and operations $v \in V_{\text{all}}$
p_{um}	Processing time of operation $u \in V_{\text{all}}$ if it is executed in mode combination $m \in \mathcal{M}_{\sigma(u)}$
r_{umk}	Resource requirements of operation $u \in V_{\text{all}}$ for resource $k \in \mathcal{R}$ if it is executed in mode combination $m \in \mathcal{M}_{\sigma(u)}$
Δ_{uvk}	Transfer time for resource $k \in \mathcal{R}^{\text{trf}}$ from operation $u \in V_{\text{all}}$ to operation $v \in V_{\text{all}}$
S_u	Starting time of operation $u \in V_{\text{all}}$ in a feasible schedule
C_u	Completion time of operation $u \in V_{\text{all}}$ in a feasible schedule

Acknowledgements

This thesis is the result of the PhD studies that I have been engaged in during my time as a research associate at the Institute of Applied Stochastics and Operations Research at the Clausthal University of Technology. In the following, I would like to thank those people that have helped in various ways during this time.

First and foremost, my thanks go to my doctoral adviser Prof. Dr. Sigrid Knust who has not only offered me the opportunity to write this thesis but has also supported and encouraged my work during this time. For this, her readiness to discuss the various questions that have cropped up along the way, as well as her invaluable advice, I am, and remain, very thankful. Furthermore, I would like to thank Prof. Dr. Michael Kolonko for his support during my time as a research associate in Clausthal as well as Prof. Dr. Jürgen Zimmermann for accepting to review this thesis.

For bringing the problem of hospital evacuations to my attention as well as for the invitation to attend an evacuation exercise in a hospital, I would like to thank Mr. Stefan Kruse of Asklepios Kliniken GmbH. Also, my thanks go to Mr. Heiko Schnitker of the fire brigade of the city of Osnabrück for answering my questions related to the evacuation of hospitals as well as to Mr. Peter Broszeit of Asklepios Harzkliniken GmbH for providing me with floor plans of the Asklepios Harzklinik Goslar.

Then, for their feedback and advice, I would like to thank my friends and colleagues both in Clausthal as well as in Osnabrück. In particular, my thanks go to Stefan Waldherr for the discussions and help related to a proof in this thesis.

I am very grateful for the support and encouragement of my family not only during the creation of this thesis.

Last but not least, I would like to thank my boyfriend Yiyang Su for his understanding and support during this time as well as for reminding me that work is not everything.

Bibliography

R. K. Ahuja, J. B. Orlin, and R. E. Tarjan. Improved time bounds for the maximum flow problem. *SIAM Journal on Computing*, 18(5):939–954, 1989.

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1993.

A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032, 2008.

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

C. Artigues, S. Demassey, and E. Néron. *Resource-Constrained Project Scheduling.* Control systems, robotics and manufacturing series. John Wiley & Sons, New York, USA, 2010.

T. Baar, P. Brucker, and S. Knust. Tabu search algorithms and lower bounds for the resource-constrained project scheduling problem. In S. Voss, S. Martello, I. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 3–13. Kluwer Academic Publishers, Boston, USA, 1998.

P. Baptiste and S. Demassey. Tight LP bounds for resource constrained project scheduling. *OR Spectrum*, 26(2):251–262, 2004.

M. Bartusch, R. Möhring, and F. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16(1):199–240, 1988.

A. Beaudry, G. Laporte, T. Melo, and S. Nickel. Dynamic transportation of patients in hospitals. *OR Spectrum*, 32:77–107, 2010.

D. D. Bedworth and J. E. Bailey. *Integrated Production Control Systems: Management, Analysis, Design.* John Wiley & Sons, Inc., New York, USA, 1982.

T. Berthold, S. Heinz, M. Lübbecke, R. Möhring, and J. Schulz. A constraint integer programming approach for resource-constrained project scheduling. In A. Lodi, M. Milano, and P. Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 6140 of *Lecture Notes in Computer Science*, pages 313–317. Springer, Berlin, Heidelberg, Germany, 2010.

D. R. Bish, E. Agca, and R. Glick. Decision support for hospital evacuation and emergency response. *Annals of Operations Research*, 2011. to appear.

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

F. F. Boctor. Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research*, 49(1):3–13, 1990.

J. Böttcher, A. Drexl, R. Kolisch, and F. Salewski. Project scheduling under partially renewable resource constraints. *Management Science*, 45(4):543–559, 1999.

K. Bouleimen and H. Lecocq. 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):268–281, 2003.

E. H. Bowman. The schedule-sequencing problem. *Operations Research*, 7 (5):621–624, 1959.

S. Bretschneider. *Mathematical Models for Evacuation Planning in Urban Areas.* Springer, Berlin, Heidelberg, Germany, 2013.

P. Brucker and S. Knust. A linear programming and constraint propagation-based lower bound for the rcpsp. *European Journal of Operational Research*, 127(2):355–362, 2000.

P. Brucker and S. Knust. *Complex Scheduling.* Springer, Berlin, Heidelberg, Germany, 2006.

P. Brucker, S. Knust, A. Schoo, and O. Thiele. A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 107(2):272–288, 1998.

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

R. E. Burkard, K. Dlaska, and B. Klinz. The quickest flow problem. *Mathematical Methods of Operations Research*, 37(1):31–58, 1993.

California Emergency Medical Services Authority. Hospital incident command system. URL http://www.emsa.ca.gov/hics/. Visited on 2012-09-11.

L. G. Chalmet, R. L. Francis, and P. B. Saunders. Network models for building evacuation. *Fire Technology*, 18(1):90–113, 1982.

L. Chen and E. Miller-Hooks. The building evacuation problem with shared information. *Naval Research Logistics*, 55(4):363–376, 2008.

A. K. Childers and K. M. Taaffe. Healthcare facility evacuations: lessons learned, research activity, and the need for engineering contributions. *Journal of Healthcare Engineering*, 1(1):125–140, 2010.

W. Choi, H. W. Hamacher, and S. Tufekci. Modeling of building evacuation problems by network flows with side constraints. *European Journal of Operational Research*, 35(1):98–110, 1988.

N. Christofides, R. Alvarez-Valdes, and J. M. Tamarit. Project scheduling with resource constraints: a branch and bound approach. *European Journal of Operational Research*, 29(3):262–273, 1987.

R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison-Wesley, Reading, USA, 1967.

E. W. Davis. Project scheduling under resource constraints - historical review and categorization of procedures. *AIIE Transactions*, 5(4):297–313, 1973.

D. Debels, B. De Reyck, R. Leus, and M. Vanhoucke. A hybrid scatter search / electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*, 169(2):638–653, 2006.

S. Demassey, C. Artigues, and P. Michelon. Constraint-propagation-based cutting planes: an application to the resource-constrained project scheduling problem. *INFORMS Journal on Computing*, 17(1):52–65, 2005.

E. L. Demeulemeester and W. Herroelen. *Project Scheduling: A Research Handbook*. International Series in Operations Research & Management Science. Kluwer Academic Publishers, Boston, USA, 2002.

E. L. Demeulemeester and W. S. Herroelen. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38(12):1803–1818, 1992.

E. L. Demeulemeester and W. S. Herroelen. New benchmark results for the resource-constrained project scheduling problem. *Management Science*, 43(11):1485–1492, 1997.

J. Du, J. Y.-T. Leung, and C. S. Wong. Minimizing the number of late jobs with release time constraint. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 11:97–107, 1992.

S. E. Elmaghraby. *Activity Networks: Project Planning and Control by Network Models*. John Wiley & Sons, New York, USA, 1977.

R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5 (6):345, 1962.

J. W. Fondahl. A non-computer approach to the critical path method for the construction industry. Technical report, Department of Civil Engineering, Stanford University, 1961.

L. R. Ford, Jr. and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6(3):419–433, 1958.

P. Fortemps and M. Hapke. On the disjunctive graph for project scheduling. *Foundations of Computing and Decision Sciences*, 22(3):195–209, 1997.

D. Gale. Transient flows in networks. *Michigan Mathematical Journal*, 6(1): 59–63, 1959.

H. L. Gantt. A graphical daily balance in manufacture. In *Transactions of the American Society of Mechanical Engineers*, volume XXIV, pages 1322–1336. 1903.

H. L. Gantt. *Work, Wages, and Profits*. The Engineering magazine, New York, USA, 1910.

M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976.

F. Glover. Tabu search - part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

F. Glover. Tabu search - part II. *ORSA Journal on Computing*, 2:4–32, 1990.

F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

D. Golmohammadi and D. Shimshak. Estimation of the evacuation time in an emergency situation in hospitals. *Computers & Industrial Engineering*, 61(4):1256–1267, 2011.

B. H. Gray and K. Hebert. Hospitals in hurricane katrina: challenges facing custodial institutions in a disaster. *Journal of Health Care for the Poor and Underserved*, 18(2):283–298, 2007.

P. Gretenkort and H. Harke. Ärztliche Leitungsfunktion bei einer innerklinischen Gefahrenlage. *Anästhesiologie & Intensivmedizin*, 42(3):170–175, 2001.

N. G. Hall and C. Sriskandarajah. A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, 44(3):510–525, 1996.

H. W. Hamacher and S. A. Tjandra. Mathematical modelling of evacuation problems: a state of the art. In M. Schreckenberg and S. D. Sharma, editors, *Pedestrian and Evacuation Dynamics*, pages 227–266. Springer, Berlin, Heidelberg, Germany, 2002.

T Hanne, T. Melo, and S. Nickel. Bringing robustness to patient flow management through optimized patient transports in hospitals. *Interfaces*, 39 (3):241–255, 2009.

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

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

S. Hartmann and D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010.

S. Hartmann and A. Drexl. Project scheduling with multiple modes: a comparison of exact algorithms. *Networks*, 32(4):283–297, 1998.

S. Hartmann and R. Kolisch. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127(2):394–407, 2000.

W. Herroelen, B. De Reyck, and E. Demeulemeester. Resource-constrained project scheduling: A survey of recent developments. *Computers & Operations Research*, 25(4):279–302, 1998.

B. Hoppe and É. Tardos. Polynomial time algorithms for some evacuation problems. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 433–441, Arlington, USA, 1994.

B. Hoppe and É. Tardos. The quickest transshipment problem. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 512–521, San Francisco, USA, 1995.

IBM Corporation. IBM CPLEX Optimizer. URL `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`. Visited on 2013-08-16.

L. A. Kaplan. *Resource-Constrained Project Scheduling with Preemption of Jobs*. PhD thesis, University of Michigan, Michigan, USA, 1988.

L. A. Kaplan. Resource-constrained project scheduling with setup times. Unpublished Paper, Department of Management Science, University of Tennessee, Knoxville, USA, 1991.

I. Katter, O. Kunitz, and A. Deller. Tagebuch einer Krankenhausevakuierung. *Der Anaesthesist*, 57(7):693–703, 2008.

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

J. E. Kelley, Jr. and M. R. Walker. Critical-path planning and scheduling. In *Proceedings of the Eastern Joint Computer Conference*, pages 160–173, New York, USA, 1959.

A. Kirchner and A. Schadschneider. Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. *Physica A: Statistical Mechanics and its Applications*, 312(1–2):260–276, 2002.

R. Klein. Bidirectional planning: improving priority rule-based heuristics for scheduling resource-constrained projects. *European Journal of Operational Research*, 127(3):619– 638, 2000.

R. Klein and A. Scholl. Computing lower bounds by destructive improvement: an application to resource-constrained project scheduling. *European Journal of Operational Research*, 112(2):322–346, 1999.

T. Koch. *Rapid Mathematical Programming*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 2004.

Y. A. Kochetov and A. A. Stolyar. Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In *Proceedings of the 3rd International Workshop of Computer Science and Information Technologies*, Ufa, Russia, 2003.

E. Köhler, K. Langkau, and M. Skutella. Time-expanded graphs for flow-dependent transit times. In R. Möhring and R. Raman, editors, *Algorithms - ESA 2002*, volume 2461 of *Lecture Notes in Computer Science*, pages 49–561. Springer, Berlin, Heidelberg, Germany, 2002.

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

R. Kolisch. Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, 14(3):179–192, 1996b.

R. Kolisch and A. Drexl. Adaptive search for solving hard project scheduling problems. *Naval Research Logistics*, 43(1):23–40, 1996.

R. Kolisch and A. Drexl. Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 29(11):987–999, 1997.

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

R. Kolisch and R. Padman. An integrated survey of deterministic project scheduling. *Omega*, 29(3):249–272, 2001.

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

R. Kolisch, A. Sprecher, and A. Drexl. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, 41(10):1693–1703, 1995.

R. Kolisch, C. Schwindt, and A. Sprecher. Benchmark instances for project scheduling problems. In J. Węglarz, editor, *Project Scheduling*, volume 14 of *International Series in Operations Research & Management Science*, pages 197–212. Springer, New York, USA, 1999.

Konrad-Zuse-Zentrum für Informationstechnik Berlin. Zimpl. URL `http://zimpl.zib.de/`. Visited on 2013-08-16.

B. Kotnyek. An annotated overview of dynamic network flows. Technical Report 4936, Institut national de recherche en informatique et en automatique (INRIA), 2003.

N. O. Kröger. Optimierte Kran- und Förderanlagenplanung im Megahub. Master's thesis, Universität Osnabrück, Osnabrück, Germany, 2013.

D. Krüger. *Multi-Project Scheduling with Resource Transfers*. Books on Demand GmbH, Norderstedt, Germany, 2009.

D. Krüger and A. Scholl. A heuristic solution framework for the resource constrained (multi-) project scheduling problem with sequence-dependent transfer times. *European Journal of Operational Research*, 197(2):492–508, 2009.

D. Krüger and A. Scholl. Managing and modelling general resource transfers in (multi-) project scheduling. *OR Spectrum*, 32(2):369–394, 2010.

K. Y. Li and R. J. Willis. An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research*, 56(3):370–379, 1992.

M. Lipp, H. Paschen, M. Daubländer, R. Bickel-Petrup, A. Thierbach, R. Müller, and W. Dick. Planung deutscher Krankenhäuser für Großschadensfälle. *Notfall + Rettungsmedizin*, 1(4):208–213, 1998.

A. Lova, P. Tormos, M. Cervantes, and F. Barber. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, 117(2): 302–316, 2009.

G. G. Løvås. Models of wayfinding in emergency evacuations. *European Journal of Operational Research*, 105(3):371–389, 1998.

D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar. Application of a technique for research and development program evaluation. *Operations Research*, 7(5):646–669, 1959.

A. Mascis and D. Pacciarelli. Machine scheduling via alternative graphs. Technical Report RT-DIA-46-2000, Dipartimento di Informatica e Automazione, Università Roma Tre, Rome, Italy, 2000.

A. Mascis and D. Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517, 2002.

M. Mika, G. Waligóra, and J. Węglarz. Modelling setup times in project scheduling. In J. Józefowska and J. Węglarz, editors, *Perspectives in Modern Project Scheduling*, volume 92, pages 131–163. Springer, New York, USA, 2006.

M. Mika, G. Waligóra, and J. Węglarz. Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. *European Journal of Operational Research*, 187(3):1238–1250, 2008.

A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science*, 44(5):714–729, 1998.

G. Müller. *Kriterien für Evakuierungsempfehlungen bei Chemikalienfreisetzungen*, volume 32 of *Zivilschutz-Forschung*. Bundesamt für Zivilschutz, Bonn, Germany, 1998.

K. Neumann, C. Schwindt, and J. Zimmermann. *Project Scheduling with Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions*. Springer, Berlin, Heidelberg, Germany, 2003.

K. Neumann, C. Schwindt, and J. Zimmermann. Resource-constrained project scheduling with time windows. In J. Józefowska and J. Węglarz, editors, *Perspectives in Modern Project Scheduling*, volume 92, pages 375–407. Springer, New York, USA, 2006.

K. Nonobe and T. Ibaraki. Formulation and tabu search algorithm for the resource constrained project scheduling problem. In *Essays and Surveys in Metaheuristics*, volume 15 of *Operations Research / Computer Science Interfaces Series*, pages 557–588. Springer, New York, USA, 2002.

E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):797–813, 1996.

X. Pan, C. Han, K. Dauber, and K. Law. A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations. *AI & Society*, 22(2):113–132, 2007.

F. S. Pappert, E. Angelidis, and O. Rose. Framework for simulation based scheduling of assembly lines. In *Proceedings of the 2010 Winter Simulation Conference*, pages 1690–1698, Baltimore, USA, 2010.

J. H. Patterson. A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. *Management Science*, 30(7):854–867, 1984.

J. H. Patterson, R. Słowiński, F. B. Talbot, and J. Węglarz. An algorithm for a general class of precedence and resource constrained scheduling problems. In R. Słowiński and J. Węglarz, editors, *Advances in Project Scheduling*, pages 3–28. Elsevier, Amsterdam, Netherlands, 1989.

A. A. B. Pritsker, L. J. Waiters, and P. M. Wolfe. Multiproject scheduling with limited resources: a zero-one programming approach. *Management Science*, 16(1):93–108, 1969.

A. Quilliot and H. Toussaint. Flow models for project scheduling with transfer delays. In *Proceedings of the Federated Conference on Computer Science and Information Systems*, pages 439–446, Wrocław, Poland, 2012.

B. Roy. Transitivité et connexité. *Comptes Rendus de l'Académie des Sciences*, 249:216–218, 1959.

B. Roy and B. Sussmann. Les problèmes d'ordonnancement avec contraintes disjonctives. Technical Report Note D.S. no. 9 bis, SEMA, Paris, France, 1964.

F. Salewski, A. Schirmer, and A. Drexl. Project scheduling under resource and mode identity constraints: model, complexity, methods, and application. *European Journal of Operational Research*, 102(1):88–110, 1997.

A. Schutt, T. Feydy, P. Stuckey, and M. Wallace. Why cumulative decomposition is not as bad as it sounds. In I. Gent, editor, *Principles and Practice of Constraint Programming - CP 2009*, volume 5732 of *Lecture Notes in Computer Science*, pages 746–761. Springer, Berlin, Heidelberg, Germany, 2009.

R. Słowiński. Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. *European Journal of Operational Research*, 7(3):265–273, 1981.

A. Sprecher and A. Drexl. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 107(2):431–450, 1998.

A Sprecher, R. Kolisch, and A. Drexl. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 80(1):94–102, 1995.

A. Sprecher, S. Hartmann, and A. Drexl. An exact algorithm for project scheduling with multiple modes. *OR Spectrum*, 19(3):195–203, 1997.

E. Sternberg, G. C. Lee, and D. Huard. Counting crises: Us hospital evacuations, 1971-1999. *Prehospital and Disaster Medicine*, 19(2):150–157, 2004.

J. P. Stinson, E. W. Davis, and B. M. Khumawala. Multiple resource-constrained scheduling using branch and bound. *AIIE Transactions*, 10 (3):252–259, 1978.

K. M. Taaffe, R. Kohl, and D. L. Kimbler. Hospital evacuation: issues and complexities. In *Proceedings of the 2005 Winter Simulation Conference*, pages 943–950, Orlando, USA, 2005.

K. M. Taaffe, M. Johnson, and D. Steinmann. Improving hospital evacuation planning using simulation. In *Proceedings of the 2006 Winter Simulation Conference*, pages 509–515, Monterey, USA, 2006.

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

E. Tayfur and K. M. Taaffe. Allocation of resources for hospital evacuations via simulation. In *Proceedings of the 2007 Winter Simulation Conference*, pages 1148–1154, Washington, USA, 2007.

E. Tayfur and K. M. Taaffe. A model for allocating resources during hospital evacuations. *Computers & Industrial Engineering*, 57(4):1313–1323, 2009.

S. A. Tjandra. *Dynamic Network Optimization with Application to the Evacuation Problem.* PhD thesis, Technische Universität Kaiserslautern, Kaiserslautern, Germany, 2003.

P. Tormos and A. Lova. Integrating heuristics for resource-constrained project scheduling: one step forward. Technical report, Department of Statistics and Operations Research, Universidad Politécnica de Valencia, 2003.

B. Urban, U. Kreimeier, S. Prückner, K. G. Kanz, and C. K. Lackner. Krankenhaus-Alarm- und Einsatzpläne für externe Schadenslagen an einem Großklinikum. *Notfall + Rettungsmedizin*, 9(3):296–303, 2006.

A. Valls, F. Ballestín, and S. Quintanilla. Justification and rcpsp: a technique that pays. *European Journal of Operational Research*, 165(2):375–386, 2005.

A. Valls, F. Ballestín, and S. Quintanilla. A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 185(2):495–508, 2008.

P. J. M. van Laarhoven, E. H. L. Aarts, and J. K. Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1):113–125, 1992.

M. Vanhoucke. Setup times and fast tracking in resource-constrained project scheduling. *Computers & Industrial Engineering*, 5(4):1062–1070, 2008.

S. Waldherr, J. Poppenborg, and S. Knust. The transportation problem with second tier resources. Submitted for publication, 2013.

S. Warshall. A theorem on boolean matrices. *Journal of the ACM*, 9(1): 11–12, 1962.

J. D. Wiest. *The Scheduling of Large Projects with Limited Resources.* PhD thesis, Carnegie Institute of Technology, Pittsburgh, USA, 1963.

J. D. Wiest. Some properties of schedules for large projects with limited resources. *Operations Research*, 12(3):395–418, 1964.

T. Wolf. *Modellierung von Räumungen in Krankenhäusern und anderen Pflegeeinrichtungen.* Wuppertaler Berichte zum Brand- und Explosionsschutz. VdS-Schadenverhütung, Köln, Germany, 2001.

X. Zheng, T. Zhong, and M. Liu. Modeling crowd evacuation of a building based on seven methodological approaches. *Building and Environment*, 44(3):437–445, 2009.

X. Zheng, W. Li, and C. Guan. Simulation of evacuation processes in a square with a partition wall using a cellular automaton model for pedestrian dynamics. *Physica A: Statistical Mechanics and its Applications*, 389(11):2177–2188, 2010.