Applications of Resource-Constrained Project Scheduling in Service Operations Management

INAUGURALDISSERTATION

zur Erlangung der Würde eines Doctor rerum oeconomicarum
der Wirtschafts- und Sozialwissenschaftlichen Fakultät der Universität Bern

Adrian Zimmermann

von Seedorf BE

Betreuer: Prof. Dr. Norbert Trautmann
Professur für Quantitative Methoden der BWL
Departement Betriebswirtschaftslehre
Schützenmattstrasse 14, 3012 Bern

Bern, 22. November 2016

Die Fakultät hat diese Arbeit am 15. Dezember 2016 auf Antrag der beiden Gutachter Prof. Dr. Thomas Myrach und Prof. Dr. Stefan Creemers als Dissertation angenommen, ohne damit zu den darin ausgesprochenen Auffassungen Stellung nehmen zu wollen.

Acknowledgements

I elaborated this thesis during my time as a PhD student at the Chair of Quantitative Methods at the Department of Business Administration, University of Bern.

First and foremost, I would like to thank my supervisor Prof. Dr. Norbert Trautmann for his continuous support, patience, and valuable advice throughout my PhD studies. Special thanks go to my colleagues and co-workers who have provided many helpful comments during our common time at the Chair of Quantitative Methods.

Moreover, I would like to thank Prof. Dr. Thomas Myrach and Prof. Dr. Stefan Creemers for their willingness to act as external referees of my thesis.

I am also very grateful for the joint work with our industrial partners, papilio AG and bestview GmbH. I gratefully acknowledge the financial support provided by papilio AG during my first year as a PhD student.

Finally, I would like to express my gratitude to my parents, Tatjana and Stefan Zimmermann, and my brother, Martin Zimmermann, who have been a great support during this period.

Adrian Zimmermann

The present dissertation includes the following three papers.

Paper I

Zimmermann, A., Trautmann, N. (2017). A list-scheduling heuristic for the short-term planning of assessment centers. *Journal of Scheduling.* Advance online publication. DOI: 10.1007/s10951-017-0521-5

Paper II

Rihm, T., Trautmann, N., Zimmermann, A. (2016). MIP formulations for an application of project scheduling in human resource management. *Flexible Services and Manufacturing Journal.* Advance online publication. DOI: 10.1007/s10696-016-9260-8

Paper III

Zimmermann, A. (2017). A mixed-integer programming-based heuristic for project scheduling with work-content constraints. *European Journal of Industrial Engineering.* Advance online publication. DOI: 10.1504/EJIE.2017.10006712

Contents

Introduction		1
Paper I:	A list-scheduling heuristic for the planning of assessment centers	3
Paper II:	MIP formulations for an application of project scheduling in human resource management	34
Paper III:	A mixed-integer programming-based heuristic for project scheduling with work-content constraints	81

Introduction

Firms in the service industry sell intangible goods such as the selection of highly-skilled personnel to their customers. In general, such firms employ expensive staff to perform these services. Hence, an important aspect of service operations management is the planning of service operations such that personnel utilization is optimized. This thesis consists of three papers that present solution approaches for respective planning situations which are based on concepts and methods for the resource-constrained project scheduling problem (RCPSP). The RCPSP consists in scheduling a set of precedence-related activities that require time and scarce resources for execution such that the project duration is minimized.

In the first paper, we consider the problem of scheduling assessment centers. This problem has been stated to us by a human resource management (HRM) service provider. In an assessment center, candidates for job positions perform a series of tasks during which they are observed and evaluated by so-called assessors. The scheduling problem consists of determining (a) the start times of the tasks for each candidate and (b) the assignment of the assessors to the candidates and the tasks; thereby, complex rules for the assessor assignment must be considered. For this problem, we develop a list-scheduling heuristic procedure. At first, the candidates' tasks are sorted in a list; then, the tasks are scheduled sequentially based on their order in this list. Our computational results for a set of real-life problem instances indicate that the heuristic provides very good feasible schedules in a short amount of computation time. The main contributions of the paper are the development of a schedule-generation scheme for this complex planning situation and of an appropriate sorting mechanism. The heuristic procedure has been implemented in a software system and is used by the HRM service provider.

In the second paper, we develop five novel alternative mixed-integer linear programming (MIP) formulations for the assessment center planning problem described above. In general, different formulations can be used to model the same planning problem; however, the performance of MIP approaches strongly depends on the underlying formulation and therefore, alternative formulations should be considered for each planning problem. For the RCPSP, the difference in performance between such MIP formulations has only been compared based on generic test instances. In a comparative study, we analyze the performance of the five MIP formulations based on four real-life instances and 240 test instances derived from real-life data. The results indicate that good or optimal solutions are obtained for all instances within short computation time. The main contribution of the paper lies in the development of the novel MIP formulations and various new lower bounds for the scheduling of assessment centers, and in the comparison of these MIP formulations based on real-life data.

In the third paper, we consider the problem of scheduling the activities of a project in which each activity is characterized by a work content. In this generalized version of the RCPSP, the activities' usage of the scarce resources may change over time. Due to this generalization, this problem setting covers a wide range of problems in service operations management. For this project scheduling problem, we develop an MIP-based heuristic procedure. The heuristic procedure schedules the activities iteratively and reschedules subsets of activities to improve the utilization of the available resource capacities. Our computational results for a standard test set from the literature show that our heuristic outperforms the state-of-the-art methods. The paper contributes to the development of novel MIP-based solution methods in the field of resource-constrained project scheduling; in particular, the MIP models of our heuristic efficiently exploit the structural characteristics of the planning problem.

Paper I

A list-scheduling heuristic for the planning of assessment centers

Adrian Zimmermann Norbert Trautmann

Department of Business Administration
University of Bern

Contents

1.1 Introduction **4**
1.2 The planning problem **6**
1.3 Problem definition and related literature **8**
1.3.1 Related planning problems 9
1.3.2 Interpretation as extended MRCPSP 9
1.4 List-scheduling procedure **11**
1.4.1 Multi-pass procedure with random sampling 11
1.4.2 List generation 13
1.4.3 Schedule generation scheme 19
1.4.4 Assessor assignment 23
1.4.5 Results for the illustrative example 23
1.5 Computational study **23**
1.5.1 Test set 26
1.5.2 Computational results 28
1.6 Conclusions **31**
Bibliography **32**

Abstract

Many companies operate assessment centers to help them select candidates for open job positions. During the assessment process, each candidate performs a set of tasks, and the candidates are evaluated by so-called assessors. Additional constraints such as preparation and evaluation times, actors' participation in tasks, no-go relationships, and prescribed time windows for lunch breaks contribute to the complexity of planning such assessment processes. We propose a multi-pass list-scheduling heuristic for this novel planning problem; to this end, we develop novel procedures for devising appropriate scheduling lists and for generating a feasible schedule. The computational results for a set of example problems that represent or are derived from real cases indicate that the heuristic generates optimal or near-optimal schedules within relatively short CPU times.

1.1 Introduction

Empirical evidence indicates that human capital is a key success factor in firm performance (cf., e.g., Huselid 1995). A firm's human resource managers undertake the task of developing human capital for the benefit of the firm by recruiting new employees from a pool of candidates. Such personnel decisions require an evaluation of candidates' skills, know-how, and personalities. To perform these evaluations, human resource managers often use assessment centers (cf., e.g., Spychalski et al. 1997).

The assessment center planning problem (ACP) discussed in this paper has been reported to us by a human resource management (HRM) service provider that organizes assessment centers for firms on a regular basis. In these assessment centers, the candidates perform sets of tasks, during which they are observed and evaluated by assessors, who are typically managers or psychologists. Each candidate should be observed by approximately

half of all available assessors. So-called no-go relationships prohibit specific assessors from being assigned to certain candidates. After the assessment is completed, all assessors meet and compile an overall evaluation for each candidate. Some tasks are designed to involve role-playing and require one or more actors. The requirements for candidates, assessors, and actors can vary over the course of a task. In addition to completing all tasks, each candidate takes a lunch break within a prescribed time window. The ACP consists of determining (1) feasible start times for all tasks and lunch breaks for each candidate and (2) a feasible assignment of assessors and actors to all tasks such that the total waiting time for the assessors is minimized; the assessors meet before the start and after the completion of all tasks and lunch breaks, and therefore this objective corresponds to minimizing the duration of the assessment.

For this ACP, we provided a mixed-integer linear programming (MILP) formulation in Zimmermann and Trautmann (2014); this MILP generates optimal or near-optimal solutions for small instances, but for larger instances, the required CPU time may become prohibitively long. In principle, the ACP could be interpreted as an extension of the multi-mode resource-constrained project scheduling problem (MRCPSP, cf., e.g., Talbot 1982): each task performed by a candidate corresponds to a project activity, the assessment-center participants correspond to renewable resources, and each assignment of assessors to a task corresponds to an alternative execution mode. The ACP differs from the MRCPSP with respect to the specific assessor-assignment constraints, the time-varying resource requirements, and the time windows for lunch breaks.

In this paper, we propose a novel heuristic of the list-scheduling type (cf., e.g., Adam et al. 1997) for the ACP. First, the activities are ordered in a list; we devise four sets of alternative sorting criteria. Second, the activities are scheduled sequentially; we develop an appropriate schedule generation scheme (SGS) that comprises a procedure for assigning the assessors to the activities. We integrate the sorting criteria and the SGS into a multi-pass method comprising random sampling. For each set of sorting criteria, the

Table 1.1: Illustrative example: main data

Candidates	Assessors	Actors	Tasks
{C1, C2, C3}	{A1, A2, A3, A4}	{R1}	{E1, E2, E3, E4}

SGS is applied repeatedly; thereby, the order of the activities in the list is varied by applying random sampling. At the end, the best schedule obtained is returned. In an experimental performance analysis, we applied our multi-pass method to four real-world instances and 240 test instances that we constructed based on real-world data. It turned out that our novel approach requires short CPU time to generate optimal or near-optimal schedules.

The remainder of this paper is structured as follows. In Section 1.2, we illustrate the ACP with an example. In Section 1.3, we specify the ACP as the extension of an MRCPSP and discuss the related literature. In Section 1.4, we present our multi-pass list-scheduling procedure. In Section 1.5, we report on our computational results. In Section 1.6, we present concluding remarks and give an outlook on future research directions.

1.2 The planning problem

In this section, we illustrate the ACP with an example that is based on real-world data.

The assessment takes place over one day, although no maximum length of time is prescribed. The number of participants and the number of tasks are shown in Table 1.1. To improve comparability among the overall evaluations, each candidate must perform the same set of tasks. In addition, each candidate must perform each task exactly once.

In the assignment of assessors to the tasks, the following restrictions must be considered. To ensure an objective overall evaluation, each candidate should be observed by at least half of the number of assessors rounded down (i.e., 2 assessors). Because of fairness considerations, each candidate should be observed by approximately the same number of

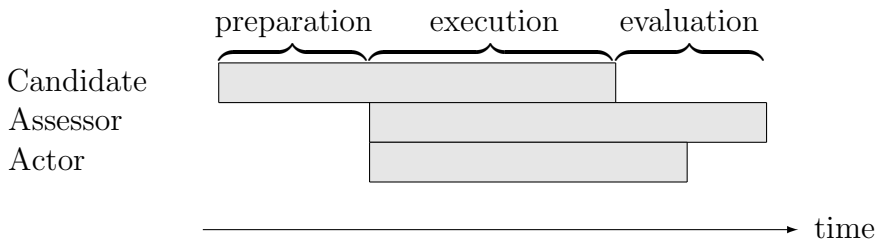


Figure 1.1: Varying requirements for candidates, assessors, and actors during a task

assessors. Hence, no candidate should be observed by more than half of the number of assessors rounded up plus one (i.e., 3 assessors). The difference between the upper and lower limits facilitates the assessor assignment without compromising fairness. Once an assessor has observed a candidate, the number of additional times that this same assessor can observe that same candidate is not limited. We refer to these restrictions as the 50% assignment rule. Additionally, a no-go relationship prohibits the assignment of assessor A4 to candidate C2. Such relationships arise if the candidate and the assessor know one another personally. We assume that an assignment of the assessors to the candidates exists which is feasible with respect to all of these constraints. We refer to assessors with one or more no-go relationships as no-go assessors.

The data for the tasks and the lunch break are listed in Table 1.2. Depending on the task type, one or two assessors must be present. Because task E1 involves role-playing, an actor is required. For example, the actor might represent an unhappy customer with whom the candidate must interact. Tasks E1 and E2 include a preparation period in which only the candidate is required. During the actual execution of the task, the candidate is joined by the assessors and, if required, the actor. After execution of tasks E1, E2, and E3, the assessors and actors have time to record their observations; this time is referred to as evaluation time and can differ for assessors and actors. Figure 1.1 illustrates these time-varying participant requirements. The duration and the preparation, execution, and evaluation times are stated in 5-minute time periods. The candidates have a lunch break

Table 1.2: Illustrative example: tasks and lunch break data

Task	E1	E2	E3	E4	Lunch break
Duration	18	29	16	6	6
Preparation time	8	19	0	0	-
Execution time	8	8	12	6	-
Evaluation time	2	2	4	0	-
Required number of assessors	2	2	2	1	-
Required number of actors	1	-	-	-	-

Table 1.3: Illustrative example: activities and lunch breaks

Candidate	C1	C2	C3
Task E1	1	2	3
Task E2	4	5	6
Task E3	7	8	9
Task E4	10	11	12
Lunch break	B	B	B

during which they cannot be involved in any task. The earliest and latest possible start times for the lunch break are time points 30 and 78, respectively. The assessors and actors take some short lunch break during the evaluation times of the tasks or during some idle times between the tasks, but these breaks are not scheduled.

Because the four tasks must be performed once by each of the three candidates, there are a total of 12 activities. Table 1.3 shows the indices of these activities. The lunch breaks for the candidates are denoted by B.

1.3 Problem definition and related literature

In Section 1.3.1, we sketch two well-known planning problems related to the ACP. In Section 1.3.2, we interpret the ACP as an extended MRCPSP. We show that the variants of the MRCPSP considered in the literature differ from the ACP.

1.3.1 Related planning problems

The ACP consists of two subproblems: an assignment subproblem and a scheduling subproblem. These two subproblems are also encountered within the MRCPSP and within the timetabling problem.

Timetabling involves allocating resources to objects that are being placed in space time (cf. Wren 1996). For example, in course timetabling for high schools, resources (students and teachers) are assigned to objects (events that correspond to individual meetings between students and teachers), and the objects are assigned to classrooms and time slots (cf. Carter and Laporte 1998). The objects and time slots are of equal duration, and the scheduling of events thus corresponds to an assignment subproblem. In an assessment center, different tasks have different durations; hence, the formulation of the ACP as a timetabling problem is impractical.

The objective of the basic MRCPSP is to (1) assign an execution mode to each activity and (2) determine a start time for each activity with the aim of minimizing project duration. There are precedence relations between activities (i.e., an activity can start only after all its predecessor activities have been completed). Renewable and non-renewable resources are required to execute the activities. Renewable resources are available with a constant capacity during each time period of the planning horizon (e.g., employees with a capacity of eight hours in each work day). The capacity of non-renewable resources is permanently reduced when an activity is executed (e.g., a monetary budget for the entire project). The selected mode determines how many units of which resource are required by an activity and the activity's duration. The activities must be scheduled to avoid violating precedence relations and resource capacities.

1.3.2 Interpretation as extended MRCPSP

Table 1.4 lists all of the problem elements present in the ACP and shows whether these elements are considered in the basic MRCPSP (✓) or not (✗). Notably, the ACP does

Table 1.4: Comparison: ACP and basic MRCPSP

Elements of planning problem	ACP	basic MRCPSP
Objective: minimize duration	✓	✓
Activities	✓	✓
Renewable resources	✓	✓
Alternative execution modes	✓	✓
Time-varying resource requirements	✓	✗
Time windows	✓	✗
Mode-assignment constraints	✓	✗

not include precedence relations or non-renewable resources. The basic MRCPSP can be extended to include time-varying resource requirements of tasks; for example, Cavalcante et al. (2001) and Hartmann (1999) consider time-varying resource requirements for labor and equipment, respectively. The time windows for lunch breaks can be considered by defining the release dates and activity deadlines. For instance, Drezet and Billaut (2008) consider activity time windows in an MRCPSP extension in which the resources are multi-skilled employees.

Specific mode-assignment constraints can be added to the basic MRCPSP. Thus, Li and Womer (2008) and Tareghian and Taheri (2007) associate prescribed quality values with each mode, and the modes must be selected such that the quality of the project is maximized or a minimum quality level is reached. Salewski et al. (1997) consider the MRCPSP with mode-identity constraints. The set of activities is partitioned into subsets, and all other activities of that subset must be executed in that same mode after a mode has been selected for an activity. None of these mode-assignment constraints corresponds to the assessor-assignment constraints of the planning problem discussed in this study. In the ACP, associating prescribed quality values with alternative assessor assignments is not feasible because all quality values depend on the overall assessor assignments to a candidate. Similarly, the assessor-assignment constraints prevent a meaningful partitioning of the assessment-center activities into subsets that require the same assessors.

1.4 List-scheduling procedure

In this section, we present our multi-pass list-scheduling procedure. A preliminary version of this heuristic can be found in Zimmermann and Trautmann (2015). We model each candidate and each assessor as a renewable resource with a capacity of one, and we model the set of actors as a renewable resource with a capacity that equals the number of actors. According to the HRM service provider, the duration of each task's preparation, execution, and evaluation time is expressed in 5-minute time periods. Hence, we divide the planning horizon into 5-minute time periods. In the following, we use the notation provided in Table 1.5.

In Section 1.4.1, we give an overview of the multi-pass procedure. In Section 1.4.2, we discuss the alternative criteria for ordering the activities in the list. In Section 1.4.3, we present the SGS that is used to generate a single schedule based on the order of the activities in the list. In Section 1.4.4, we describe how the assessors are assigned to the activities during the application of the SGS. In Section 1.4.5, we depict two schedules for the illustrative example obtained by our multi-pass list-scheduling procedure.

1.4.1 Multi-pass procedure with random sampling

For the MRCPSP, an efficient way to improve the performance of a procedure that generates a single schedule is to apply the procedure multiple times for different orders of the activities (cf., e.g., Boctor 1993). Thereby, the order of the activities is varied by applying different sorting criteria and random sampling. Different sorting criteria are used because the application of the same criteria does not provide the best solution for each problem instance. Because there is only a limited number of alternative sorting criteria, random sampling is employed to generate further activity orders. The benefit of such multi-pass procedures with random sampling is that they are simple and provide good solutions with a low computational effort.

For the ACP, we sequentially generate four different lists by applying alternative sorting

Table 1.5: Notation

A	Number of assessors $a = 1, \ldots, A$
C	Number of candidates $c = 1, \ldots, C$
D	Duration of schedule
D^*	Duration of best schedule obtained thus far
E	Set of tasks
E_c	Set of tasks not yet included in the list for candidate c
L	Number of positions in the list $l = 1, \ldots, L$
c_l	Candidate assigned to position l in the list
e_l	Task assigned to position l in the list
$count_e$	Number of times task e has been included in the list thus far
l^*	Last position in first part of the list
p^B	Duration of lunch break
p_e	Duration of task e
p_e^C	Preparation time of task e
r_e	Number of assessors required by task e
t^S	Earliest resource-feasible start time of an activity

criteria, and apply the SGS for each of the four lists. Thereby, we employ the random-sampling method of Damodaran et al. (2011): instead of selecting the first unscheduled activity in the list, the activity is randomly chosen among the first k unscheduled activities; the value of parameter k needs to be selected in advance.

The multi-pass procedure applied to each list is summarized as a flowchart in Figure 1.2. Before the first application of the SGS, the list is generated and the value of D^* is initialized. The sum of the durations of all tasks and the lunch break multiplied by the number of candidates corresponds to an upper bound for D^*, i.e., $D^* := C(\sum_{e \in E} p_e + p^B)$. Then, the SGS is applied multiple times until a prescribed time limit has been reached. After this procedure has been repeated for all four lists, the best solution obtained over all four lists is returned.

We employ the following local-search logic in the multi-pass procedure. Each time a schedule has been generated, the duration of the generated schedule D is compared with the value of D^*. If the generated schedule has a shorter duration, then the randomized order in which the activities were scheduled is used as the new list and the value of D^* is

set to D.

1.4.2 List generation

Priority rules that have been proposed in the literature are not appropriate for determining the order of activities in the ACP because (a) there are no precedence constraints between the tasks and (b) there is no difference between activities that refer to the same task. In the following, we derive novel criteria for ordering activities in the list.

The list includes all assessment-center activities. One activity corresponds to a unique combination of candidate and task. Because of their time windows, we do not include lunch breaks in the list. The list-generation procedure is depicted as a flowchart in Figure 1.4.

First, the number of positions in the list, L, is calculated by multiplying the number of tasks, $|E|$, by the number of candidates C. Next, the candidates $1, \ldots, C$, are assigned $|E|$ times to the positions $1, \ldots, L$. Finally, the tasks are assigned to the positions $1, \ldots, L$, such that each task is included once for each candidate. An activity list is obtained such that a list-adjustment procedure can be applied. In that case, an adjusted activity list is obtained.

The list consists of three parts to which the following criteria apply (cf. Figure 1.3). If these criteria are not sufficient to distinguish between tasks, the task index is used as a tie-breaker.

Sorting criteria: last part of list

To the last C positions in the list, the task of shortest duration is assigned. The activities that are in the last part of the list may have to be scheduled at the end of the partially generated schedule. Figure 1.5 illustrates how an overall shorter schedule may be obtained by considering the shortest task last. The dark grey bars represent the time during which the participants are occupied in the partial schedule. The light grey bars correspond to a task with long duration (top) and a task with short duration (bottom), respectively, that

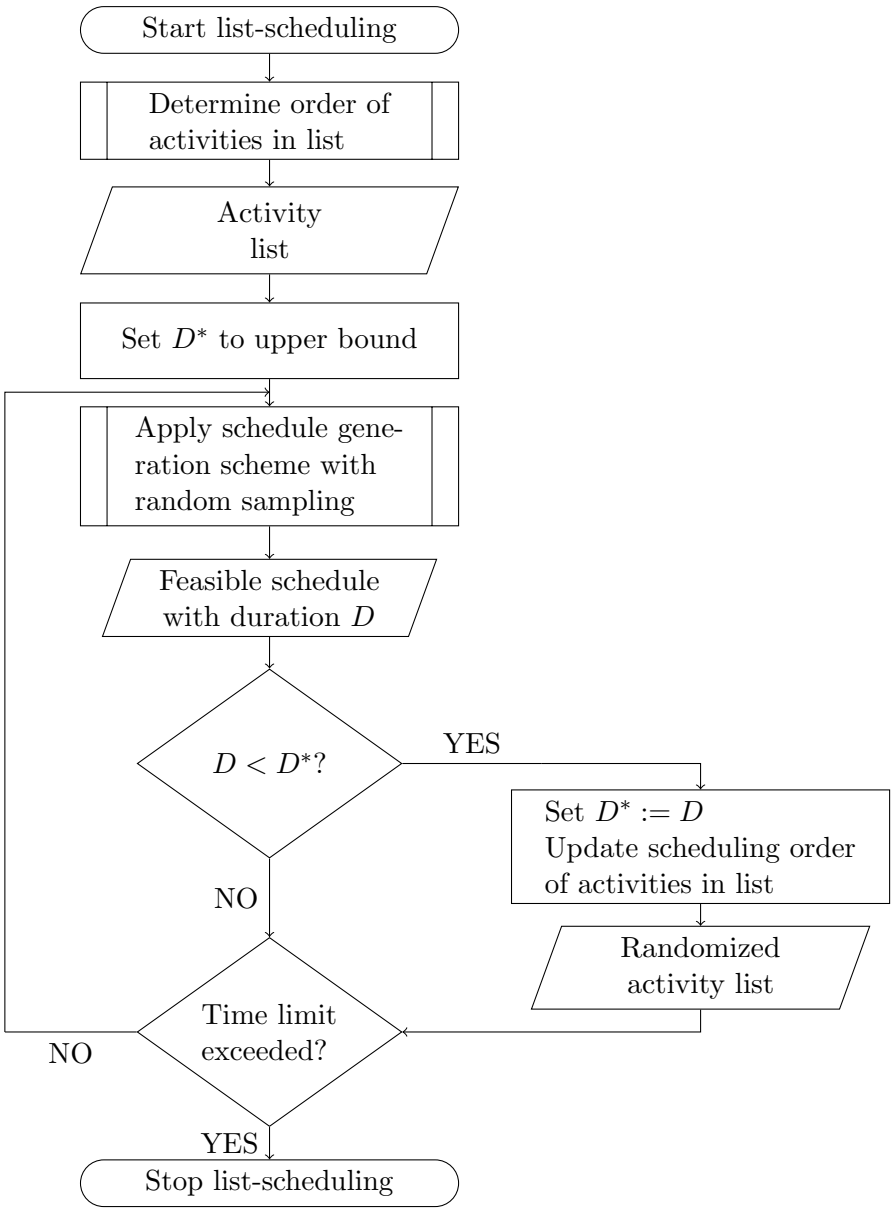


Figure 1.2: Flowchart: multi-pass list-scheduling procedure for one list

Position	1	2	...													L
Candidate	1	2	...	C	1	2	...	C	1	2	...	C	1	2	...	C
Criteria	Shortest preparation time			Largest distance between same task									Shortest duration			

Figure 1.3: Criteria for assigning tasks to positions

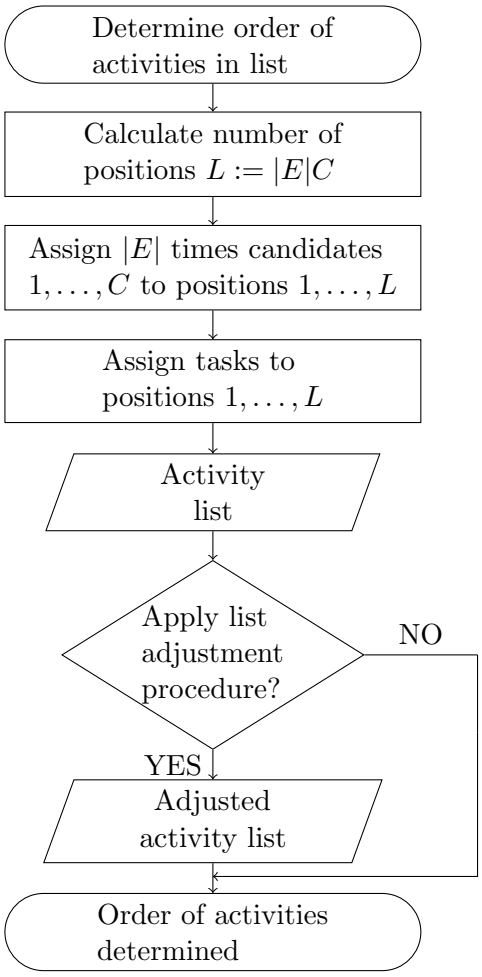


Figure 1.4: Flowchart: list-generation procedure

have to be scheduled at the end of the partial schedule.

The pseudocode for assigning the task with shortest duration to the last C positions in the list is presented in Algorithm 1. First, the task of shortest duration e^* is determined (line 1). This task is then assigned to the positions $L - C + 1, \ldots, L$ (line 3). Each time the task is assigned to a position, the counter $count_{e^*}$ (which is initialized with $count_e = 0$ for all $e \in E$) is increased by one (line 4), and the task is removed from the set E_{c^*} of the corresponding candidate c^* assigned to that position (lines 5 and 6). Because task e^* has been considered for all candidates, it is removed from the set of tasks E (line 7).

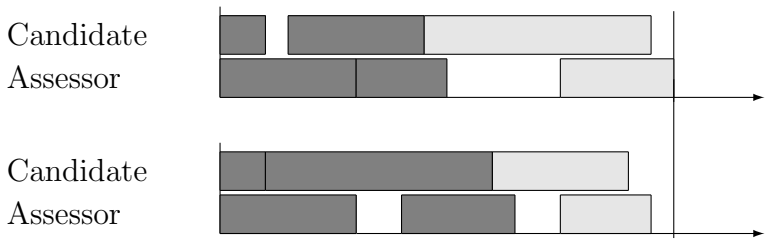


Figure 1.5: Reduced schedule duration by considering shortest task last

Algorithm 1: Assignment of tasks to last C positions in the list

1 $e^* \leftarrow \min\{e \in E : p_e = \min_{e \in E}(p_e)\}$
2 **for** $l = L - C + 1$ **to** L **do**
3 $\quad e_l \leftarrow e^*$
4 $\quad count_{e^*} \leftarrow count_{e^*} + 1$
5 $\quad c^* \leftarrow c_l$
6 $\quad E_{c^*} \leftarrow E_{c^*} \setminus \{e^*\}$
7 $E \leftarrow E \setminus \{e^*\}$

Sorting criteria: first part of list

To the positions in the first part of the list, the task with the shortest preparation time is assigned. The number of positions for which these criteria apply corresponds to the number of activities that can be performed simultaneously with respect to the number of assessors required. By scheduling the corresponding activities first, some of the assessors' waiting time can be reduced.

The pseudocode for assigning the task with the shortest preparation time to the positions in the first part of the list is presented in Algorithm 2. First, the task with the shortest preparation time e^* is determined (line 1). Then, the number of positions in the first part of the list l^* is calculated (line 2). This number corresponds to the minimum of C and the closest integer lower than or equal to the number of assessors A divided by the required number of assessors of the selected task r_{e^*}. The selected task is then assigned to the positions $1, \ldots, l^*$ (lines 3 and 4). Again, the counter $count_{e^*}$ is increased, and the task is removed from the set E_c (lines 5, 6, and 7).

Algorithm 2: Assignment of tasks to first positions in the list

1 $e^* \leftarrow \min\{e \in E : p_e^C = \min_{e \in E}(p_e^C)\}$
2 $l^* \leftarrow \min(\lfloor A/r_{e^*} \rfloor, C)$
3 **for** $l = 1$ **to** l^* **do**
4 $\quad e_l \leftarrow e^*$
5 $\quad count_{e^*} \leftarrow count_{e^*} + 1$
6 $\quad c^* \leftarrow c_l$
7 $\quad E_{c^*} \leftarrow E_{c^*} \setminus \{e^*\}$

Sorting criteria: middle part of list

To the positions in the middle part of the list, the tasks are assigned such that the distance between two positions of the same task is maximized. As a secondary criterion, either the minimum or maximum preparation time is used. A list is generated with each of the two secondary criteria. By scheduling activities referring to different tasks such that they run (partially) in parallel, some candidates may begin the preparation of one task while the assessors are occupied with the execution of another task.

The pseudocode for assigning the tasks to the positions in the middle part of the list is presented in Algorithm 3. First, for each candidate c, the tasks in E_c are ordered based on the secondary criterion employed (line 2). Then, for each position $l^* + 1, \ldots, L - C$, the corresponding candidate c^* is selected (line 4). The tasks in E_{c^*} are checked sequentially whether they have been included in the list the least often thus far (line 6). In that case the task is assigned to the position (line 7), $count_e$ and the set E_{c^*} are updated (lines 8 and 9), and the next position in the list is selected (line 10).

Sorting criteria: optional list adjustment

The list-adjustment procedure adjusts the list as follows.

First, the tasks assigned to the positions $l^* + 1, \ldots, C$ are switched with the task with the longest preparation time. By scheduling the corresponding activities such that they run (partially) in parallel with activities included in the first part of the list, some of the candidates can already begin with the preparation of a task while the assessors are

Algorithm 3: Assignment of tasks to remaining positions in the list

```
for c = 1 to C do
    order tasks in E_c based on secondary criterion
for l = l* + 1 to L − C do
    c* ← c_l
    for e ∈ E_{c*} do
        if count_e = min_{e∈E_{c*}} count_e then
            e_l ← e
            count_e ← count_e + 1
            E_{c*} ← E_{c*} \ {e}
            break
```

Table 1.6: Illustrative example: four alternative lists

Position	Secondary criteria											
	maximum preparation time						minimum preparation time					
	non-adjusted list			adjusted list			non-adjusted list			adjusted list		
	candidate	task	activity	candidate	task	activity	candidate	task	activity	candidate	task	activity
1	C1	E3	7	C1	E3	7	C1	E3	7	C1	E3	7
2	C2	E3	8	C2	E3	8	C2	E3	8	C2	E3	8
3	C3	E2	6	C3	E2	6	C3	E1	3	C3	E2	6
4	C1	E1	1	C1	E1	1	C1	E2	4	C1	E2	4
5	C2	E2	5	C2	E2	5	C2	E1	2	C2	E1	2
6	C3	E1	3	C3	E1	3	C3	E2	6	C3	E1	3
7	C1	E2	4	C3	E3	9	C1	E1	1	C3	E3	9
8	C2	E1	2	C2	E1	2	C2	E2	5	C2	E2	5
9	C3	E3	9	C1	E2	4	C3	E3	9	C1	E1	1
10	C1	E4	10	C1	E4	10	C1	E4	10	C1	E4	10
11	C2	E4	11	C2	E4	11	C2	E4	11	C2	E4	11
12	C3	E4	12	C3	E4	12	C3	E4	12	C3	E4	12

occupied.

Second, the order of activities in the last C positions of the middle part of the list is switched by reversing the order in which the candidates are considered. If this part of the list contains activities with short preparation times, then considering the candidates in reverse order can reduce some of the assessors' waiting time.

The four alternative lists for the illustrative example are depicted in columns 4, 7, 10, and 13 of Table 1.6. The dashed lines indicate the three different parts of the list.

1.4.3 Schedule generation scheme

The flowchart of the SGS is depicted in Figure 1.6. Based on their order in the list, activities are selected sequentially and scheduled as early as possible. To this end, the earliest resource-feasible start time t^S is determined. It may no longer be possible to schedule the lunch break within its time window when the selected activity is scheduled at t^S. In that case, the corresponding lunch break is scheduled first, and t^S is calculated anew for the selected activity. After all of the activities in the list have been scheduled, all remaining lunch breaks are scheduled, and a procedure for rescheduling the lunch breaks is applied.

Earliest resource-feasible start time t^S

The flowchart of the procedure that determines the earliest resource-feasible start time t^S for the selected activity is depicted in Figure 1.7. The procedure is initiated by selecting the first time period of the planning horizon. From the selected time period onward, the availability of the required candidate and–in the case of a role-play task–the actors are determined. When determining availabilities, we consider only those time periods during which the respective participants must be present.

Once the candidate (and, in the case of a role-play task, an actor) is available, a sub-procedure for the assessor assignment is initiated (see Section 1.4.4). If the assessor assignment is successful, t^S is set to the currently selected time period, and the procedure ends. Otherwise, the next time period of the planning horizon is selected, and the availabilities of the participants are examined anew.

Lunch breaks

Before an activity is scheduled at t^S, the following procedure determines whether the lunch break time window is violated. Assuming that the activity is scheduled at t^S, the candidate's resulting availability is determined throughout the lunch break time window.

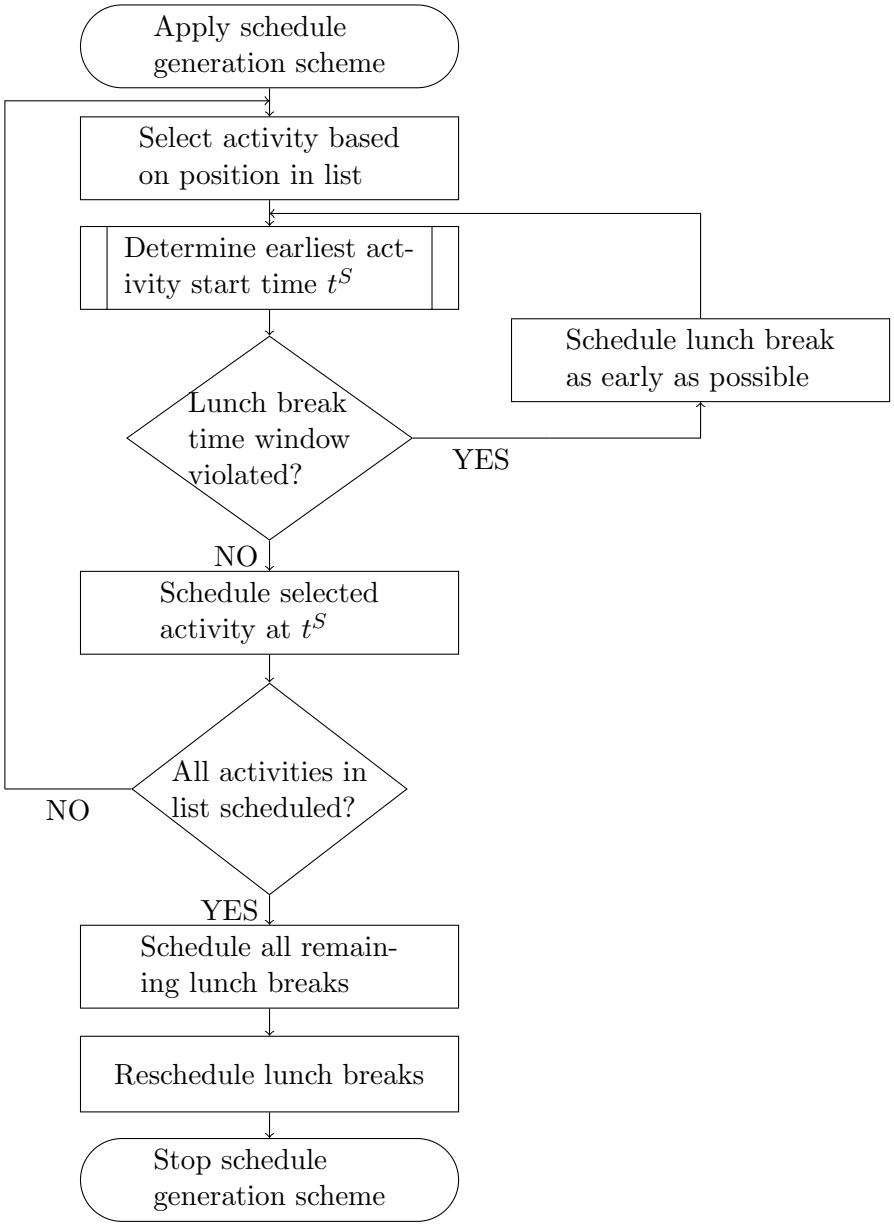


Figure 1.6: Flowchart: schedule generation scheme

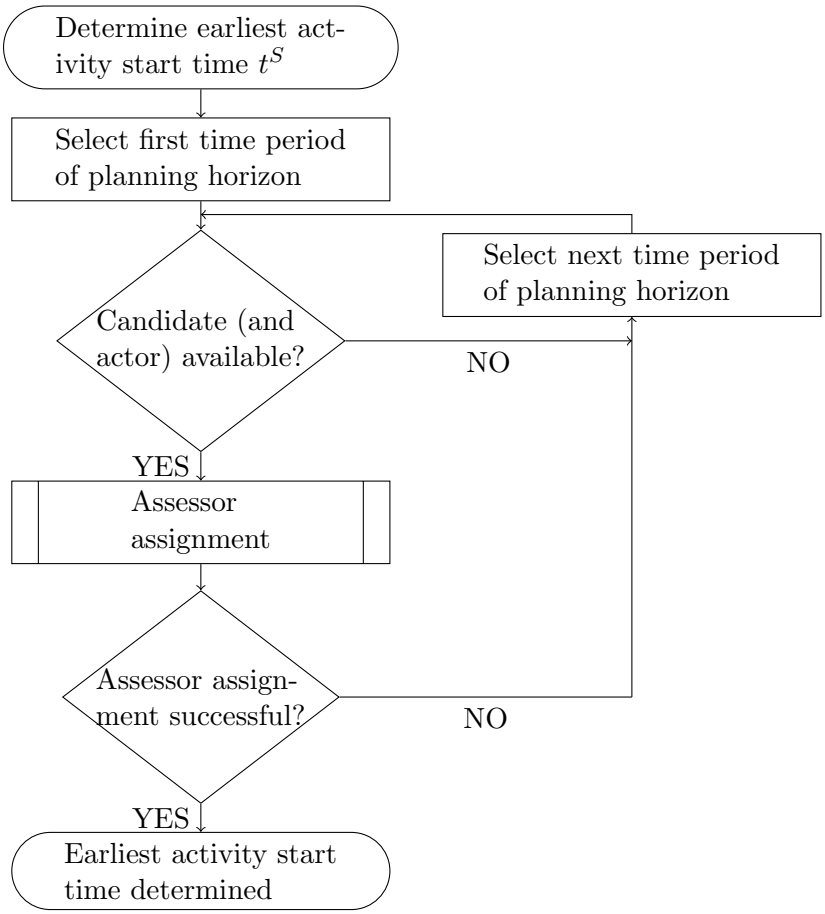


Figure 1.7: Flowchart: procedure for determining the earliest resource-feasible activity start time t^S

If it is impossible to schedule the lunch break within this time window, then the lunch break is scheduled before the activity.

The situation described above might never arise, and some lunch breaks might thus never be considered during the scheduling of activities in the list. After all activities have been scheduled, all remaining lunch breaks are scheduled at their earliest resource-feasible start times.

Scheduling the lunch break after the completion of the last activity might unnecessarily increase the duration of the schedule. Beginning with the candidate with the smallest index value, for each candidate we determine whether the lunch break is taken after the

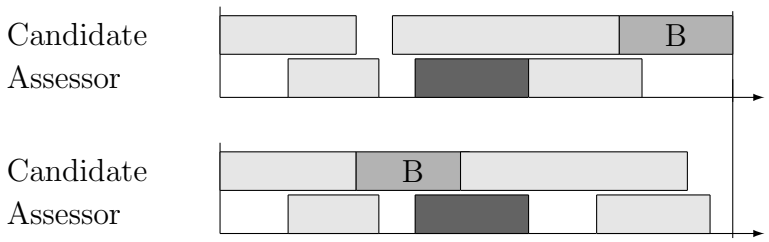


Figure 1.8: Reduced schedule duration through lunch break switch

last activity. If the schedule duration is reduced, we switch the order of the last activity and the lunch break. The reduction of a schedule's duration through such a switch is illustrated in Figure 1.8. The light grey bars correspond to the activities performed by the same candidate. The dark grey bar represents the time during which the assessor is occupied with another candidate's activity.

1.4.4 Assessor assignment

The flowchart of the procedure that assigns the assessors to an activity is depicted in Figure 1.9. The assessors are first ordered randomly and then selected sequentially based on that random order. An assessor is temporarily assigned if (1) the assignment does not violate the lower and upper limit of the 50% assignment rule, (2) there is no no-go relationship with the required candidate, and (3) the assessor is available.

The procedure ends when the required number of assessors has been assigned and the temporarily assignments are fixed–or once all assessors have been examined. If the required number of feasible assessors could not be assigned, then any temporary assignments to the selected activity are reversed.

Whether an assessor assignment violates the 50% assignment rule depends on which assessors have observed the required candidate thus far. The following two cases and the corresponding feasibility criteria are considered.

a) The number of different assessors who observe the required candidate equals the upper limit. Only assessors who have already observed the required candidate at

least once are feasible.

b) The number of different assessors who observe the required candidate lies below the lower limit. Only assessors who have not yet observed the required candidate are feasible.

In any other case, no additional feasibility criteria apply. Note that the criteria in a) or b) might become active after the first assessor has been assigned. Additionally, because of the random order in which assessors are considered, assessor assignment may vary in each application of the SGS.

1.4.5 Results for the illustrative example

For the illustrative example, we obtained an optimal schedule with a duration of 70 time units by applying the MILP of Zimmermann and Trautmann (2014). Two alternative schedules for the illustrative example obtained by our multi-pass list-scheduling procedure are depicted in Figure 1.10. The duration of the best schedule generated with the list based on the maximum preparation time criterion and random assessor assignment is 71 time units. An optimal schedule is generated using the random-sampling method.

1.5 Computational study

We have implemented the multi-pass list-scheduling heuristic presented in Section 1.4 in Java and applied it to four real-world instances and 240 test instances generated based on real-world data. Compared with the analysis presented in Zimmermann and Trautmann (2015), these test instances enable a more thorough analysis of the performance of the procedure.

We compare our novel approach with the MILP presented in Zimmermann and Trautmann (2014). We implemented the MILP in AMPL and used the Gurobi Optimizer 6.5 as solver. We limited the CPU time of the solver to 3,600 seconds for the real-world instances

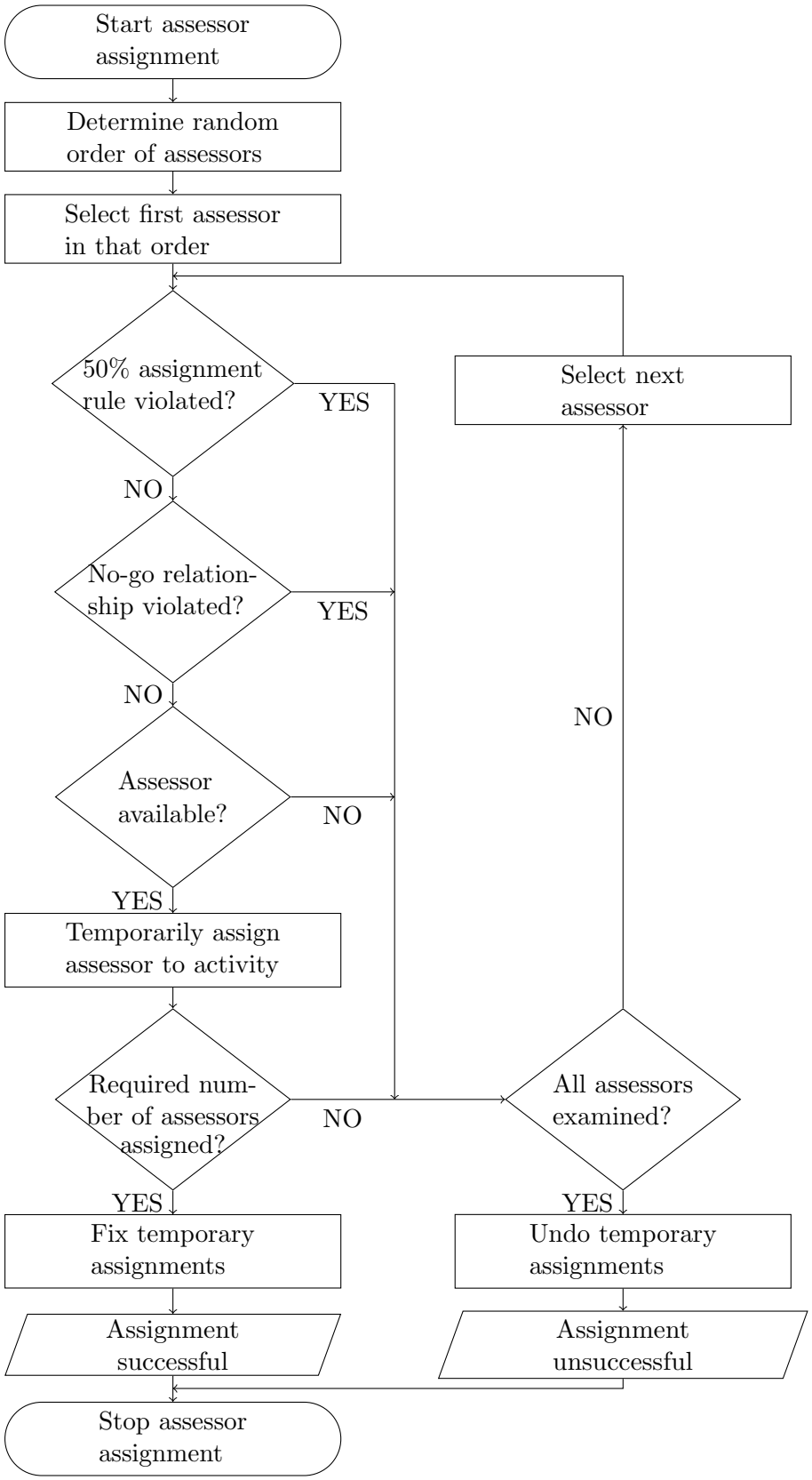


Figure 1.9: Flowchart: procedure for assigning the required number of assessors to an activity

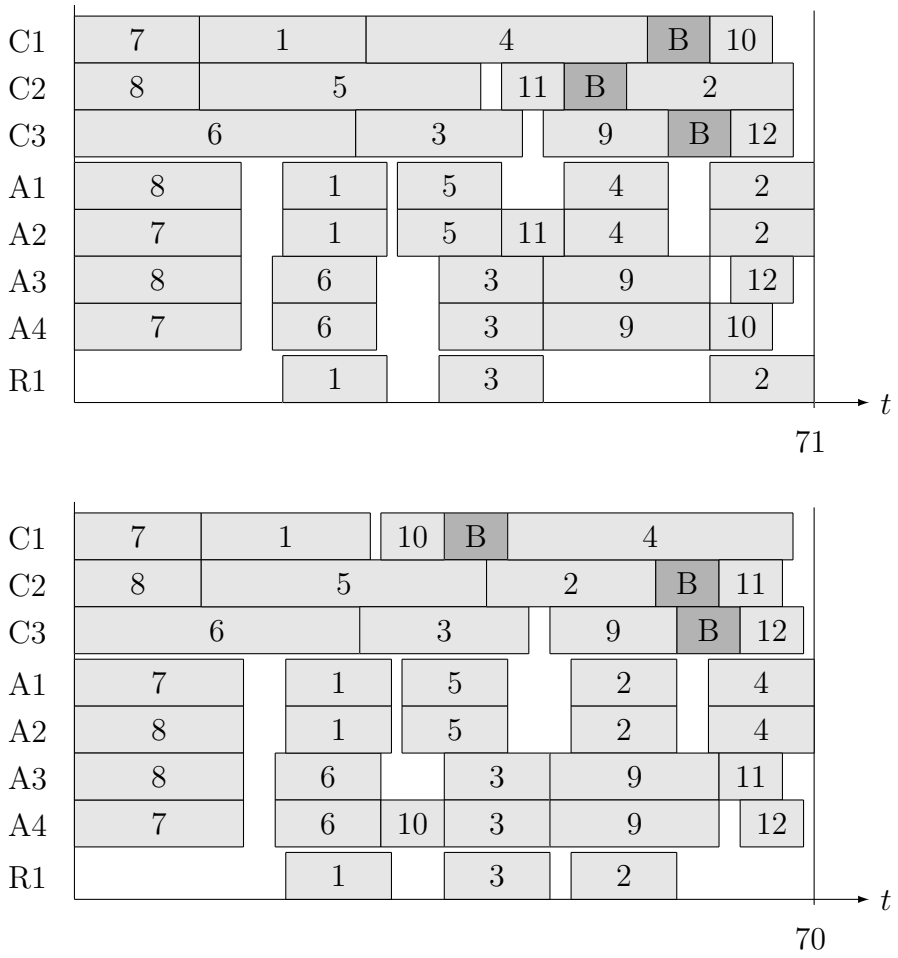


Figure 1.10: Illustrative example: schedules obtained by the heuristic without random sampling (top) and with random sampling (bottom)

and to 1,200 seconds for the test instances. According to the HRM service provider, it is important that the heuristic generates good feasible solutions in a short amount of time. For example, in the event that some of the participants have to cancel their participation at short notice, the entire assessment must be rescheduled. Hence, the HRM service provider prescribed a CPU time limit of 10 seconds. In order to evaluate the speed at which the heuristic generates good solutions, we also applied the heuristic using CPU time limits of 1 second, 5 seconds, and 100 seconds. The available CPU time was divided equally among the multi-pass procedures for the individual lists. All calculations were performed on a desktop PC with an Intel Core i5 CPU with 2.7GHz and 4GB RAM.

Table 1.7: Real-world instances: main data

Instance	C	A	R	I	no-go relationships
R1	7	10	2	42	no
R2	11	11	3	66	no
R3	9	11	3	54	yes
R4	6	9	3	36	no

In Section 1.5.1, we describe the test instances that we used in our computational study. In Section 1.5.2, we discuss our computational results.

1.5.1 Test set

The data for four real-world instances were provided by the HRM service provider. These instances include the same set of five tasks and one lunch break for each candidate. The number of candidates C, assessors A, actors R, and activities and lunch breaks I of the instances are listed in Table 1.7. The last column indicates whether any no-go relationships exist.

Additionally, we generated 240 test instances by varying five factors in a full factorial design. The factors and their experimental levels are summarized in Table 1.8. The employed experimental levels are based on the four real-world instances and additional real-world data made available to us by the HRM service provider. Factors C and $|E|$ correspond to the number of candidates and tasks of each instance. In combination, the employed levels of these two factors determine the number of activities and lunch breaks I for each instance. We partitioned 15 real-world tasks into three subsets: (a) role-play tasks, (b) non-role-play tasks that require only one assessor, and (c) non-role-play tasks that require two assessors. The tasks for an instance were randomly selected as follows: two tasks from subset (a), one task from subset (b), and–depending on $|E|$–one or two tasks from subset (c). The number of assessors A of an instance equals the integer closest to the

Table 1.8: Test instances: factors and experimental levels

Factor	Corresponds to	Experimental levels
C	number of candidates	$4, 5, \ldots, 10, 11$
$\lvert E\rvert$	number of tasks	$4, 5$
a^S	average number of assignments per assessor	$6.0, 8.5, 10.4$
a^N	proportion of assessors who are no-go assessors	$1/6, 1/3$
n	average number of no-go relationships per no-go assessor	$2, 3$

total number of assignments divided by the factor a^S. The total number of assignments corresponds to the sum of assessor requirements over all activities. The experimental levels used for a^S correspond to the observed real-world minimum, average, and maximum. Let A^N denote the set of assessors with at least one no-go relationship (i.e., the set of no-go assessors). The number of no-go assessors $|A^N|$ for an instance equals the integer closest to $a^N A$. We randomly include assessors in the set A^N until we reach this number. The number of no-go relationships N for an instance equals $n|A^N|$. The no-go relationships are randomly assigned to pairs of candidates and assessors belonging to set A^N such that (1) each assessor in A^N has at least one no-go relationship and (2) at least $\lfloor A/2 \rfloor$ different assessors can be assigned to each candidate. The actors are not considered to be a critical resource, and they are paid for each role-play task in which they actually perform. Hence, we set the number of actors $R = 3$ for all instances, which corresponds to the observed real-world maximum.

For each combination of factor levels, we generate an instance that leads to $8 \cdot 2 \cdot 3 \cdot 2 \cdot 2 = 192$ test instances. We also generate instances without no-go relationships (i.e., $a^N = n = 0$). This step leads to an additional $8 \cdot 2 \cdot 3 = 48$ test instances.

Table 1.9: Real-world instances: results

	R1	R2	R3	R4
MILP (3600sec)	90	122	101	84
CPU	3128	2987	1386	344
LB	80	110	90	80
Heur. (1sec)	86	114	98	83
Heur. (5sec)	84	**113**	97	**82**
Heur. (10sec)	83	**113**	**96**	**82**
Heur. (100sec)	**82**	**113**	**96**	**82**

1.5.2 Computational results

The results for the four real-world instances R1, ..., R4 are reported in Table 1.9. The objective function value of the best schedules obtained by the MILP are shown in row [MILP (3600sec)]. Furthermore, the row [CPU] shows the number of seconds after which the MILP obtained the best schedules, and the row [LB] lists the lower bounds obtained by the MILP. None of the instances were solved to optimality within the prescribed CPU time limit. The rows [Heur. (1sec)], [Heur. (5sec)], [Heur. (10sec)], and [Heur. (100sec)] show the results for the heuristic with a CPU time limit of 1, 5, 10, and 100 seconds, respectively. In comparison with the MILP, the heuristic obtains better solutions for all four instances with a CPU time limit of 1 second or longer.

The aggregated results for all 240 test instances are reported in Table 1.10. The column [average OFV] lists the average objective function values and the column [average GAP] shows the average gaps. To calculate the gaps, we used the lower bound LB obtained by the MILP and the formula GAP=(OFV–LB)/OFV. Column 4 shows the number of instances solved to optimality. A heuristic solution is optimal if its objective function value equals the lower bound obtained by the MILP. Column 5 shows the number of times that the procedures generated the best solutions; for example, for 216 instances, [Heur. (100sec)] generates solutions with an OFV that is smaller than or equal to the OFV obtained by the other procedures. The row [Heur. (10sec, random list)] shows the

Table 1.10: Test instances: aggregated results

	average OFV	average GAP [%]	number of opt.	number of best
MILP (1200sec)	109.4	9.3	22	72
Heur. (1sec)	105.3	6.1	22	77
Heur. (5sec)	104.7	5.5	29	119
Heur. (10sec)	104.5	5.3	30	136
Heur. (10sec, random list)	108.1	8.6	17	35
Heur. (10sec, no sampling)	106.8	7.5	16	43
Heur. (100sec)	**104.0**	**4.8**	**38**	**216**

results of our heuristic when the lists are generated randomly. The row [Heur. (10sec, no sampling)] shows the results of our heuristic when no random sampling is employed. Notably, we still run the heuristic for 10 seconds in this case because of the randomized assignment of the assessors to the activities. On average, the heuristic obtains better solutions than the MILP regardless of the prescribed CPU time limit. An increase of the CPU time limit from 1 to 5 seconds leads to a larger improvement of the average gap than an increase from 5 to 10 seconds. Compared to a random activity order, the employment of our sorting criteria improves the average gap from 8.6% to 5.3%. Furthermore, the average gap is improved from 7.5% to 5.3% with the application of the random-sampling method. The results obtained by [Heur. (100sec)] indicate that a CPU time limit of 100 seconds leads to even better solutions; however, compared against the results for a CPU time limit of 10 seconds, the improvement is relatively small.

Table 1.11 shows the average results obtained with each of the four lists and a total CPU time limit of 10 seconds, i.e., a CPU time limit of 2.5 seconds per list. On average, the non-adjusted list generated with the minimum preparation time criterion leads to the best results. However, neither list obtains the best results for each instance of the test set.

The average objective function values and gaps obtained by [MILP (1200sec)] and [Heur. (10sec)] for different factor levels are reported in Table 1.12. Each instance of the test set has between 20 and 66 activities and lunch breaks. We list the average results

Table 1.11: Test instances: list-dependent results

	average OFV	average GAP [%]	number of opt.	number of best
Max. preparation time				
non-adjusted list	105.4	6.2	22	126
adjusted list	105.9	6.6	14	96
Min. preparation time				
non-adjusted list	**105.3**	**6.1**	**27**	**134**
adjusted list	105.8	6.4	17	101

for the three different ranges of I that correspond to small-, medium-, and large-sized instances. For the three ranges of I, the heuristic obtains lower average gaps. Similarly, the heuristic obtains lower average gaps for all levels of the factors a^S, a^N, and n.

Table 1.12: Test instances: factor-dependent results

		MILP (1200sec)		Heur. (10sec)	
		average OFV	average GAP [%]	average OFV	average GAP [%]
I	20 to 30	114.2	4.0	**113.7**	**3.6**
	31 to 50	102.7	8.0	**99.7**	**5.4**
	51 to 66	115.2	18.1	**101.4**	**7.4**
a^S	6	81.9	10.3	**78.4**	**7.0**
	8.5	111.9	10.9	**105.7**	**5.9**
	10.4	134.4	6.6	**129.4**	**3.1**
a^N	0	108.5	8.8	**103.6**	**4.8**
	1/6	109.6	9.5	**104.5**	**5.4**
	1/3	109.7	9.3	**105.0**	**5.5**
n	0	108.5	8.8	**103.6**	**4.8**
	2	108.8	9.0	**103.6**	**4.8**
	3	110.4	9.7	**105.8**	**6.1**

1.6 Conclusions

In this work, we developed a multi-pass list-scheduling heuristic for a real-world scheduling problem arising in the context of assessment centers, which has been reported to us by an HRM service provider. Our computational results for a set of real-world instances and a set of instances constructed based on real-world data indicate that our heuristic consistently generates good schedules with a small amount of computational time. For the set of real-world instances, compared to the schedules manually generated by the HRM service provider, the heuristic obtains better schedules in a much shorter amount of time. A simplified version of our list-scheduling heuristic has been implemented in the planning software used by the HRM service provider.

Some assessments include group tasks that are performed by multiple candidates simultaneously in the presence of several assessors. The list-scheduling heuristic presented in this paper should be extended by exploring procedures for forming these types of groups and scheduling activities accordingly. Furthermore, the heuristic presented in this paper can be used to analyze the performance of alternative solution approaches to be developed in the future, e.g., MIP-based heuristics such as sketched in Rihm and Trautmann (2016).

Bibliography

Adam, T. L., Chandy, K. M., Dickson, J. R., 1997. A comparison of list schedules for parallel processing systems. Communications of the ACM 17(12), 685–689.

Boctor, F. F., 1993. Heuristics for scheduling projects with resource restrictions and several resource-duration modes. The International Journal of Production Research 31 (11), 2547–2558.

Carter, M. W., Laporte, G., 1998. Recent developments in practical course timetabling. In: Burke, E. K., Carter, M. W. (Eds.), Practice and Theory of Automated Timetabling II. Springer, Berlin, pp. 3–19.

Cavalcante, C. C. B., de Souza, C. C., Savelsbergh, M. W. P., Wang, Y., Wolsey, L. A., 2001. Scheduling projects with labor constraints. Discrete Applied Mathematics 112 (1–3), 27–52.

Damodaran, P., Vélez-Gallego, M. C., Maya, J., 2011. A GRASP approach for makespan minimization on parallel batch processing machines. Journal of Intelligent Manufacturing 22 (5), 767–777.

Drezet, L.-E., Billaut, J.-C., 2008. A project scheduling problem with labour constraints and time-dependent activities requirements. International Journal of Production Economics 112 (1), 217–225.

Hartmann, S., 1999. Project scheduling under limited resources: models, methods, and applications. In: Number 478 in Lecture Notes in Economics and Mathematical Systems. Springer, Berlin.

Huselid, M. A., 1995. The impact of human resource management practices on turnover, productivity, and corporate financial performance. Academy of Management Journal 38 (3), 635–672.

Li, H., Womer, K., 2008. Modeling the supply chain configuration problem with resource constraints. European Journal of Operational Research 26 (6), 646–654.

Rihm, T., Trautmann, N., 2016. A decomposition approach for an assessment center planning problem. In: Ruiz, R., Alvarez-Valdes, R. (Eds.), Proceedings of the 15th International Conference on Project Management and Scheduling. Valencia, pp. 206–209.

Salewski, F., Schirmer, A., Drexl, A., 1997. Project scheduling under resource and mode identity constraints: model, complexity, methods, and application. European Journal of Operational Research 102 (1), 88–110.

Spychalski, A. C., Quinones, M. A., Gaugler, B. B., Pohley, K., 1997. A survey of assessment center practices in organizations in the united states. Personnel Psychology 50 (1), 71–90.

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

Tareghian, H. R., Taheri, S. H., 2007. A solution procedure for the discrete time, cost and quality tradeoff problem using electromagnetic scatter search. Applied Mathematics and Computation 190 (2), 1136–1145.

Wren, A., 1996. Scheduling, timetabling and rostering – a special relationship? In: Burke, E. K., Ross, P. (Eds.), Practice and Theory of Automated Timetabling. Springer, Berlin, pp. 46–75.

Zimmermann, A., Trautmann, N., 2014. Scheduling of assessment centers: an application of resource-constrained project scheduling. In: Fliedner, T., Kolisch, R., Naber, A. (Eds.), Proceedings of the 14th International Conference on Project Management and Scheduling. Munich, pp. 263–266.

Zimmermann, A., Trautmann, N., 2015. A list-scheduling approach for the planning of assessment centers. In: Hanzálek, Z., Kendall, G., McCollum, B., Šůcha, P. (Eds.), Proceedings of the Multidisciplinary International Scheduling Conference: Theory and Application. Prague, pp. 541–554.

Paper II

MIP formulations for an application of project scheduling in human resource management

Tom Rihm Norbert Trautmann Adrian Zimmermann

Department of Business Administration
University of Bern

Contents

2.1 Introduction 35
2.2 Planning problem 38
2.2.1 Illustration of the planning problem 38
2.2.2 Relation to the RCPSP 41
2.3 Literature review 41
2.3.1 MIP formulations for the RCPSP 42
2.3.2 Comparative studies of MIP formulations 43
2.4 MIP formulations for the ACP 45
2.4.1 Formulation CT–A 45
2.4.2 Formulation CT–F 51
2.4.3 Formulation CT–O 54
2.4.4 Formulation DT–P 58
2.4.5 Formulation DT–O 60
2.5 Lower bounds 61
2.5.1 Lower bounds based on the assessors' workload 62
2.5.2 Lower bounds based on the candidates' workload 63
2.6 Comparative analysis 68
2.6.1 Instances 68
2.6.2 Computational results: real-life instances 70
2.6.3 Computational results: test instances 72
2.6.4 Computational results: problem-specific lower bounds 76
2.7 Conclusions 76
Bibliography 78

Abstract

In the literature, various discrete-time and continuous-time mixed-integer linear programming (MIP) formulations for project scheduling problems have been proposed. The performance of these formulations has been analyzed based on generic test instances. The objective of this study is to analyze the performance of discrete-time and continuous-time MIP formulations for a real-life application of project scheduling in human resource management. We consider the problem of scheduling assessment centers. In an assessment center, candidates for job positions perform different tasks while being observed and evaluated by assessors. Because these assessors are highly qualified and expensive personnel, the duration of the assessment center should be minimized. Complex rules for assigning assessors to candidates distinguish this problem from other scheduling problems discussed in the literature. We develop two discrete-time and three continuous-time MIP formulations, and we present problem-specific lower bounds. In a comparative study, we analyze the performance of the five MIP formulations on four real-life instances and a set of 240 instances derived from real-life data. The results indicate that good or optimal solutions are obtained for all instances within short computational time. In particular, one of the real-life instances is solved to optimality. Surprisingly, the continuous-time formulations outperform the discrete-time formulations in terms of solution quality.

2.1 Introduction

Over the past decades, mixed-integer linear programming (MIP) methods have been significantly improved (cf., e.g., Koch et al., 2011; Bixby, 2012) and successfully applied to a large variety of real-life scheduling problems in manufacturing and services. Two

major advantages of MIP methods are the flexibility to account for changes in the problem setting and the possibility to obtain upper or lower bounds on the solutions. In general, different formulations can be used to model the same planning problem. Because the performance of MIP approaches is determined by the underlying formulation (cf., e.g., Vielma, 2015), alternative formulations should be considered for each planning problem.

In this paper, we investigate an assessment center planning problem (ACP). This problem was reported to us by a human resource management service provider that organizes assessment centers (AC) for firms. The goal of an AC is to evaluate some candidates' job-related skills and abilities for one or several open positions (cf., e.g., Collins et al., 2003). In an AC, each candidate performs multiple tasks, and for each task, a prescribed number of assessors (i.e., psychologists or managers) is required. Some tasks involve role play and additionally require a prescribed number of actors. For example, the actors might represent unhappy customers with whom the candidate must interact. Tasks sometimes require a preparation time during which only the candidate is present. During the execution of the task, the candidate is joined by the assessors and the actor. Some tasks include a subsequent evaluation during which the assessors and the actors discuss their observations. This evaluation time can differ between assessors and actors. Each candidate takes a lunch break within a prescribed time window. When assigning assessors to tasks, the following rules must be considered: each candidate should be observed by approximately half the number of assessors; if a candidate and an assessor know each other personally, no observation is allowed, which is called a no-go relationship. Assessors are expensive, and hence, their total waiting time should be minimized. Because the assessors meet before the start and after the completion of all tasks and lunch breaks, this objective corresponds to minimizing the total duration of the AC (in what follows the AC duration for short). The planning problem consists of (1) scheduling all tasks and a lunch break for each candidate and (2) determining which assessors are assigned to which candidate during which task such that the AC duration is minimized.

The ACP can be interpreted as an extension of the resource-constrained project scheduling problem (RCPSP). The RCPSP consists of scheduling a set of activities subject to completion-start precedence and renewable-resource constraints such that the project duration is minimized. For the ACP, each candidate's tasks and lunch break correspond to project activities, and the candidates, assessors, and actors represent renewable resources. However, the ACP does not involve precedence relationships among the activities, but the above-described additional constraints. In the literature, different MIP formulations have been proposed for the RCPSP. In discrete-time (DT) formulations, the planning horizon is divided into a set of time intervals of equal length, and the activities can only start or end at the endpoints of these intervals. Conversely, in continuous-time (CT) formulations, the activities can start at any point in time. The DT formulations usually involve binary time-indexed variables. However, the meaning of these variables differ between the formulations, e.g., so-called pulse variables indicate whether an activity starts or ends at a specific point in time (cf. Pritsker et al., 1969; Christofides et al., 1987; Kopanos et al., 2014), and on/off variables specify whether an activity is in progress at a given time (cf. Kaplan, 1988; Mingozzi et al., 1998; Kopanos et al., 2014). The CT formulations differ with regard to the modeling of the resource constraints, e.g., Artigues et al. (2003) use resource-flow variables, and Kopanos et al. (2014) use overlapping variables. For a comprehensive overview of different MIP formulations for the RCPSP, we refer to Artigues et al. (2015).

In this paper, we provide two DT formulations and three CT formulations for the ACP. The two DT formulations are based on pulse variables (DT–P) and on/off variables (DT–O), respectively. The three CT formulation use assessor-assignment variables (CT–A), resource-flow variables (CT–F), and overlapping variables (CT–O), respectively, to model the resource constraints. Moreover, we provide problem-specific lower bounds. The different MIP formulations are tested on four real-life instances and 240 test instances based on real-life data. For all instances, good or optimal solutions are obtained within short computational time. In detail, formulation CT–A consistently outperforms the other

four formulations in terms of solution quality. However, using DT–P, the best MIP-based lower bounds are obtained. Furthermore, only with DT–P, optimality is proven for one of the real-life instances within the prescribed time limit. Nevertheless, in contrast to the RCPSP, the CT formulations provide better solutions than the DT formulations.

The remainder of this paper is structured as follows. In Section 2.2, we describe the ACP using an illustrative example and relate the ACP to the RCPSP. In Section 2.3, we provide an overview of the related literature. In Section 2.4, we present the MIP formulations for the ACP. In Section 2.5, we derive the problem-specific lower bounds. In Section 2.6, we discuss the design and the results of our comparative analysis. In Section 2.7, we provide some concluding remarks and an outlook on future research.

2.2 Planning problem

In Section 2.2.1, we describe the problem features of the ACP in detail and illustrate them through an example. In Section 2.2.2, we discuss the relation between the ACP and the RCPSP.

2.2.1 Illustration of the planning problem

In our illustrative example, the participants of the AC are as follows: there are three candidates, C_1, C_2 and C_3; four assessors, A_1, A_2, A_3 and A_4; and an actor, P_1. A no-go relationship exists between candidate C_3 and assessor A_2. Each of the three candidates must perform the three tasks E_1, E_2, and E_3, and take a lunch break.

The tasks of the illustrative example are listed in Table 2.1. The durations of the tasks are stated in 5-minute time units. Tasks E_1 and E_3 require two assessors, and task E_2 requires one assessor. Task E_1 involves role play and requires one actor. Tasks E_1 and E_2 include a preparation time, and tasks E_1 and E_3 include an evaluation time. Figure 2.1 shows at which time during the execution of task E_1 the candidate, the assessors, and

the actor are required. The evaluation time differs between the assessors and the actor. Due to fairness and objectivity considerations, no waiting times are allowed between the preparation, the execution, and the evaluation. A waiting time for a candidate would increase the preparation time, whereas a waiting time for the assessors and actors could bias their evaluations of the candidate.

The earliest and latest possible start times for the lunch break are 20 and 30, respectively. The duration of the lunch break is 6 time units. Because each candidate has a lunch break and performs each of the three tasks exactly once, a total of 12 activities are considered. Table 2.2 shows the indices of these activities.

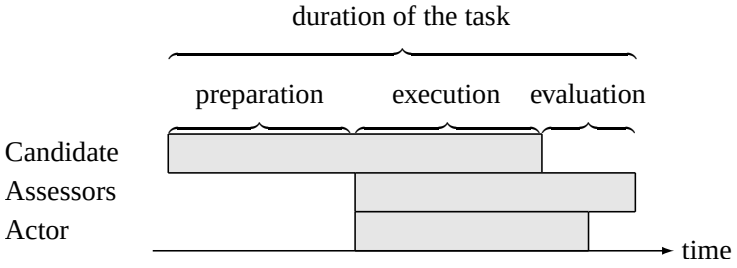


Figure 2.1: Varying requirements for candidate, assessors, and actor during task E_1

Table 2.1: Tasks of illustrative example

Task	E_1	E_2	E_3
Required number of assessors	2	1	2
Required number of actors	1	-	-
Duration	20	10	12
Duration of preparation time (candidates)	8	3	-
Duration of execution time	8	7	8
Duration of evaluation time (assessors)	4	-	4
Duration of evaluation time (actors)	2	-	-

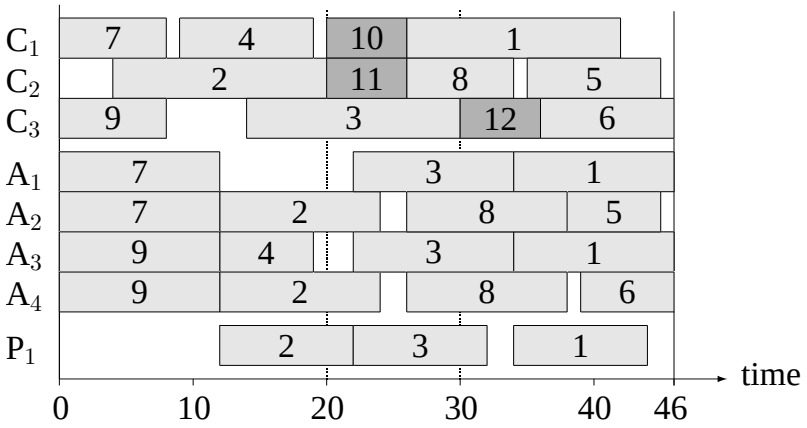
The rules for assigning assessors to candidates are as follows: each candidate should be observed by at least half of the total number of assessors rounded down and by at most half of the total number of assessors rounded up plus one. The lower limit ensures an objective overall evaluation for each candidate, and the upper limit is motivated by

fairness considerations. The difference between the upper and lower limits facilitates the assessor assignment without affecting fairness. The number of times that an assessor can observe the same candidate is not limited. In the illustrative example, each candidate must be observed by 2 to 3 different assessors. Additionally, because a no-go relationship exists, candidate C_3 can never be observed by assessor A_2.

An optimal schedule for the illustrative example is presented in Figure 2.2. The dotted lines indicate the earliest and latest start times for the lunch breaks, and the solid line indicates the AC duration. Whether an assessor has been assigned to a candidate at least once is indicated by a checkmark (✓).

Table 2.2: Activity indices of the illustrative example

Candidate	Task			Lunch
	E_1	E_2	E_3	break
C_1	1	4	7	10
C_2	2	5	8	11
C_3	3	6	9	12



Assignments			
	Candidate		
Assessor	C_1	C_2	C_3
A_1	✓		✓
A_2	✓	✓	
A_3	✓		✓
A_4		✓	✓

Figure 2.2: Optimal schedule of the illustrative example (left) and corresponding assessor assignment (right)

2.2.2 Relation to the RCPSP

The ACP includes many problem features of the well-known RCPSP. Both planning problems consider activities that require prescribed amounts of some renewable resources during their execution. In the case of the ACP, the execution of each task and the lunch break for each candidate correspond to a project activity, and the candidates, assessors, and actors can be interpreted as renewable resources. The ACP does not involve precedence relationships among the activities.

In the RCPSP, only the capacities and not the individual units of the renewable resources are considered. However, in the ACP, the assessor-assignment rules require that all activities that use a particular resource unit can be identified. Therefore, the assessor-assignment rules cannot be formulated in the RCPSP.

If each assessor is interpreted as a renewable resource with unit capacity, then alternative execution modes must be defined in order to represent the alternative assessor assignments. This corresponds to the multi-mode extension of the RCPSP (MRCPSP). Because each candidate must be observed by approximately half the number of the assessors, the assessor assignments interdepend. Such interdependencies between modes are not considered in the MRCPSP. Before assigning any assessors to a candidate, all modes are feasible. However, selecting the modes for some activities causes several of the modes of the other activities to be infeasible.

2.3 Literature review

In Section 2.3.1, we provide an overview of different MIP formulations for the RCPSP which can be used as the basis for MIP formulations of the ACP. In Section 2.3.2, we discuss recent works that focus on comparing MIP formulations for extensions of the RCPSP and for specific real-life problems.

2.3.1 MIP formulations for the RCPSP

In DT formulations, binary time-indexed variables are used that indicate the start, end, or the state (e.g., in progress) of an activity at a specific time. For DT formulations, three types of binary variables can be distinguished (cf. Artigues et al., 2015). Beside the pulse and on/off variables described in the Introduction, there are step variables that indicate whether an activity starts at or before a specific point in time (cf. Klein, 2000; Bianco and Caramia, 2013). Furthermore, Bianco and Caramia (2013) introduce continuous variables that specify the percentage of completion of the activities at each point in time.

In CT formulations, the activities can start or finish at any time rather than at predefined time points such as in DT. Artigues et al. (2003) present a CT formulation based on resource flows. Besides the continuous start-time variables, this formulation requires two additional sets of variables. The first set consists of binary sequencing variables that determine for each pair of activities whether one precedes the other or whether both are executed in parallel. The second set consists of continuous resource-flow variables for modeling the resource constraints. Kopanos et al. (2014) present another CT formulation with continuous start-time variables, binary sequencing variables, and binary overlapping variables. In combination with the sequencing variables, the overlapping variables are used to model the resource constraints. Other CT formulations are based on events (e.g. Koné et al., 2011) or on minimal forbidden sets (e.g. Alvarez-Valdes and Tamarit, 1993).

For the RCPSP, the performances of these different MIP formulations are compared in Bianco and Caramia (2013), Koné et al. (2011), and Kopanos et al. (2014). They all use generic test instances, which are provided in, e.g., Kolisch and Sprecher (1997) and Vanhoucke et al. (2008). For these test instances, Koné et al. (2011) and Kopanos et al. (2014) show that the performance is primarily affected by the number of activities and the length of the planning horizon. The performances of the DT formulations are negatively affected by the length of the planning horizon because the numbers of variables and constraints depend on the number of time points considered. In contrast, the

performances of the CT formulations are negatively affected by the number of activities because the number of sequencing variables increases exponentially with the number of activities. Typically, DT-based formulations are the most competitive and yield the best LP relaxations. However, no formulation consistently dominates the others, as different formulations perform better for different problem settings.

In this study, we adapt different RCPSP formulations such that they can be applied to the ACP. From the DT formulations, we select the RCPSP formulations of Pritsker et al. (1969) and Kopanos et al. (2014). The basic DT formulation of Pritsker et al. (1969) still performs very well compared to newer formulations (cf., e.g., Koné et al., 2011). Kopanos et al. (2014) show that their two DT formulations outperform other DT formulations presented in the literature. Their DT formulations differ with regard to the modeling of the precedence constraints. For the ACP, these two formulations are identical because there are no precedence constraints. From the CT formulations, we adapt the formulations of Artigues et al. (2003) and Kopanos et al. (2014). The CT formulation of Artigues et al. (2003) performs well compared to other CT formulations if there are specific problem characteristics such as long activity durations (cf., e.g., Koné et al., 2011). Kopanos et al. (2014) show that their two CT formulations outperform other CT formulations presented in the literature; we adapted their best-performing CT formulation.

2.3.2 Comparative studies of MIP formulations

In addition to the aforementioned comparative studies of the RCPSP, the performances of alternative MIP formulations have also been compared for various other planning problems. In the following, we provide an overview of such comparative studies for extensions of the RCPSP and for some real-life problems.

Some extensions of the RCPSP for which alternative MIP formulations have been compared are as follows. In Koné et al. (2013), the performances of alternative DT and CT formulations are compared for an extension of the RCPSP with so-called storage resources.

Storage resources are consumed and produced at the project activities' start times and completion times, respectively. As in Koné et al. (2011), the authors conclude that no MIP formulation consistently yields the best results. A comparative performance analysis of alternative DT formulations for the RCPSP with flexible resource profiles is provided in Naber and Kolisch (2014). With flexible resource profiles, the resource utilization of an activity is not constant but rather can be adjusted from period to period. The results of the comparative study in Naber and Kolisch (2014) indicate that an MIP formulation based on Bianco and Caramia (2013) dominates all other DT formulations. In the study of Zapata et al. (2008), alternative DT and CT formulations for the MRCPSP with multiple projects are compared. The authors conclude that the best MIP formulation depends on the specific characteristics of each problem instance.

Comparative analyses have also been conducted for MIP formulations in real-life applications. Stefansson et al. (2011) develop DT and CT formulations for a large-scale production scheduling problem originating from a pharmaceutical producer. In this problem, customers order specific products, which need to be produced in a four-stage production process such that the requested quantity and delivery date of the order are met. The results obtained for eight test instances indicate that the CT formulation obtains better solutions within shorter computational time than the DT formulation. Furthermore, in Chen et al. (2012), a comparative analysis of different mixed-integer nonlinear programming formulations for the scheduling of crude-oil refinement operations is presented. The planning problem includes several processing steps, from unloading marine vessels to producing various crude-oil based products. In a recent study, Ambrosino et al. (2015) evaluated the performance of two alternative MIP formulations for the multi-port master bay plan problem. This problem involves the placement of containers on a containership such that the overall berthing costs of the ship's multi-port journey are minimized.

2.4 MIP formulations for the ACP

In this section, we present our five MIP formulations for the ACP. The notation of the MIP formulations is provided in Tables 2.3 and 2.4. In Section 2.4.1, we present the CT formulation that uses the assessor-assignment decisions to model the resource constraints (CT–A). In Section 2.4.2, we derive the CT formulation with resource-flow variables (CT–F). In Section 2.4.3, we present the CT formulation with overlapping variables (CT–O). In Sections 2.4.4 and 2.4.5, we present the DT formulation with pulse variables (DT–P) and the DT formulation with on/off variables (DT–O), respectively.

Table 2.3: Sets and parameters of the MIP formulations

C	Set of candidates
A	Set of assessors
P	Set of actors
N	Set of candidate-assessor pairs (c, a) with a no-go relationship
I	Set of activities $i = 1, \ldots, n$ (including lunch breaks)
I_c	Set of activities that require candidate $c \in C$
I^A, I^P	Set of activities that require assessors (I^A) and actors (I^P)
I^L	Set of lunch breaks
ES^L, LS^L	Earliest (ES^L) and latest (LS^L) start time for the lunch breaks
p_i	Duration of activity i
p_i^C	Preparation time of activity i for candidates
p_i^A, p_i^P	Evaluation time of activity i for assessors (p_i^A) and actors (p_i^P)
r_i^A, r_i^P	Number of assessors (r_i^A) and actors (r_i^P) required by activity i
M	Sufficiently large number
T	Upper bound on the duration of the assessment center

2.4.1 Formulation CT–A

In this section, we present the continuous-time formulation that uses the assessor-assignment decisions to model the resource constraints (CT–A). In a preliminary version

of this MIP formulation (cf. Grüter et al., 2014), each activity is split into several sub-activities to model the preparation, the execution, and the evaluation times. However, this results in an unnecessary large number of variables and constraints. In the following, we model the ACP without splitting the activities.

We distinguish between three types of resources: candidates, assessors, and actors. Each candidate is modeled as a renewable resource with capacity 1. The set of all assessors (actors) is modeled as one renewable resource with a capacity that equals the number of assessors (actors). Due to the capacity of 1, the resource constraints for the candidates are modeled using binary sequencing variables, i.e., $Y_{ij}^C = 1$ ($Y_{ij}^C = 0$) if activity i (j) is completed some time before the start of activity j (i) by the corresponding candidate. For the assessors and actors, the resource constraints are modeled using binary sequencing variables (Y_{ij}^A and Y_{ij}^P), and binary assignment variables (Z_{ia}^A and Z_{ip}^P). For the assessors, the sequencing variable Y_{ij}^A is equal to 1 if activity i is completed some time before the start of activity j. Otherwise, Y_{ij}^A is 0, i.e., activities i and j are processed simultaneously or j finishes some time before i begins. Because the ACP does not include precedence relationships, there are no prescribed values for the sequencing variables. The assignment variable Z_{ia}^A is equal to 1 if assessor a is assigned to activity i; otherwise $Z_{ia}^A = 0$. For the actors, the sequencing and assignment variables (Y_{ij}^P and Z_{ip}^P) are interpreted in the same

Table 2.4: Variables of the MIP formulations

D	AC duration
S_i	Start time of activity i for the candidate
X_{it}	$\begin{cases} = 1, & \text{if activity } i \text{ starts at time point } t; \\ = 0, & \text{otherwise.} \end{cases}$
Y_{ij}^C	$\begin{cases} = 1, & \text{if activity } i \text{ is performed before } j > i \text{ by a candidate;} \\ = 0, & \text{otherwise.} \end{cases}$
Y_{ij}^A	$\begin{cases} = 1, & \text{if activity } i \text{ is performed before } j \neq i \text{ by the assessors;} \\ = 0, & \text{otherwise.} \end{cases}$
Y_{ij}^P	$\begin{cases} = 1, & \text{if activity } i \text{ is performed before } j \neq i \text{ by the actors;} \\ = 0, & \text{otherwise.} \end{cases}$
Z_{ia}^A	$\begin{cases} = 1, & \text{if assessor } a \text{ is assigned to activity } i; \\ = 0, & \text{otherwise.} \end{cases}$
Z_{ip}^P	$\begin{cases} = 1, & \text{if actor } p \text{ is assigned to activity } i; \\ = 0, & \text{otherwise.} \end{cases}$
V_{ca}	$\begin{cases} = 1, & \text{if assessor } a \text{ is assigned to candidate } c \text{ at least once;} \\ = 0, & \text{otherwise.} \end{cases}$
F_{ij}^C	$\begin{cases} = 1, & \text{if a candidate is sent from activity } i \text{ to } j; \\ = 0, & \text{otherwise.} \end{cases}$
F_{ij}^A	Number of assessors sent from activity i to j
F_{ij}^P	Number of actors sent from activity i to j
$\widehat{Y}_{ij}$	$\begin{cases} = 1, & \text{if activity } i \text{ starts before or at the same time as } j \text{ for assessors;} \\ = 0, & \text{otherwise.} \end{cases}$
O_{ji}^A	$\begin{cases} = 1, \text{if activity } j \text{ finishes after the start of activity } i \text{ for assessors;} \\ = 0 \text{ or } 1, \text{otherwise.} \end{cases}$
O_{ji}^P	$\begin{cases} = 1, \text{if activity } j \text{ finishes after the start of activity } i \text{ for actors;} \\ = 0 \text{ or } 1, \text{otherwise.} \end{cases}$
W_{it}	$\begin{cases} = 1, & \text{if } i \text{ is processed at time } t \text{ by the candidates;} \\ = 0, & \text{otherwise.} \end{cases}$

way. Finally, variable V_{ca} is used to model the assessor-assignment rule, i.e., $V_{ca} = 1$ if assessor a is assigned to candidate c at least once.

The objective is to minimize the AC duration D.

$$\text{Min } D$$

The duration corresponds to the latest completion time of an activity that is defined by constraints (2.1).

$$D \geq S_i + p_i \qquad (i \in I) \tag{2.1}$$

Constraints (2.2)–(2.5) determine the resource-feasible start times of the activities. Constraints (2.2) are binding if candidate c completes activity i before the start of activity j. Otherwise, constraints (2.3) are binding. Because candidate c is not required during the evaluation time, activity j can start at most p_i^A time units before the completion of activity i (cf. Figure 2.3).

$$S_j \geq S_i - M + (p_i - p_i^A + M)Y_{ij}^C \qquad (c \in C, i, j \in I_c : i < j) \tag{2.2}$$

$$S_i \geq S_j - M + (p_j - p_j^A + M)(1 - Y_{ij}^C) \qquad (c \in C, i, j \in I_c : i < j) \tag{2.3}$$

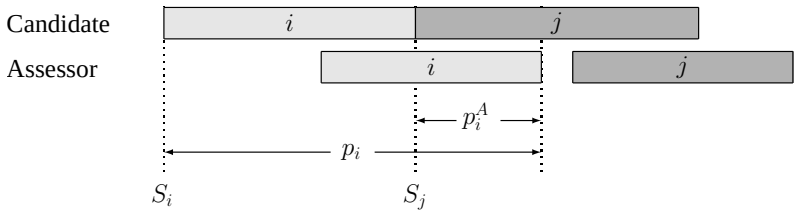


Figure 2.3: Minimum time lag between start times of activities i and j for candidates

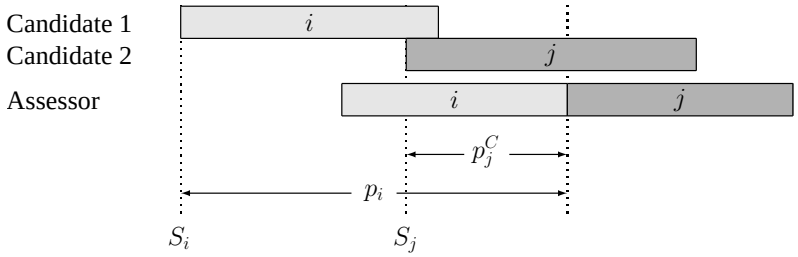


Figure 2.4: Minimum time lag between start times of activities i and j for assessors

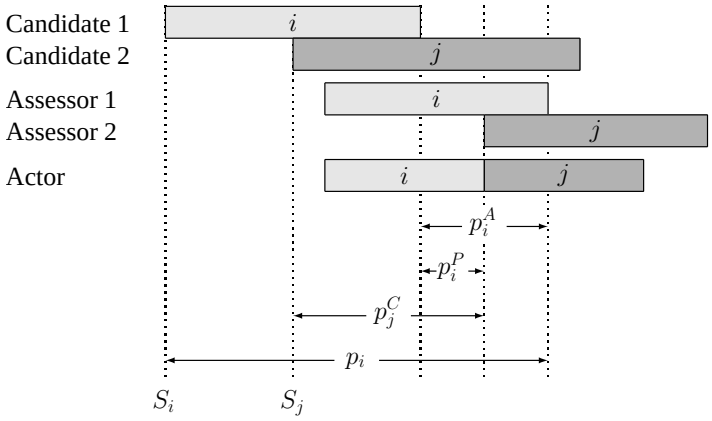


Figure 2.5: Minimum time lag between start times of activities i and j for actors

Constraints (2.4) and (2.5) enforce a sequence of activities for the assessors and actors, respectively. In the case that activity i is executed before activity j by the assessors, constraints (2.4) are binding. Because the assessors are not required during the preparation time, activity j can start at most p_j^C time units before the completion of activity i (cf. Figure 2.4). Similarly, constraints (2.5) are binding if activity i is executed before activity j by the actors. For the actors, activity i is completed after $p_i - p_i^A + p_i^P$ time units. Activity j can start at most p_j^C time units before that completion time (cf. Figure 2.5).

$$S_j \geq S_i - M + (p_i - p_j^C + M)Y_{ij}^A \qquad (i, j \in I^A : i \neq j) \qquad (2.4)$$

$$S_j \geq S_i - M + (p_i - p_i^A + p_i^P - p_j^C + M)Y_{ij}^P \qquad (i, j \in I^P : i \neq j) \qquad (2.5)$$

Constraints (2.6) ensure that the lunch breaks are scheduled within the prescribed time window.

$$ES^L \leq S_i \leq LS^L \qquad (i \in I^L) \tag{2.6}$$

Constraints (2.7) and (2.8) imply that the required numbers of assessors and actors are assigned to each activity.

$$\sum_{a \in A} Z^A_{ia} = r^A_i \qquad (i \in I^A) \tag{2.7}$$

$$\sum_{p \in P} Z^P_{ip} = r^P_i \qquad (i \in I^P) \tag{2.8}$$

Constraints (2.9) and (2.10) link the assignment variables to the sequencing variables. If the same assessor a or the same actor p is assigned to two activities i and j, then a sequence between these two activities is enforced.

$$Y^A_{ij} + Y^A_{ji} \geq Z^A_{ia} + Z^A_{ja} - 1 \qquad (i, j \in I^A, a \in A : i < j) \tag{2.9}$$

$$Y^P_{ij} + Y^P_{ji} \geq Z^P_{ip} + Z^P_{jp} - 1 \qquad (i, j \in I^P, p \in P : i < j) \tag{2.10}$$

Constraints (2.11) and (2.12) ensure that either activity i precedes activity j, j precedes i, or i and j are processed in parallel.

$$Y^A_{ij} + Y^A_{ji} \leq 1 \qquad (i, j \in I^A : i < j) \tag{2.11}$$

$$Y^P_{ij} + Y^P_{ji} \leq 1 \qquad (i, j \in I^P : i < j) \tag{2.12}$$

Constraints (2.13) enforce that the number of assessors assigned to each candidate lies within the bounds imposed by the assessor-assignment rule.

$$\left\lfloor \frac{|A|}{2} \right\rfloor \leq \sum_{a \in A} V_{ca} \leq \left\lceil \frac{|A|}{2} \right\rceil + 1 \qquad (c \in C) \tag{2.13}$$

Constraints (2.14) determine whether an assessor a has been assigned to a candidate c at least once. V_{ca} must be equal to 1 if assessor a is assigned to at least one activity that requires candidate c. If assessor a is never assigned to an activity that requires candidate c, then V_{ca} must be equal to 0.

$$\sum_{i \in I_c \setminus I^L} \frac{Z_{ia}^A}{|I_c \setminus I^L|} \leq V_{ca} \leq \sum_{i \in I_c \setminus I^L} Z_{ia}^A \qquad (c \in C, a \in A) \tag{2.14}$$

Finally, constraints (2.15) model the no-go relationships.

$$V_{ca} = 0 \qquad ((c,a) \in N) \tag{2.15}$$

In sum, formulation (CT–A) reads as follows:

$$\text{(CT–A)} \quad \begin{cases} \text{Min } D & \\ \text{s.t. } (2.1)\text{–}(2.15) & \\ \quad S_i \geq 0 & (i \in I) \\ \quad Y_{ij}^C \in \{0,1\} & (c \in C,\ i,j \in I_c :\ i < j) \\ \quad Y_{ij}^A \in \{0,1\} & (i,j \in I^A :\ i \neq j) \\ \quad Y_{ij}^P \in \{0,1\} & (i,j \in I^P :\ i \neq j) \\ \quad V_{ca} \in \{0,1\} & (c \in C,\ a \in A) \\ \quad Z_{ia}^A \in \{0,1\} & (i \in I^A,\ a \in A) \\ \quad Z_{ip}^P \in \{0,1\} & (i \in I^P,\ p \in P) \end{cases}$$

2.4.2 Formulation CT–F

In this section, we present the continuous-time formulation with resource-flow variables (CT–F), which is based on the RCPSP formulation of Artigues et al. (2003). This MIP formulation was first proposed in Zimmermann and Trautmann (2014). The following

explanations closely follow that study.

To model the resource flows, formulation CT–F requires the dummy activities 0 and $n+1$; both have a duration of zero, and $r_0^A = r_{n+1}^A = |A|$ ($r_0^P = r_{n+1}^P = |P|$) is equal to the total number of available assessors (actors). Variable F_{ij}^C (F_{ij}^A, F_{ij}^P) denotes the quantity of candidates (assessors, actors) sent from activity i (upon completion) to activity j (at the beginning). This resource flow prevents the corresponding activities from being executed simultaneously. For the assessors (actors), the sequencing variable Y_{ij}^A (Y_{ij}^P) is equal to 1 if some assessors (actors) are sent from activity i to activity j. Because each activity requires exactly one candidate, any flow of candidates between two activities will be either 0 or 1. Since the resource-flow variable F_{ij}^C is defined as binary, this variable is used simultaneously as a resource-flow and as a sequencing variable. As a sequencing variable, F_{ij}^C equals 1 if and only if activity j is executed after activity i.

The following constraints have to be considered. Constraints (2.16) determine resource-feasible start times of the activities for the candidates. The feasible start times of the activities for the assessors and actors are determined as in formulation CT–A. Constraints (2.16) are binding if a candidate is sent from activity i to activity j ($F_{ij}^C = 1$).

$$S_j \geq S_i - M + (p_i - p_i^A + M)F_{ij}^C \qquad (c \in C;\ i,j \in I_c :\ i \neq j) \qquad (2.16)$$

Constraints (2.17)–(2.22) are the resource-flow conservation constraints. Constraints (2.17) ensure that each activity i sends 1 unit of resource $c \in C$ to either an activity $j \neq i$ or the dummy activity $n+1$ (if activity i is the last activity performed by candidate c). Constraints (2.18) ensure that each activity j receives 1 unit of resource $c \in C$ from either an activity $i \neq j$ or the dummy activity 0 (if activity j is the first activity performed by

candidate c).

$$\sum_{j \in I_c \cup \{n+1\}:\ j \neq i} F_{ij}^C = 1 \qquad (c \in C;\ i \in I_c \cup \{0\}) \tag{2.17}$$

$$\sum_{i \in I_c \cup \{0\}:\ i \neq j} F_{ij}^C = 1 \qquad (c \in C;\ j \in I_c \cup \{n+1\}) \tag{2.18}$$

Constraints (2.19)–(2.22) conserve the resource flow of assessors and actors, respectively. The number of assessors r_i^A (actors r_i^P) required by activity i must be sent to and received from other activities that require the same resource.

$$\sum_{j \in I^A \cup \{n+1\}:\ j \neq i} F_{ij}^A = r_i^A \qquad (i \in I^A \cup \{0\}) \tag{2.19}$$

$$\sum_{j \in I^P \cup \{n+1\}:\ j \neq i} F_{ij}^P = r_i^P \qquad (i \in I^P \cup \{0\}) \tag{2.20}$$

$$\sum_{i \in I^A \cup \{0\}:\ i \neq j} F_{ij}^A = r_j^A \qquad (j \in I^A \cup \{n+1\}) \tag{2.21}$$

$$\sum_{i \in I^P \cup \{0\}:\ i \neq j} F_{ij}^P = r_j^P \qquad (j \in I^P \cup \{n+1\}) \tag{2.22}$$

Constraints (2.23) and (2.24) link the resource-flow variables to the sequencing variables for assessors and actors, respectively.

$$F_{ij}^A \leq \min(r_i^A, r_j^A) Y_{ij}^A \qquad (i, j \in I^A :\ i \neq j) \tag{2.23}$$

$$F_{ij}^P \leq \min(r_i^P, r_j^P) Y_{ij}^P \qquad (i, j \in I^P :\ i \neq j) \tag{2.24}$$

The sequencing variables Y_{ij}^A and Y_{ij}^P are only used to link the flow variables F_{ij}^A and F_{ij}^P to the start times of the activities. The flow variables F_{ij}^A and F_{ij}^P can be greater than 1. For this reason, they cannot be used as sequencing variables.

Constraints (2.1), which determine the AC duration D, and the sequencing constraints for the assessors (2.4) and actors (2.5), and constraints (2.6), which specify the time window

for the lunch breaks, are also included. The same applies to the assessor-assignment constraints (2.7), (2.9), and (2.11)–(2.15).

In sum, formulation (CT–F) reads as follows:

$$
\text{(CT–F)} \quad \begin{cases}
\text{Min } D & \\
\text{s.t. } (2.16)\text{–}(2.24) & \\
\quad (2.1), (2.4)\text{–}(2.7), (2.9) & \\
\quad (2.11)\text{–}(2.15) & \\
\quad S_i \geq 0 & (i \in I) \\
\quad F_{ij}^C \in \{0,1\} & (c \in C;\ i,j \in I_c \cup \{0, n+1\} :\ i \neq j) \\
\quad F_{ij}^A \geq 0 & (i,j \in I^A \cup \{0, n+1\} :\ i \neq j) \\
\quad F_{ij}^P \geq 0 & (i,j \in I^P \cup \{0, n+1\} :\ i \neq j) \\
\quad Y_{ij}^A \in \{0,1\} & (i,j \in I^A :\ i \neq j) \\
\quad Y_{ij}^P \in \{0,1\} & (i,j \in I^P :\ i \neq j) \\
\quad V_{ca} \in \{0,1\} & (c \in C,\ a \in A) \\
\quad Z_{ia}^A \in \{0,1\} & (i \in I^A,\ a \in A)
\end{cases}
$$

2.4.3 Formulation CT–O

In this section, we present the continuous-time formulation with overlapping variables (CT–O), which is based on the RCPSP formulation of Kopanos et al. (2014).

For activities that cannot be processed in parallel (i.e., two activities which require the same candidate), we use the sequencing variables Y_{ij}^C. For activities that can be processed in parallel, the resource constraints are modeled with the following binary variables.

- For the assessors and the actors, we introduce the sequencing variables $\widehat{Y}_{ij}$. Specifically, $\widehat{Y}_{ij} = 1$ if activity i starts before or at the same time as activity j for the assessors. These sequencing variables are not defined separately for assessors and

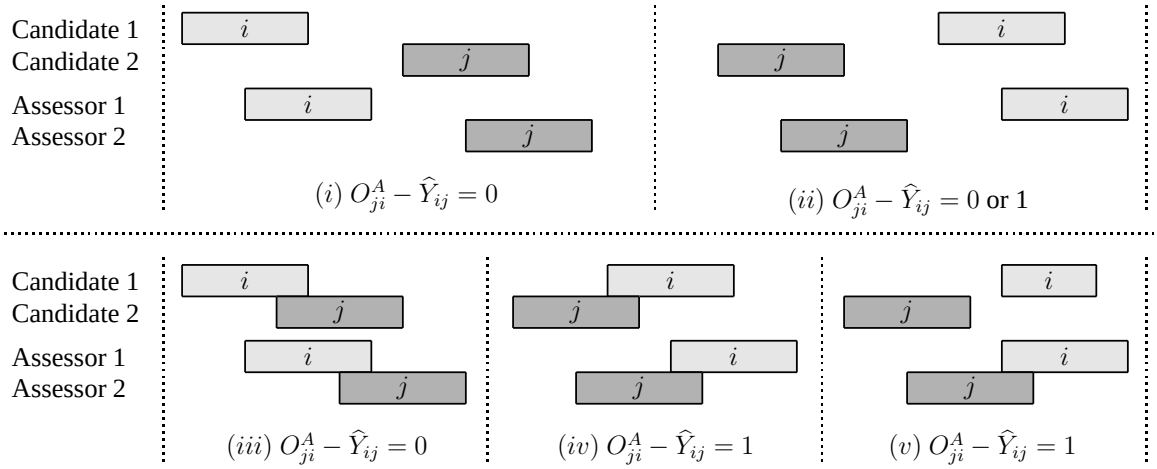


Figure 2.6: Five possible cases (i)–(v) that illustrate the values of the sequencing and overlapping variables

actors, because the activities start at the same time for them.

- For the assessors, we introduce the overlapping variables O^A_{ji}. Specifically, $O^A_{ji} = 1$ if activity j finishes after the start of activity i for the assessors. If activity j finishes before or at the same time as activity i starts, then O^A_{ji} is equal to 0 or 1. The overlapping variables for the actors O^P_{ji} are defined in the same way.

To illustrate how these variables jointly determine whether two activities $i, j \in I^A$ are processed in parallel by the assessors, several possible cases are depicted in Figure 2.6. For case (ii), the variable O^A_{ji} can be equal to zero or one, but for cases (iv) and (v), the variable must be equal to one.

Constraints (2.25) determine the resource-feasible start times of the activities for the candidates. Constraints (2.26) ensure that either activity i precedes activity j, or j precedes i. In contrast to constraints (2.2) and (2.3), the sequencing variables Y^C_{ij} are used for any pair of activities involving the same candidate.

$$S_i + p_i - p^A_i \leq S_j + MY^C_{ji} \qquad (c \in C, i, j \in I_c : i \neq j) \qquad (2.25)$$

$$Y^C_{ij} + Y^C_{ji} = 1 \qquad (c \in C, i, j \in I_c : i > j) \qquad (2.26)$$

Constraints (2.27)–(2.29) determine the resource-feasible start times of the activities which can be processed in parallel. Thereby, parameter λ is used to exclude some symmetric solutions, i.e., for two activities $i > j$ which start at the same time, it is specified that $\widehat{Y}_{ji} = 1$ and $\widehat{Y}_{ij} = 0$. As proposed in Kopanos et al. (2014), we set $\lambda = 0.1$.

$$S_j + p_j^C \leq S_i + p_i^C + M\widehat{Y}_{ij} \qquad (i, j \in I^A : i > j) \tag{2.27}$$

$$S_i + p_i^C + \lambda \leq S_j + p_j^C + (M + \lambda)\widehat{Y}_{ji} \qquad (i, j \in I^A : i > j) \tag{2.28}$$

$$\widehat{Y}_{ij} + \widehat{Y}_{ji} = 1 \qquad (i, j \in I^A : i > j) \tag{2.29}$$

Constraints (2.30) and (2.31) link the overlapping variables to the start times of the activities.

$$(S_j + p_j) - (S_i + p_i^C) \leq MO_{ji}^A \qquad (i, j \in I^A : i \neq j) \tag{2.30}$$

$$(S_j + p_j - p_j^A + p_j^P) - (S_i + p_i^C) \leq MO_{ji}^P \qquad (i, j \in I^P : i \neq j) \tag{2.31}$$

Constraints (2.32) and (2.33) ensure that all activities that are executed in parallel do not require more than the available number of assessors and actors, respectively. Thereby, the term $O_{ji}^A - \widehat{Y}_{ij} = 1$ if activity j starts before activity i and if both activities overlap for the assessors. The same applies to the actors.

$$r_i^A + \sum_{j \in I^A : j \neq i} r_j^A (O_{ji}^A - \widehat{Y}_{ij}) \leq |A| \qquad (i \in I^A) \tag{2.32}$$

$$r_i^P + \sum_{j \in I^P : j \neq i} r_j^P (O_{ji}^P - \widehat{Y}_{ij}) \leq |P| \qquad (i \in I^P) \tag{2.33}$$

Constraints (2.34) and (2.35) ensure that the terms $O^A_{ji} - \widehat{Y}_{ij}$ and $O^P_{ji} - \widehat{Y}_{ij}$ are greater than or equal to zero.

$$\widehat{Y}_{ij} \leq O^A_{ji} \qquad (i,j \in I^A : i \neq j) \tag{2.34}$$

$$\widehat{Y}_{ij} \leq O^P_{ji} \qquad (i,j \in I^P : i \neq j) \tag{2.35}$$

Constraints (2.36) link the sequencing and overlapping variables to the assignment variables. If the same assessor a is assigned to two activities i and j, then both activities cannot overlap for the assessors.

$$(O^A_{ji} - \widehat{Y}_{ij}) + Z^A_{ia} + Z^A_{ja} \leq 2 \qquad (a \in A,\ i,j \in I^A : i \neq j) \tag{2.36}$$

Constraints (2.1), which determine the AC duration D, and constraints (2.6), which specify the time window for the lunch breaks, are also included. The same applies to the assessor-assignment constraints (2.13)–(2.15).

In sum, formulation (CT–O) reads as follows:

$$\text{(CT–O)} \quad \begin{cases} \text{Min } D & \\ \text{s.t. } (2.25)\text{–}(2.36) & \\ \quad (2.1), (2.6), (2.7), (2.13)\text{–}(2.15) & \\ \quad S_i \geq 0 & (i \in I) \\ \quad Y^C_{ij} \in \{0,1\} & (c \in C,\ i,j \in I_c : i \neq j) \\ \quad \widehat{Y}_{ij} \in \{0,1\} & (i,j \in I^A : i \neq j) \\ \quad O^A_{ji} \in \{0,1\} & (i,j \in I^A : i \neq j) \\ \quad O^P_{ji} \in \{0,1\} & (i,j \in I^P : i \neq j) \\ \quad V_{ca} \in \{0,1\} & (c \in C,\ a \in A) \\ \quad Z^A_{ia} \in \{0,1\} & (i \in I^A,\ a \in A) \end{cases}$$

2.4.4 Formulation DT–P

In this section, we present the discrete-time formulation with pulse variables (DT–P), which is based on the RCPSP formulation of Pritsker et al. (1969). This formulation involves the discretization of the planning horizon into uniform time intervals. The endpoints of a time interval are denoted by the time points t and $t+1$, respectively ($t = 0, \ldots, T-1$). Binary pulse variables X_{it} state if activity i starts at time t. For each time point t, resource constraints are formulated that ensure that the resource capacities are not violated. We extend the resource constraints of the RCPSP formulation such that the preparation and evaluation times of the AC activities are considered.

For the ACP, the following constraints have to be taken into consideration. The AC duration corresponds to the latest completion time of an activity, which is defined by constraints (2.37).

$$D \geq \sum_{t=0}^{T-p_i} (t + p_i) X_{it} \qquad (i \in I) \tag{2.37}$$

Constraints (2.38) and (2.39) ensure that each activity starts once. Furthermore, constraints (2.39) state that the lunch breaks are scheduled within the prescribed time window.

$$\sum_{t=0}^{T-p_i} X_{it} = 1 \qquad (i \in I \backslash I^L) \tag{2.38}$$

$$\sum_{t=ES^L}^{LS^L} X_{it} = 1 \qquad (i \in I^L) \tag{2.39}$$

Constraints (2.40) to (2.42) ensure that the resource capacities are not violated. Constraints (2.40) ensure that each candidate performs at most one activity at the same time t. Candidate c performs activity i at time t if the activity started between time $t-(p_i-p_i^A)+1$ and t. Constraints (2.41) and (2.42) ensure that all activities that are scheduled in parallel do not require more than the maximum available numbers of assessors and actors, respectively. An assessor performs activity i at time t if the activity started between time $t-p_i+1$ and $t-p_i^C$. An actor performs activity i at time t if the activity started between time $t-(p_i-p_i^A+p_i^P)+1$ and $t-p_i^C$.

$$\sum_{i\in I_c}\sum_{\tau=\max(0,t-p_i+p_i^A+1)}^{t} X_{i\tau} \leq 1 \qquad (c\in C, t=0,\ldots,T) \tag{2.40}$$

$$\sum_{i\in I^A}\sum_{\tau=\max(0,t-p_i+1)}^{t-p_i^C} r_i^A X_{i\tau} \leq |A| \qquad (t=0,\ldots,T) \tag{2.41}$$

$$\sum_{i\in I^P}\sum_{\tau=\max(0,t-p_i+p_i^A-p_i^P+1)}^{t-p_i^C} r_i^P X_{i\tau} \leq |P| \qquad (t=0,\ldots,T) \tag{2.42}$$

Additionally, the assessor-assignment constraints (2.7), (2.9), (2.11), and (2.13)–(2.15) are also included. Constraints (2.43) link the variables X_{it} to the sequencing variables Y_{ij}^A.

$$\sum_{t=0}^{T-p_j} tX_{jt} \geq \sum_{t=0}^{T-p_i} tX_{it} - M + (p_i - p_j^C + M)Y_{ij}^A \qquad (i,j\in I^A:\ i\neq j) \tag{2.43}$$

In sum, formulation (DT–P) reads as follows:

$$
\text{(DT–P)} \quad \begin{cases}
\text{Min } D & \\
\text{s.t. } (2.37)\text{–}(2.43) & \\
\quad (2.7), (2.9), (2.11), (2.13)\text{–}(2.15) & \\
\quad X_{it} \in \{0,1\} & (i \in I, t = 0, \ldots, T) \\
\quad Y^A_{ij} \in \{0,1\} & (i, j \in I^A : \ i \neq j) \\
\quad V_{ca} \in \{0,1\} & (c \in C, \ a \in A) \\
\quad Z^A_{ia} \in \{0,1\} & (i \in I^A, \ a \in A)
\end{cases}
$$

2.4.5 Formulation DT–O

In this section, we present the discrete-time formulation with on/off variables (DT–O), which is based on the RCPSP formulation of Kopanos et al. (2014). For the RCPSP, Kopanos et al. (2014) extend the formulation of Pritsker et al. (1969) with binary on/off variables W_{it}, which specify if activity i is in progress at time t. With these variables, the resource constraints can be modeled in a different manner than in Pritsker et al. (1969).

For the ACP, we extend the formulation DT–P (cf. Section 2.4.4) with binary on/off variables. Due to the preparation and the evaluation time, these on/off variables must be defined individually for candidates, assessors, and actors. However, this results in a large number of additional variables, which has a negative impact on the performance. For this reason, we only define the on/off variables for the candidates, and take the resource constraints of DT–P for the assessors and the actors. Hence, the resource constraints (2.40) for the candidates are replaced by constraints (2.44)–(2.46).

Constraints (2.44) ensure that each candidate performs at most one activity at a time.

$$
\sum_{i \in I_c : t \leq T - p^A_i - 1} W_{it} \leq 1 \qquad (c \in C, t = 0, \ldots, T) \tag{2.44}
$$

Constraints (2.45) link the pulse variables X_{it} to the on/off variables W_{it}.

$$W_{it} = \sum_{\tau=\max(0,t-p_i+p_i^A+1)}^{t} X_{i\tau} \qquad (i \in I, t = 0, \ldots, T - p_i^A - 1) \tag{2.45}$$

Constraints (2.46) are valid equalities that tighten the formulation.

$$\sum_{t=0}^{T-p_i^A-1} W_{it} = p_i - p_i^A \qquad (i \in I) \tag{2.46}$$

In sum, formulation (DT–O) reads as follows:

$$\text{(DT–O)} \quad \begin{cases} \text{Min } D & \\ \text{s.t. } (2.44)\text{–}(2.46) & \\ \quad (2.37)\text{–}(2.39), (2.41)\text{–}(2.43) & \\ \quad (2.7), (2.9), (2.11), (2.13)\text{–}(2.15) & \\ \quad X_{it} \in \{0,1\} & (i \in I, t = 0, \ldots, T) \\ \quad W_{it} \in \{0,1\} & (i \in I, t = 0, \ldots, T - p_i^A - 1) \\ \quad Y_{ij}^A \in \{0,1\} & (i, j \in I^A : \ i \neq j) \\ \quad V_{ca} \in \{0,1\} & (c \in C, \ a \in A) \\ \quad Z_{ia}^A \in \{0,1\} & (i \in I^A, \ a \in A) \end{cases}$$

2.5 Lower bounds

In this section, we derive some lower bounds for the AC duration. In Section 2.5.1, we present four lower bounds based on the assessors' workload. In Section 2.5.2, we present two lower bounds based on the candidates' workload.

2.5.1 Lower bounds based on the assessors' workload

In this section, we present four different lower bounds ($LB_1, \ldots, LB_4$) that are based on the assessors' workload. In contrast to lower bounds LB_1 and LB_2, lower bounds LB_3 and LB_4 consider the no-go relationships.

Lower bound LB_1 corresponds to the average workload of the assessors increased by the shortest preparation time of an activity. This preparation time is included because the assessors are never required before that time. The lower bound LB_1 reads as follows.

$$LB_1 = \left\lceil \sum_{i \in I^A} \frac{r_i^A (p_i - p_i^C)}{|A|} \right\rceil + \min_{i \in I^A} p_i^C$$

Lower bound LB_2 is obtained by considering only the activities that require two assessors. The total workload of these activities must be completed by an even number of assessors. Hence, if the number of assessors $|A|$ is odd, then the following lower bound LB_2 is valid.

$$LB_2 = \left\lceil \sum_{i \in I^A : r_i^A = 2} \frac{2(p_i - p_i^C)}{|A| - 1} \right\rceil + \min_{i \in I^A} p_i^C$$

Lower bound LB_3 takes the no-go relationships of each assessor into consideration. The workload of all activities to which assessor a cannot be assigned due to no-go relationships is evenly distributed among the remaining $|A| - 1$ assessors and increased by the shortest preparation time. For each assessor $a \in A$, this corresponds to a lower bound.

$$LB_3 = \max_{a \in A} \left\lceil \sum_{c \in C : (c,a) \in N} \sum_{i \in I_c} \frac{r_i^A (p_i - p_i^C)}{|A| - 1} \right\rceil + \min_{i \in I^A} p_i^C$$

Lower bound LB_4 combines the underlying ideas of LB_2 and LB_3. We only consider activities that require two assessors and for which the corresponding candidates have a no-go relationship with assessor a. For these activities, an even number of assessors is required at any time. However, if the number of assessors is even and assessor a cannot

be assigned to these activities due to the no-go relationships, it follows that one assessor $a^* \neq a$ is not needed. Hence, the workload of all activities that require two assessors and to which assessor a cannot be assigned is evenly distributed among the remaining $|A| - 2$ assessors. Again, the shortest preparation time of an activity is added to increase the lower bound. Hence, if the number of assessors $|A|$ is even, then lower bound LB_4 is valid.

$$LB_4 = \max_{a \in A} \left\lceil \sum_{c \in C:(c,a) \in N} \sum_{i \in I_c : r_i^A = 2} \frac{2(p_i - p_i^C)}{|A| - 2} \right\rceil + \min_{i \in I^A} p_i^C$$

2.5.2 Lower bounds based on the candidates' workload

In this section, we present two lower bounds for the AC duration based on the candidates' workload. The first lower bound (LB_5) is valid in general, and the second lower bound (LB_6) is only valid under certain conditions. Because each candidate must perform the same tasks, we do not need to differentiate between different candidates. Hence, in the following, we consider the tasks to be executed by each candidate and the lunch break rather than activities for individual candidates. The set of tasks and the lunch break are denoted by Q and l, respectively. It should be noted that the lunch break is not included in Q. Let p_q, p_q^C, and p_q^A be the duration, the preparation time, and the assessors' evaluation time of task $q \in Q$, respectively. The duration of the lunch break is p_l, and its preparation time (p_l^C) and evaluation time (p_l^A) are zero.

Because the tasks and the lunch break must be performed sequentially, lower bound LB_5 is valid.

$$LB_5 = \sum_{q \in Q \cup \{l\}} \left(p_q - p_q^A\right)$$

The term $\min_{q \in Q \cup \{l\}} p_q^A$ could be added to LB_5 because the AC cannot end before all tasks and the lunch break are completed. However, the evaluation time of the lunch break is always equal to zero and, thus, this term is always zero. The lunch break cannot be excluded from this term, because each candidate can have the lunch break at the end if

the latest possible start time is not violated.

To motivate lower bound LB_6, we first consider an illustrative example with two candidates and three assessors. Each candidate has to perform a task (activities k_1 and k_2) that requires two assessors and a lunch break (activities l_1 and l_2); activities k_1 and k_2 cannot be scheduled in parallel due to the limited number of assessors. Figure 2.7 depicts two feasible schedules for this example. In the schedule on the left, both candidates have the lunch break at the end. Due to the limited number of assessors, candidate C_2 has a waiting time. In this case, the AC duration D corresponds to the lower bound LB_5 plus the waiting time. In the schedule on the right, candidate C_2 performs the lunch break first. In this case, the AC duration D correspond to the lower bound LB_5 plus the evaluation time of the task.

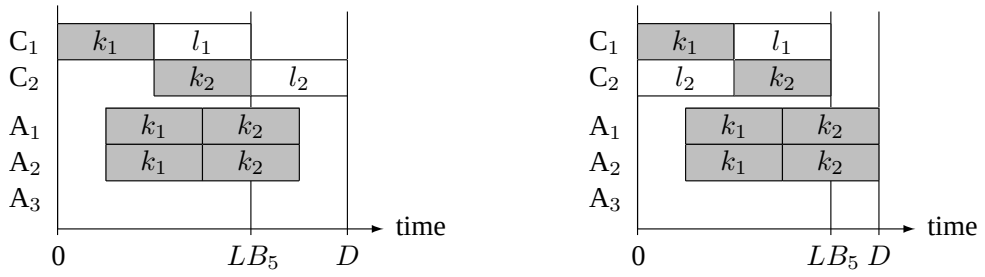


Figure 2.7: Schedules of an example with (left) and without (right) waiting time for the candidates

In this example, either a candidate has a waiting time, or the last activity of a candidate does not correspond to the lunch break. With this in mind, we propose lower bound LB_6, which is valid under certain conditions. According to our industry partner, these conditions are fulfilled by a considerable number of real-life instances.

Theorem 1. *Let r be a task with the shortest evaluation time. If (i) $\lfloor |A|/2 \rfloor < |C|$ and (ii) all tasks except task r require two or more assessors, then the following lower bound is*

valid.

$$LB_6 = \delta_0 + \min(\delta_1, \delta_2)$$

whereas:

$$\delta_0 = \sum_{q \in Q \cup \{l\}} \left(p_q - p_q^A\right)$$

$$\delta_1 = \min_{q \in Q \setminus \{r\}} p_q^A$$

$$\delta_2 = \min_{q \in Q \setminus \{r\}} (p_q - p_q^C - p_q^A) + \min_{q \in Q \setminus \{r\}} p_q^A - \max(p_l, p_r - p_r^A)$$

Proof. If the conditions (i) and (ii) hold for a given problem instance, any feasible solution belongs either to case 1 or to case 2.

- Case 1: The last activity of at least one candidate does not correspond to a lunch break or an activity of task r. It results that after the candidate completes this last activity, the assessors have an evaluation time of at least δ_1. Hence, $\delta_0 + \delta_1$ is a lower bound if the solution belongs to case 1.

- Case 2: The last activity of each candidate either corresponds to a lunch break or an activity of task r. We show that in this case, at least one candidate has a waiting time of at least δ_2 because condition (i) implies that not all candidates can perform an activity that requires two assessors at the same time. δ_2 corresponds to the length of the minimum time interval during which the required number of assessors exceeds the number of available assessors.

  Let k denote an arbitrary task that requires two assessors. To determine δ_2, we first consider the four possibilities for ordering the last activities such that the lunch break or task r are performed at the end by each candidate (cf. Figure 2.8).

  a) The lunch break is performed at the end and preceded by task r. Task r is preceded by task k.

  b) Task r is performed at the end and preceded by the lunch break. The lunch

break is preceded by task k.

c) Task r is performed at the end and preceded by task k. The lunch break ends some time before task k.

d) The lunch break is performed at the end and preceded by task k. Task r ends some time before task k.

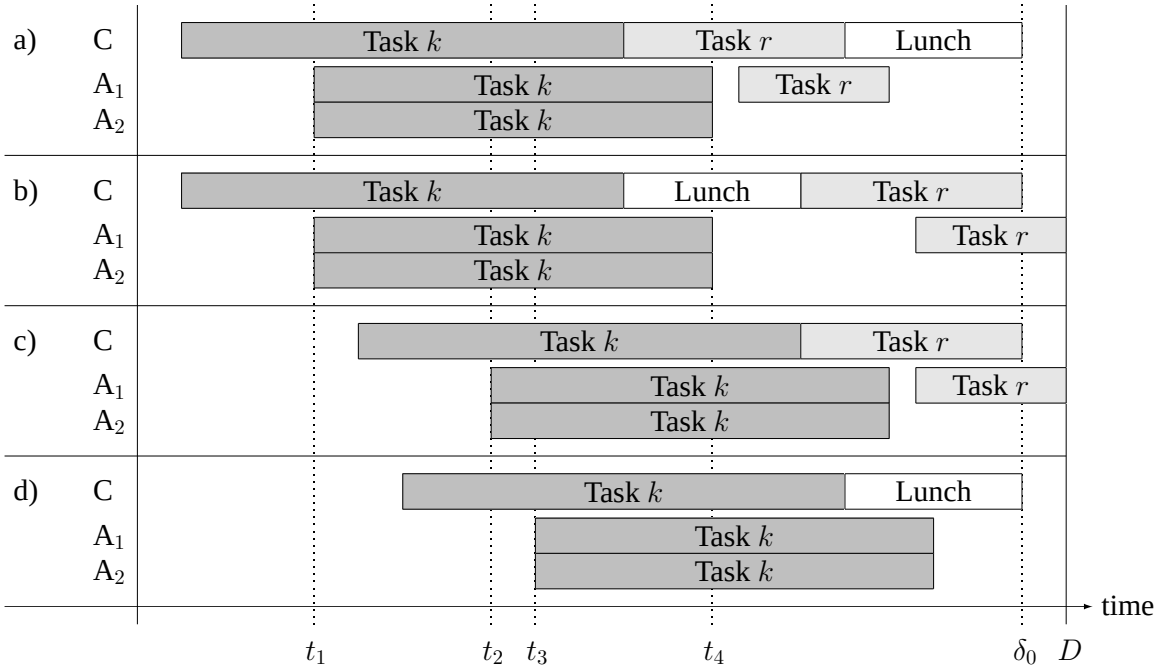


Figure 2.8: All possible orders of the last activities and corresponding assessor requirements

In Figure 2.8, the time point t_4 in a) and b) corresponds to the earliest possible finish time of task k for the assessors. The time points t_1, t_2, and t_3 correspond to the possible start times of task k for the assessors if no candidate has a waiting time. The values of these time points are as follows.

$$t_1 = \delta_0 - p_l - (p_r - p_r^A) - (p_k - p_k^C - p_k^A)$$

$$t_2 = \delta_0 - (p_r - p_r^A) - (p_k - p_k^C - p_k^A)$$

$$t_3 = \delta_0 - p_l - (p_k - p_k^C - p_k^A)$$

$$t_4 = \delta_0 - p_l - (p_r - p_r^A) + p_k^A$$

Overall, the latest possible start time of task k for the assessors corresponds to

$$\max(t_1, t_2, t_3) = \delta_0 - \min(p_l, p_r - p_r^A) - (p_k - p_k^C - p_k^A).$$

If $t_4 > \max(t_1, t_2, t_3)$ and no candidate has a waiting time, then there is a time interval with a minimum length of $t_4 - \max(t_1, t_2, t_3)$ during which every candidate performs a task that requires two assessors. Because $\lfloor |A|/2 \rfloor < |C|$, the required number of assessors exceeds the available number of assessors in this interval. To resolve this conflict, at least one task k must be delayed, which leads to a minimum waiting time for at least one candidate of $t_4 - \max(t_1, t_2, t_3)$.

To derive a lower bound for the AC duration, we determine the smallest possible value of t_4 and the largest possible value of $\max(t_1, t_2, t_3)$ as follows.

$$t_4 \geq \delta_0 - p_l - (p_r - p_r^A) + \min_{q \in Q \setminus \{r\}} p_q^A$$
$$\max(t_1, t_2, t_3) \leq \delta_0 - \min(p_l, p_r - p_r^A) - \min_{q \in Q \setminus \{r\}} (p_q - p_q^C - p_q^A)$$

Hence, the minimum waiting time corresponds to

$$\begin{aligned} t_4 - \max(t_1, t_2, t_3) &\geq \min_{q \in Q \setminus \{r\}} (p_q - p_q^C - p_q^A) + \min_{q \in Q \setminus \{r\}} p_q^A - \max(p_l, p_r - p_r^A) \\ &= \delta_2. \end{aligned}$$

Thereby, we used $\alpha + \beta - \min(\alpha, \beta) = \max(\alpha, \beta)$, where α, β are two arbitrary numbers. Hence, $\delta_0 + \delta_2$ is a lower bound if the solution belongs to case 2.

Overall, $LB_6 = \delta_0 + \min(\delta_1, \delta_2)$ is a lower bound for the AC duration if conditions (i) and (ii) hold. □

In the performance analysis, we use the maximum of these problem-specific lower bounds. If for an instance the necessary conditions for any of the lower bounds are not

fulfilled, we set their respective value to 0.

$$LB^{+} = \max(LB_1, LB_2, \ldots, LB_6)$$

2.6 Comparative analysis

We implemented the MIP formulations presented in Section 2.4 in AMPL, and we used the Gurobi Optimizer 6.0.5 as solver. All calculations were performed on an HP workstation with an Intel Xeon 2.67 GHz CPU and 24 GB RAM. The computational experiment was performed using four real-life instances and 240 test instances derived from real-life data. We limited the CPU time of the solver to 3,600 seconds for the real-life instances and to 600 seconds for the test instances. We used Gurobi with its default settings. Additionally, we applied Gurobi with the parameter MIPFocus set to 1. The parameter MIPFocus determines the MIP solution strategy of the solver. When this parameter is set to 1, Gurobi focuses on quickly generating good feasible solutions rather than increasing the lower bound. The default setting is 0, which aims to balance between finding good feasible solutions and proving optimality. For the DT formulations, the upper bound of the AC duration was set to $T = 200$ for all instances; this value is prescribed by the human resource provider.

In Section 2.6.1, we describe the instances that we used in our computational study. In Section 2.6.2, we discuss our computational results for the real-life instances. In Section 2.6.3, we provide the results for the test instances. In Section 2.6.4, we compare our problem-specific lower bounds.

2.6.1 Instances

The number of candidates $|C|$, assessors $|A|$, actors $|P|$, tasks $|E|$ and activities $|I|$ of the four real-life instances are listed in Table 2.5. The last column indicates whether at least

Table 2.5: Real-life instances

Instance	$\|C\|$	$\|A\|$	$\|P\|$	$\|E\|$	$\|I\|$	No-go relationships
RL1	7	10	2	5	42	no
RL2	11	11	3	5	66	no
RL3	9	11	3	5	54	yes
RL4	6	9	3	5	36	no

one no-go relationship exists. We denote the real-life instances with RL1, ..., RL4.

To test the different MIP formulations, we additionally devised a test set with 240 test instances based on real-life data. For the RCPSP, the well-known test instances of Kolisch and Sprecher (1997) were generated by systematically varying the complexity factors resource strength (RS), resource factor (RF), and network complexity (NC). These factors are only partially applicable to generate the ACP instances. The factor NC corresponds to the average number of precedence relationships per activity. Because there are no precedence relationships among the activities of the AC, we do not require such a factor. The factors RF and RS correspond to the average portion of the resources used by an activity and the scarcity of the resources, respectively. The factor RF can be interpreted as the average number of assessors required by an activity. To ensure that the instances are as close to reality as possible, we selected real-life tasks with given requirements for assessors and actors. Hence, we do not require a factor such as RF. The factor RS can be interpreted as the scarcity of the assessors. We use a similar factor to determine the number of available assessors. In total, we generated the 240 test instances by varying five complexity factors. Thereby, the employed experimental levels of each complexity factor were based on real-life data provided by the human resource management service provider. The complexity factors are as follows.

The complexity factors n^C and n^E correspond to the number of candidates and tasks, respectively, and determine the number of activities of an instance. The tasks were randomly selected from a set of 15 real-life tasks. The experimental levels $n^C \in$

$\{4, 5, \ldots, 10, 11\}$ and $n^E \in \{4, 5\}$ were used.

The complexity factor a^S corresponds to the average number of assignments per assessor. This factor is used to determine the number of assessors n^A of an instance. The number of assessors is equal to the nearest integer to $\sum_{i \in I^A} r_i^A / a^S$; thus, the numerator corresponds to the total number of assessor assignments. The experimental levels $a^S \in \{6.0, 8.5, 10.4\}$ correspond to the observed real-life minimum, average, and maximum.

The complexity factor a^N corresponds to the proportion of assessors who have one or more no-go relationships (no-go assessors). The number of no-go assessors is given by the nearest integer to $a^N n^A$. The no-go assessors were randomly selected from the set of all assessors. The experimental levels $a^N \in \{\frac{1}{6}, \frac{1}{3}\}$ were used.

The complexity factor a^R corresponds to the average number of no-go relationships per no-go assessor. The number of no-go relationships is equal to the product of a^R and the number of no-go assessors. The no-go relationships were randomly assigned to pairs of candidates and no-go assessors such that (1) each no-go assessor has at least one no-go relationship and (2) at least $\lfloor |A| / 2 \rfloor$ different assessors can be assigned to each candidate. The experimental levels $a^R \in \{2, 3\}$ were used.

Because the actors are paid for each role play in which they actually perform, they are not considered to be a critical resource. Hence, the number of actors was set to 3 for all instances, which corresponds to the observed real-life maximum.

For each combination of complexity factor levels, an instance was generated; this leads to $8 \cdot 2 \cdot 3 \cdot 2 \cdot 2 = 192$ test instances. Additionally, $8 \cdot 2 \cdot 3 = 48$ test instances without no-go relationships (i.e., $a^N = a^R = 0$) were generated.

2.6.2 Computational results: real-life instances

For the real-life instances RL1, ..., RL4, the results obtained by the solver using the MIP formulations CT–A, CT–F, CT–O, DT–P, and DT–O with MIPFocus set to 0 are reported in Table 2.6. We compare the objective function values (D) with the lower bounds obtained

Table 2.6: Results for real-life instances with MIPFocus set to 0

Instance	CT–A		CT–F		CT–O		DT–P		DT–O		LB^+
	D	LB	D	LB	D	LB	D	LB	D	LB	
RL1	89	67	90	37	**88**	74	128	81	95	71	82
RL2	136	59	158	36	**132**	49	149	103	173	72	110
RL3	**106**	62	121	36	107	49	125	80	118	63	90
RL4	83	70	86	36	**82**	74	87	81	86	80	82

Table 2.7: Results for real-life instances with MIPFocus set to 1

Instance	CT–A		CT–F		CT–O		DT–P		DT–O		LB^+
	D	LB	D	LB	D	LB	D	LB	D	LB	
RL1	**86**	49	**86**	36	88	49	98	76	88	70	82
RL2	**124**	49	128	36	129	54	159	70	150	69	110
RL3	102	49	**100**	36	108	49	118	59	114	63	90
RL4	**82**	56	84	36	84	55	**82**	82	**82**	76	82

by the solver (LB) and the maximum value over all problem-specific lower bounds (LB^+). For each instance, the best objective function values obtained are highlighted in boldface. Using the default solver settings, the solver obtains on average the best objective function values with CT–O and the highest lower bounds with DT–P. For all real-life instances, these lower bounds are smaller than or equal to the problem-specific lower bound. The problem-specific lower bound of instance RL4 corresponds to the objective function value obtained with CT–O, i.e., this solution is optimal.

Table 2.7 lists the results obtained by the solver with MIPFocus set to 1. Except for CT–O, the average AC duration is improved. However, on average, the lower bounds are worse. CT–A devises the best solutions for three instances, CT–F for two instances, and DT–P and DT–O for one instance. The smallest instance (RL4) is even solved to optimality using formulation DT–P. Both, CT–A and DT–O, also find a solution with an optimal objective function value, but they do not prove optimality within the prescribed

CPU time.

2.6.3 Computational results: test instances

Based on the number of activities $|I|$, we divide the 240 test instances into small-sized (20–34 activities, 75 instances), medium-sized (35–49 activities, 90 instances), and large-sized (50–66 activities, 75 instances) instances. For these three ranges of $|I|$, the average number of variables and constraints for the different formulations are presented in Figure 2.9. Regardless of the number of activities, DT–O has the highest number of variables. For small- and medium-sized instances, DT–O has also the highest number of constraints. However, with an increasing number of activities, the number of constraints increases less for the DT formulations than for the CT formulations. For the large-sized instances, CT–O has the highest number of constraints.

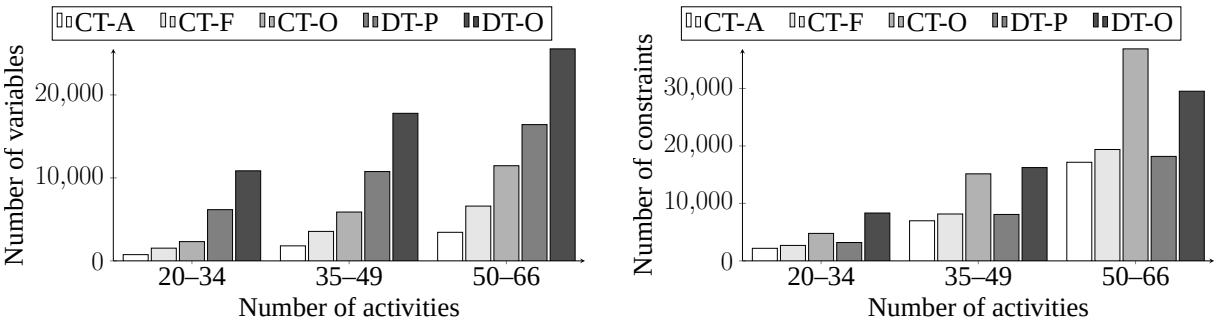


Figure 2.9: Average number of variables (left) and constraints (right)

Table 2.8 reports the average relative gaps between the obtained solutions and the problem-specific lower bound ($gap^+ = (D - LB^+)/D$), as well as the average relative gaps between the obtained solutions and the lower bounds obtained by the solver ($gap = (D - LB)/D$). To evaluate the quality of the solutions, we use gap^+. To evaluate the quality of the lower bounds provided by the solver, we use gap. For each solver setting used, the best results are highlighted in boldface.

Regardless of the solver settings employed, the best gap^+ is obtained with CT–A (10.3% for MIPFocus set to 0 and 9.3% for MIPFocus set to 1), and the worst gap^+ is

Table 2.8: Aggregated results for all 240 test instances

Formulation	MIP-Focus	CT–A	CT–F	CT–O	DT–P	DT–O
Average gap^+ [in %]	0	**10.3**	15.1	12.5	27.7	19.4
	1	**9.3**	11.1	11.2	26.8	18.5
Average *gap* [in %]	0	44.7	59.8	50.4	37.5	**36.6**
	1	56.7	65.0	55.1	44.6	**37.8**
Number of feasible solutions	0	**240**	216	**240**	**240**	234
	1	**240**	235	**240**	**240**	238
Number of optimal solutions	0	**36**	29	32	22	19
	1	27	24	**30**	22	27
Number of best solutions	0	**170**	51	80	22	60
	1	**161**	69	81	27	57

obtained with DT–P. In contrast, the smallest *gap* is obtained with DT–O. Similarly to the results of Kopanos et al. (2014), better solutions are obtained with with DT–O than with DT–P. We conclude that the CT formulations provide better solutions, and that the DT formulations provide better lower bounds. For all formulations, *gap* considerably exceeds gap^+. We deduce that the problem-specific lower bounds are considerably higher than the lower bounds obtained by the solver within the prescribed CPU time limit.

With CT–A, CT–O, and DT–P, feasible solutions are obtained for all 240 test instances within the prescribed CPU time limit. With CT–F and MIPFocus set to 0, feasible solutions are obtained only for 216 instances (i.e., 90% of the instances). With MIPFocus set to 1, this number increases to 235 (i.e., 97.9%); feasible solutions could not be obtained for five of the large-sized instances.

To determine the number of optimal solutions, we compare the objective function value obtained with the maximum value over all problem-specific lower bounds and the lower bound obtained by the solver. With 36 instances, CT–A obtains the highest number of optimal solutions.

The number of best solutions corresponds to the number of times that a formulation

generates a best solution. With MIPFocus set to 0, CT–A provides a best solution for 170 instances. This means that the other formulations generate better solutions for 70 instances only.

With MIPFocus set to 1, the average solution quality for all formulations is improved. This is indicated by a reduction of gap^+. For CT–F, this reduction is quite considerable (from 15.1% to 11.1%). This might indicate that the MIP solution strategy used by the solver exploits the resource-flow information in an efficient manner. However, the average *gap* is larger with MIPFocus set to 1 because this solver setting focuses less on improving the lower bounds but gives priority to the quick generation of good feasible solutions. Therefore, the number of feasible solutions is increased for CT–F. Surprisingly, for the CT formulations CT–A, CT–F and CT–O, the number of optimal solutions obtained is lower with MIPFocus set to 1.

Table 2.9 reports the average results for all instances with the same problem characteristics. The overall results show that with MIPFocus set to 1 the best solutions are obtained. However, for CT–A and small-sized instances, the solver performs better with MIPFocus set to 0.

The number of activities $|I|$ and the level of complexity factor a^S, which defines the number of available assessors, have a significant impact on both relative gaps. In contrast, the levels of complexity factors a^N and a^R, which define the no-go relationships, have no systematic impact on the relative gaps. Parameter f corresponds to the average duration of the activities. The performance of DT–O is affected most by the value of f. For instances with short activities ($11 \leq f \leq 13$), the performance of DT–O is almost as good as the performance of CT–A. However, for the instances with longer activities, the average gaps are much higher. Surprisingly, such an effect is not observed with DT–P.

According to the results obtained by Koné et al. (2011) for the RCPSP, DT formulations are better for instances with activities that have a short duration. Although the durations of the AC activities are quite short, we do not observe similar results for the ACP. Overall,

Table 2.9: Average gap^+ for different instance characteristics

Instance characteristics		MIP-Focus	Average gap^+				
			CT–A	CT–F	CT–O	DT–P	DT–O
$\|I\|$	20–34	0	**1.8**	3.1	2.6	10.7	11.0
		1	**2.4**	3.0	2.6	6.8	6.0
	35–49	0	**8.3**	14.9	11.4	28.0	12.7
		1	**7.8**	9.3	9.5	28.0	13.8
	50–66	0	**21.2**	31.8	23.6	44.4	36.5
		1	**18.1**	22.0	21.8	45.4	37.3
a^S	6	0	**10.1**	15.2	12.8	29.3	22.4
		1	**9.6**	11.3	11.9	23.4	20.3
	8	0	**12.1**	17.8	14.3	31.0	20.4
		1	**11.2**	12.9	12.5	32.2	19.4
	10.4	0	**8.8**	11.9	10.4	22.9	15.3
		1	**7.1**	9.0	9.1	24.8	15.8
a^N	0	0	**10.7**	16.9	12.5	25.9	18.8
		1	**9.2**	10.6	11.0	24.9	17.5
	0.17	0	**10.3**	14.9	12.8	27.4	19.4
		1	**9.4**	11.3	11.2	26.9	17.5
	0.33	0	**10.2**	14.5	12.2	29.0	19.6
		1	**9.3**	11.2	11.2	27.6	20.0
a^R	0	0	**10.7**	16.9	12.5	25.9	18.8
		1	**9.2**	10.6	11.0	24.9	17.5
	2	0	**10.3**	15.9	12.4	26.8	18.0
		1	**9.4**	11.3	10.8	27.0	17.8
	3	0	**10.2**	13.5	12.6	29.6	21.1
		1	**9.3**	11.2	11.7	27.6	19.8
f	11–13	0	**9.3**	13.0	12.2	25.6	10.6
		1	**8.6**	10.0	10.5	26.8	11.2
	13–15	0	**8.9**	13.5	10.7	24.4	18.8
		1	**7.3**	9.4	9.8	22.9	17.5
	15–17	0	**11.5**	17.0	13.5	30.3	23.2
		1	**10.7**	12.4	12.2	28.8	21.9

Table 2.10: Comparison of problem-specific lower bounds

Based on workload of	Assessors				Candidates	
Lower bound	LB_1	LB_2	LB_3	LB_4	LB_5	LB_6
Number of instances with best lower bound	93	90	0	8	32	22

the CT formulations provide the best solutions. A drawback of the DT formulations may be the large number of variables (cf. Figure 2.9) which depend on the number of time points considered. In the RCPSP, the number of variables is reduced considerably with a simple preprocessing like the definition of earliest and latest start times for the activities. However, this preprocessing is based on precedence relationships, which do not exist in the ACP. Considering the CT formulations, CT–A performs best, and CT–O performs better than CT–F.

2.6.4 Computational results: problem-specific lower bounds

Table 2.10 compares the six problem-specific lower bounds presented in Section 2.5. The last row shows the number of instances for which the different lower bounds obtained the highest values. LB_1 and LB_2 each provide the highest lower bounds for more than 90 instances. However, lower bounds that consider no-go relationships (LB_3 and LB_4) only provide the highest values for a few instances. If the conditions for LB_6 hold, this lower bound provides the highest values for 22 instances.

2.7 Conclusions

Comparisons of alternative MIP formulations in the literature for project scheduling problems are primarily based on generic test instances. In this study, we analyzed the performance of two discrete-time and three continuous-time MIP formulations in a real-life application of project scheduling. We considered the problem of planning assessment

centers. For this problem, we developed new MIP formulations, and we provided problem-specific lower bounds. In contrast to the results generally obtained for the RCPSP, our comparative study indicates that the CT formulations outperform the DT formulations in terms of solution quality. However, using the DT formulations, the best MIP-based lower bounds are obtained.

The assessment center planning problem is an interesting and challenging optimization problem for future research. An important area is the development of heuristic solution procedures. Preliminary versions of an MIP-based heuristic and a list-scheduling heuristic are presented in Rihm and Trautmann (2016) and Zimmermann and Trautmann (2015). In the MIP-based heuristic, first, the activities are scheduled without assessor assignments; second, the assessors are assigned to the activities using the CT formulation with resource-flow variables presented in this study. In the list-scheduling heuristic, the activities are scheduled sequentially based on problem-specific priority rules. The MIP formulations and the problem-specific lower bounds presented in this paper can be used to analyze the performance of such heuristic approaches.

Bibliography

Alvarez-Valdes, R., Tamarit, J., 1993. The project scheduling polyhedron: dimension, facets and lifting theorems. European Journal of Operational Research 67 (2), 204–220.

Ambrosino, D., Paolucci, M., Sciomachen, A., 2015. Experimental evaluation of mixed integer programming models for the multi-port master bay plan problem. Flexible Services and Manufacturing Journal 27 (2–3), 263–284.

Artigues, C., Koné, O., Lopez, P., Mongeau, M., 2015. Mixed-integer linear programming formulations. In: Schwindt, C., Zimmermann, J. (Eds.), Handbook on Project Management and Scheduling Vol. 1. Springer, Cham, pp. 17–41.

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

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

Bixby, R. E., 2012. A brief history of linear and mixed-integer programming computation. Doc Math Extra Volume ISMP, 107–121.

Chen, X., Grossmann, I., Zheng, L., 2012. A comparative study of continuous-time models for scheduling of crude oil operations in inland refineries. Computers & Chemical Engineering 44, 141–167.

Christofides, N., Alvarez-Valdés, R., Tamarit, J. M., 1987. Project scheduling with resource constraints: a branch and bound approach. European Journal of Operational Research 29 (3), 262–273.

Collins, J. M., Schmidt, F. L., Sanchez-Ku, M., Thomas, L., McDaniel, M., Le, H., 2003. Can basic individual differences shed light on the construct meaning of assessment center evaluations? International Journal of Selection and Assessment 11 (1), 17–29.

Grüter, J., Trautmann, N., Zimmermann, A., 2014. An MBLP model for scheduling assessment centers. In: Huisman, D., Louwerse, I., Wagelmans, A. (Eds.), Operations Research Proceedings 2013. Springer, Berlin, pp. 161–167.

Kaplan, L., 1988. Resource-constrained project scheduling with preemption of jobs. Ph.D. thesis, University of Michigan.

Klein, R., 2000. Scheduling of resource-constrained projects. Kluwer, Amsterdam.

Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R. E., Danna, E., Gamrath, G., Gleixner, A. M., Heinz, S., et al., 2011. MIPLIB 2010. Mathematical Programming Computation 3 (2), 103–163.

Kolisch, R., Sprecher, A., 1997. PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. European Journal of Operational Research 96 (1), 205–216.

Koné, O., Artigues, C., Lopez, P., Mongeau, M., 2011. Event-based MILP models for resource-constrained project scheduling problems. Computers & Operations Research 38 (1), 3–13.

Koné, O., Artigues, C., Lopez, P., Mongeau, M., 2013. Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources. Flexible Services and Manufacturing Journal 25 (1–2), 25–47.

Kopanos, G. M., Kyriakidis, T. S., Georgiadis, M. C., 2014. New continuous-time and discrete-time mathematical formulations for resource-constrained project scheduling problems. Computers & Chemical Engineering 68, 96–106.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., Bianco, L., 1998. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. Management Science 44 (5), 714–729.

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

Pritsker, A. A. B., Waiters, L. J., Wolfe, P. M., 1969. Multiproject scheduling with limited resources: a zero-one programming approach. Management Science 16 (1), 93–108.

Rihm, T., Trautmann, N., 2016. A decomposition approach for an assessment center planning problem. In: Ruiz, R., Alvarez-Valdes, R. (Eds.), Proceedings of the 15th International Conference on Project Management and Scheduling. Valencia, pp. 206–209.

Stefansson, H., Sigmarsdottir, S., Jensson, P., Shah, N., 2011. Discrete and continuous time representations and mathematical models for large production scheduling problems: a case study from the pharmaceutical industry. European Journal of Operational Research 215 (2), 383–392.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., Tavares, L. V., 2008. An evaluation of the adequacy of project network generators with systematically sampled networks. European Journal of Operational Research 187 (2), 511–524.

Vielma, J. P., 2015. Mixed integer linear programming formulation techniques. SIAM Review 57 (1), 3–57.

Zapata, J. C., Hodge, B. M., Reklaitis, G. V., 2008. The multimode resource constrained multiproject scheduling problem: alternative formulations. AIChE Journal 54 (8), 2101–2119.

Zimmermann, A., Trautmann, N., 2014. Scheduling of assessment centers: an application of resource-constrained project scheduling. In: Fliedner, T., Kolisch, R., Naber, A. (Eds.), Proceedings of the 14th International Conference on Project Management and Scheduling. Munich, pp. 263–266.

Zimmermann, A., Trautmann, N., 2015. A list-scheduling approach for the planning of assessment centers. In: Hanzálek, Z., Kendall, G., McCollum, B., Šůcha, P. (Eds.), Proceedings of the Multidisciplinary International Scheduling Conference: Theory and Application. Prague, pp. 541–554.

Paper III

A mixed-integer programming-based heuristic for project scheduling with work-content constraints[1]

Adrian Zimmermann

Department of Business Administration
University of Bern

Contents

3.1 Introduction 82
3.2 Illustrative example 86
3.3 Basic MIP formulation 87
3.4 MIP-based heuristic 91
3.4.1 Preprocessing 93
3.4.2 Scheduling a single activity 95
3.4.3 Rescheduling a subset of activities 97
3.4.4 Solution for illustrative example 101
3.5 Computational experiment 102
3.5.1 Test set 103
3.5.2 Experimental design 103
3.5.3 Numerical results 105
3.6 Conclusions and outlook 107
Bibliography 109

Abstract

We consider the project scheduling problem in which each project activity has a prescribed work content that must be completed by a so-called work-content resource and the activities' resource usage may change over time. The resource usage must lie within prescribed bounds and cannot be changed for a minimum number of consecutive periods. The amount of resource units used determines the requirements for further resources. The activities must be scheduled such that the project makespan is minimized. For this problem, we devise a mixed-integer programming-based heuristic that schedules the activities iteratively. To improve the resource usage for multiple activities simultaneously, subsets of activities are rescheduled each time the activities' resource usage appears to be inefficient. Our computational results for a standard test set from the literature show that our heuristic outperforms the state-of-the-art method for medium- and large-sized instances, and that for many small-sized instances, optimal solutions are obtained.

3.1 Introduction

To remain competitive in today's dynamic business environment, firms are often required to execute projects such as the development of new products and services. A project is a unique endeavor that can be divided into precedence-interrelated activities that require time and scarce resources for their completion. The total amount of resource units used by an activity corresponds to the work content of that activity. Traditional problem formulations in the project-scheduling literature, such as the resource-constrained project scheduling problem (RCPSP) and its multi-mode version (MRCPSP), are based on the assumption that the activities have a constant resource usage during each period of their execution. Although the MRCPSP considers alternative execution modes with different

durations for the activities, each mode still involves a constant resource usage. In practice, however, project managers are often able to change the resource usage of an activity over time. For example, let an activity have a work content of three person days. In the RCPSP, the activity may be completed in three days by one person. In the MRCPSP, an additional execution mode might be considered that involves three persons on one day. Without the assumption of a constant resource usage, however, the activity may also be completed in two days by one person on one day and two persons on the other day. This flexible resource usage allows project managers to more efficiently utilize the scarce resources. The relevance of this concept is further emphasized by the fact that most commercially available project-management software packages provide the option to specify the work content of the activities.

We consider the problem of scheduling a project with work-content constraints (cf. Fündeling, 2006). Given are a set of activities, a single work-content resource (e.g., labor), and a set of non-work-content resources (e.g., different types of tools). The work-content resource and the non-work-content resources are renewable, and they have a constant capacity over time. Each activity has a prescribed work content that must be processed by the work-content resource. The work-content constraints are as follows. In each period during its execution, the amount of the work-content resource used by an activity must lie within prescribed upper and lower bounds. Furthermore, a minimum time lag between consecutive changes in an activity's usage of the work-content resource is imposed. This time lag is referred to as the minimum block length. The requirements for the non-work-content resources depend linearly on the amount of the work-content resource used; an increase in the usage of the work-content resource leads to a proportional increase in the usage of the non-work-content resources. In each period, an activity may only use a discrete number of resource units. Hence, any fractional requirements for the non-work-content resources are rounded up to the closest integer. The activities must be executed without interruption and are subject to completion-start precedence relationships, i.e., an activity

can only start once all its predecessor activities have been completed. The problem consists of determining the start times and the resource usage of the activities such that the project makespan is minimized, the precedence relationships are satisfied, the resource capacities are never exceeded, the entire work content of each activity is processed, and the work-content constraints are met.

For this problem, Fündeling (2006) develops an exact branch-and-bound-based solution procedure. To assess the performance of the procedure, the author devises a test set that consists of instances with up to 200 activities. The computational results indicate that the CPU time required by the procedure becomes prohibitively large even for small-sized instances. Furthermore, the problem is shown to be NP-hard. To address medium- and large-sized instances, Fündeling and Trautmann (2010) present a priority-rule-based heuristic. The authors apply the heuristic to the test set devised by Fündeling (2006). Their computational results indicate that the heuristic procedure generates good feasible solutions in short CPU time. Baumann and Trautmann (2013) devise a mixed-integer linear programming (MIP) formulation and apply it to 480 problem instances with 10 activities. Although most problem instances are solved to optimality, for some instances no feasible solution is obtained in reasonable CPU time. Their computational results indicate that an exact MIP-based approach is suitable for solving small-sized instances. Hence, the question arises of whether the problem can be decomposed into smaller subproblems that can be solved efficiently using mixed-integer linear programming. Such MIP-based decomposition approaches have been applied successfully to different types of scheduling problems. For example, Fırat and Hurkens (2012) apply an MIP-based heuristic to a workforce scheduling problem in which the activities require employees with different skills, and Toffolo et al. (2016) apply an MIP-based heuristic to a variant of the MRCPSP with multiple projects. In the literature, different scheduling problems have been studied in which the activities' resource usage may change over time. However, none of these problems incorporate all the constraints considered here. Dror et al. (1987), Hackman and Leachman

(1989), Jozefowska and Weglarz (1998), Kovács (2003), Márkus et al. (2003), and Kis (2005) consider the scheduling of activities that may use a continuous number of resource units. Kuhlmann (2003) considers the problem in which the activities' resource usage may change over time but without any further restrictions. Ranjbaer and Kianfar (2010) propose a genetic algorithm for a variant of the problem in which only the work-content resource is considered. Their algorithm might exclude optimal solutions because not all possible resource usages of the activities are considered. Naber and Kolisch (2014) propose four alternative MIP formulations for a variant of the problem in which the work content processed may be larger than the prescribed work content, and the activities may use a continuous number of resource units. Baumann and Trautmann (2014) apply the MIP of Baumann and Trautmann (2013) without the integrality constraints for the resource usage. Their computational results show that the computational burden is considerably reduced when these constraints are omitted.

In this paper, we propose a novel MIP-based heuristic procedure for scheduling projects with work-content constraints. Our heuristic starts by ordering the activities in a precedence-feasible list. Based on their order in the list, the activities are then scheduled iteratively. This iterative approach allows the generation of feasible schedules in short CPU time, but it may also lead to an inefficient resource usage. Hence, each time an activity has been scheduled, the procedure determines whether the resource usage of that activity appears to be inefficient. In that case, a subset of activities is rescheduled to determine a more efficient resource usage among the activities. For the scheduling of a single activity and the rescheduling of a subset of activities, we devise corresponding MIP formulations. For determining whether an activity's resource usage is inefficient, we provide an appropriate set of conditions. If all these conditions are met, then a rescheduling step is performed. Furthermore, we propose an MIP-based preprocessing routine that can be used to enhance the performance of our heuristic procedure. To evaluate the performance of the heuristic, we have applied it to the test set devised by Fündeling (2006), which

Table 3.1: Illustrative example: parameter values of activities

Activity	1	2	3	4	5	6	7	8	9	10
Work content	47	54	40	26	45	94	65	49	49	67
Upper bound	12	8	9	10	9	13	10	5	11	12
Lower bound	2	2	7	2	4	4	3	2	7	6
Predecessors	–	–	{1, 2}	{1}	{2, 4}	{3}	{5}	{6}	{3, 6}	{7, 8}

consists of instances with 10, 20, 40, 100 and 200 activities. Our computational results indicate that the heuristic generates very good solutions for small-sized problem instances and that it outperforms the state-of-the-art method for medium- and large-sized instances. Indeed, for a large number of small-sized instances, the proposed heuristic obtains optimal solutions in very short CPU time.

The remainder of this paper is structured as follows. In Section 3.2, we illustrate the scheduling problem using an example. In Section 3.3, we present the MIP formulation of Baumann and Trautmann (2013). This MIP formulation is the basis for the MIP formulations employed in our heuristic procedure. In Section 3.4, we present our MIP-based heuristic. In Section 3.5, we describe our computational experiment. In Section 3.6, we provide some concluding remarks and an outlook on future research.

3.2 Illustrative example

In this section, we present an example to illustrate the scheduling problem. The example involves 10 activities that require only the work-content resource. The work-content resource has a capacity of 13 units, and the minimum block length is 3 periods. For each activity, Table 3.1 lists the work content and the bounds on the usage of the work-content resource per period (in resource units). The set of immediate predecessors is listed in row five.

We applied the MIP of Baumann and Trautmann (2013) to solve the illustrative

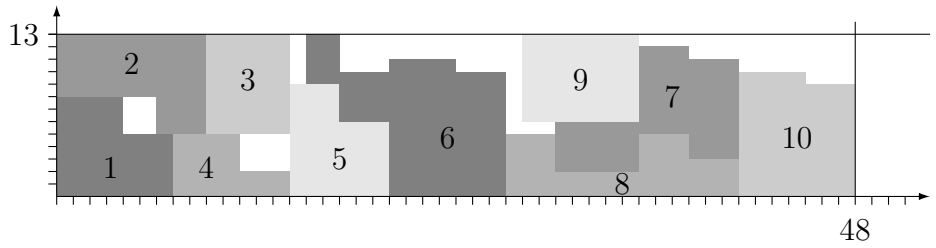


Figure 3.1: Illustrative example: best schedule obtained by the MIP of Baumann and Trautmann (2013) after 600 seconds of CPU time

example. The best schedule obtained by the MIP after 600 seconds of CPU time is presented in Figure 3.1. The makespan is 48 periods, and the MIP-based lower bound is 45 periods. The best makespan obtained by the heuristic of Fündeling and Trautmann (2010) is 52 periods.

3.3 Basic MIP formulation

The mixed-integer linear programs employed in our heuristic procedure are based on the MIP formulation of Baumann and Trautmann (2013). For completeness, we include their MIP formulation in this section. The nomenclature is provided in Table 3.2.

For each activity $i \in V$, the set of relevant periods T_i must be determined. The decision variables related to activity i are only defined for these periods. The set of relevant periods is determined based on an activity's earliest start time EST_i and on its latest finish time LFT_i. The earliest start times are calculated by forward recursion. In this way, the earliest start time of an activity i is set to the maximum earliest finish time over all its predecessor activities, i.e., $EST_i = \max_{j \in P_i}(EST_j + \underline{p}_j)$, where $\underline{p}_j = \lceil w_j / \overline{r}_{jk^*} \rceil$ is a lower bound for the duration of an activity j. Similarly, the latest finish times are calculated by backward recursion.

Using this notation, the MIP formulation reads as follows. The objective is to minimize the makespan of the project. Here, the dummy activity $n+1$ represents the end of

Table 3.2: Nomenclature

Indices	
$i,\ j$	Activity
k	Resource
k^*	Work-content resource
t	Period that starts at time $t-1$ and ends at time t
Sets	
P_i	Set of immediate predecessors of activity i
$\mathcal{R}$	Set of resources
$\mathcal{R}_i$	Set of resources required by activity i
T	Set of periods
T_i	Relevant periods for activity i ($T_i = \{EST_i + 1, \ldots, LFT_i, LFT_i + 1\}$)
V	Set of activities ($V = \{1, \ldots, n+1\}$)
V^r	Set of real activities ($V^r = \{1, \ldots, n\}$)
V^r_{kt}	Set of real activities that can be processed in period t by resource k
Parameters	
EST_i	Earliest start time of activity i
LFT_i	Latest finish time of activity i
m	Minimum block length
n	Number of activities
$\overline{p}_i$	Upper bound for the duration of an activity i, i.e., $\overline{p}_i = \lfloor w_i / \underline{r}_{ik^*} \rfloor$
$\underline{p}_i$	Lower bound for the duration of an activity i, i.e., $\underline{p}_i = \lceil w_i / \overline{r}_{ik^*} \rceil$
R_k	Capacity of resource k
$\overline{r}_{ik}$	Upper bound for the amount of resource k used by activity i
$\underline{r}_{ik}$	Lower bound for the amount of resource k used by activity i
s_{ik}	Incremental increase of the usage of resource $k \in \mathcal{R} \setminus \{k^*\}$ by activity i per additional unit of work-content resource
w_i	Work content of activity i
Decision variables	
D_{it}	$\begin{cases} = 1, & \text{if usage of resource } k^* \text{ by activity } i \text{ in period } t \text{ differs from } t-1 \\ = 0, & \text{otherwise} \end{cases}$
R_{ikt}	Amount of resource $k \in \mathcal{R}$ used by activity i in period t
X_{it}	$\begin{cases} = 1, & \text{if activity } i \text{ is processed in period } t \in T_i \\ = 0, & \text{otherwise} \end{cases}$

the project because it can only start after all real activities have been completed. Constraint (3.1) ensures that the dummy activity $n+1$ is scheduled once within the planning horizon.

$$\begin{aligned} &\text{Min.} \sum_{t \in T_{n+1}} tX_{n+1,t} - 1 \\ &\text{s.t.} \sum_{t \in T_{n+1}} X_{n+1,t} = 1 \end{aligned} \tag{3.1}$$

Constraints (3.2) ensure that an activity cannot be interrupted once it has started.

$$X_{i,t-1} - \sum_{t' \in T_i : t' < t} \frac{R_{ik^*t'}}{w_i} \leq X_{it} \qquad (i \in V^r;\ t \in T_i :\ t > EST_i + 1) \tag{3.2}$$

Constraints (3.3) guarantee that the total number of work-content resource units used by an activity coincides with its work content.

$$\sum_{t \in T_i} R_{ik^*t} = w_i \qquad (i \in V^r) \tag{3.3}$$

Constraints (3.4) and (3.5) ensure that the number of work-content resource units used by an activity lies within the prescribed lower and upper bound, respectively.

$$\underline{r}_{ik^*} X_{it} \leq R_{ik^*t} \qquad (i \in V^r;\ t \in T_i) \tag{3.4}$$

$$\overline{r}_{ik^*} X_{it} \geq R_{ik^*t} \qquad (i \in V^r;\ t \in T_i) \tag{3.5}$$

Constraints (3.6) to (3.8) force the value of variable D_{it} to be 1 if the resource usage changes between two consecutive periods $t-1$ and t.

$$R_{ik^*t} \leq \overline{r}_{ik^*} D_{it} \qquad (i \in V^r;\ t = EST_i + 1) \tag{3.6}$$

$$R_{ik^*t} - R_{ik^*,t-1} \leq \overline{r}_{ik^*} D_{it} \qquad (i \in V^r;\ t \in T_i :\ t > EST_i + 1) \tag{3.7}$$

$$R_{ik^*,t-1} - R_{ik^*t} \leq \overline{r}_{ik^*} D_{it} \qquad (i \in V^r;\ t \in T_i :\ t > EST_i + 1) \tag{3.8}$$

Constraints (3.9) ensure that an activity is not processed after its latest finish time.

$$X_{it} = 0 \qquad (i \in V^r;\ t = LFT_i + 1) \tag{3.9}$$

Constraints (3.10) ensure that the number of periods between two consecutive changes in the resource usage is larger than or equal to the prescribed minimum block length.

$$\sum_{t'=0}^{m-1} D_{i,t+t'} \leq 1 \qquad (i \in V^r;\ t \in T_i :\ t \leq LFT_i - (m-2)) \tag{3.10}$$

Constraints (3.11) ensure that an activity can only start after the work content of all its predecessor activities has been completed.

$$X_{it} \leq \sum_{t' \in T_j : t' < t} \frac{R_{jk^*t'}}{w_j} \qquad (i \in V;\ j \in P_i;\ t \in T_i) \tag{3.11}$$

Constraints (3.12) calculate the requirements of the non-work-content resources of an activity in each period.

$$\underline{r}_{ik} X_{it} + s_{ik}(R_{ik^*t} - \underline{r}_{ik^*}) \leq R_{ikt} \qquad (i \in V^r;\ k \in \mathcal{R}_i \setminus \{k^*\};\ t \in T_i) \tag{3.12}$$

Constraints (3.13) ensure that the resource capacities are never exceeded.

$$\sum_{i \in V_{kt}^r} R_{ikt} \leq R_k \qquad (k \in \mathcal{R};\ t \in T) \tag{3.13}$$

In sum, the MIP formulation (BT13) of Baumann and Trautmann (2013) reads as follows.

$$\text{(BT13)} \quad \begin{cases} \text{Min} \displaystyle\sum_{t \in T_{n+1}} t X_{n+1,t} - 1 & \\ \text{s.t.} \quad (3.1)\text{–}(3.13) & \\ \qquad R_{ikt} \in \mathbb{Z}_{\geq 0} & (i \in V^r;\ k \in \mathcal{R}_i;\ t \in T_i) \\ \qquad X_{it} \in \{0,1\} & (i \in V;\ t \in T_i) \\ \qquad D_{it} \in \{0,1\} & (i \in V^r;\ t \in T_i) \end{cases}$$

For further information, we refer to Baumann and Trautmann (2013).

3.4 MIP-based heuristic

In this section, we present our MIP-based heuristic procedure for scheduling a project with work-content constraints. The heuristic scheme is depicted as a flowchart in Figure 3.2. First, a preprocessing routine is performed that aims at determining tighter lower and upper bounds for the activities' usage of the work-content resource. Subsequently, the activities are ordered in a precedence-feasible list L using either of the priority rules proposed by Fündeling and Trautmann (2010). Then, the activities are scheduled iteratively based on their order in L. The sequential scheduling of the activities enables the construction of a feasible schedule in short CPU time. However, it may also lead to an inefficient resource usage. Hence, each time that the scheduling of an activity increases the makespan of the partial schedule constructed thus far, the resource usage of that activity is examined. If this examination indicates that the resource usage is inefficient, then a subset of activities

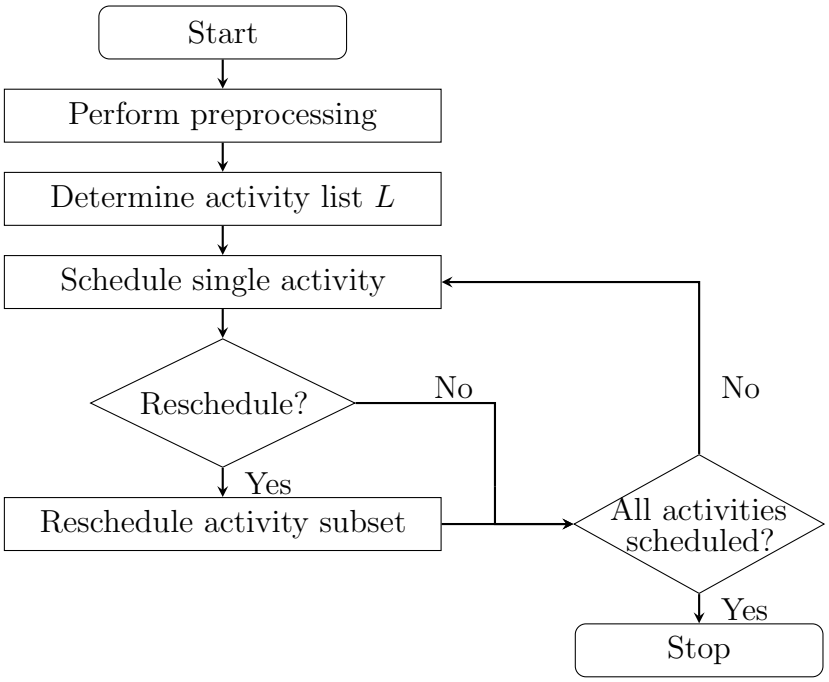


Figure 3.2: Flowchart: overview of the MIP-based heuristic procedure

is rescheduled. By considering several activities simultaneously in this rescheduling step, a more efficient resource usage may be determined. The heuristic procedure stops once all activities have been scheduled.

To determine tighter bounds for the activities' usage of the work-content resource, we solve the mixed-integer linear programs M-P$_{min}$ and M-P$_{max}$ (cf. Subsection 3.4.1). To schedule a single activity and to reschedule each subset of activities, we solve the mixed-integer linear programs M-SI (cf. Subsection 3.4.2) and M-RE (cf. Subsection 3.4.3), respectively. In Subsection 3.4.4, we demonstrate how the heuristic procedure generates an optimal schedule for the illustrative example presented in Section 3.2.

A preliminary version of this heuristic can be found in Zimmermann (2016). That version of the heuristic does not include the preprocessing routine, and the rescheduling step is performed for a fixed number of activities each time a prescribed number of iterations has passed.

In the following, we denote the makespan of the partial schedule by C, and the

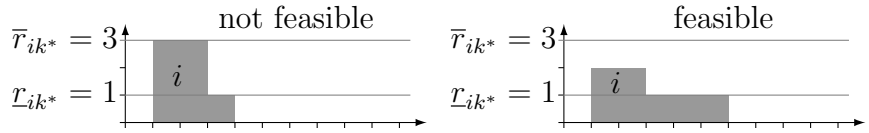


Figure 3.3: Maximum resource usage lower than prescribed upper bound

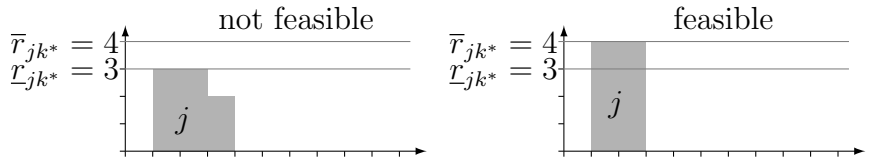


Figure 3.4: Minimum resource usage higher than prescribed lower bound

remaining capacity of resource k in each period t by R_{kt}. Notably, the parameter R_{kt} replaces the parameter R_k in constraints (3.13). At the start of the heuristic procedure, C equals 0, and R_{kt} equals the capacity of the respective resource $k \in \mathcal{R}$. We use C_i to denote the completion time of an activity. Furthermore, we assume that the planning horizon is sufficiently large, i.e., we define $T = \{1, \ldots, \sum_{i \in V^r} \overline{p}_i + 1\}$, where $\overline{p}_i = \lfloor w_i / \underline{r}_{ik^*} \rfloor$ is an upper bound for the duration of an activity i.

3.4.1 Preprocessing

Due to the minimum block length, an activity's usage of the work-content resource k^* might never be as high as the prescribed upper bound $\overline{r}_{ik^*}$ or as low as the prescribed lower bound $\underline{r}_{ik^*}$. For example, we consider two activities, i and j. Activity i has a work content of $w_i = 7$ units, and the bounds are $\overline{r}_{ik^*} = 3$ and $\underline{r}_{ik^*} = 1$. If the minimum block length is 2 periods, then no maximum resource usage of 3 units is possible (see Figure 3.3).

On the other hand, activity j has a work content of $w_j = 8$ units, and the bounds are $\overline{r}_{jk^*} = 4$ and $\underline{r}_{jk^*} = 3$. Again, if the minimum block length is 2 periods, then no minimum resource usage of 3 units is possible (see Figure 3.4).

To calculate the minimum or maximum resource requirements of an activity, the resource usage of that activity must be determined over its entire duration such that all work-content constraints are met and the resource capacities are never exceeded. Fündeling and Trautmann (2010) show that the problem of determining a feasible resource usage of an activity is NP-complete. They formulate this problem as a SUBSET SUM problem and solve it using a dynamic-programming-based approach (cf. Gary and Johnson, 1979). In this work, we employ an MIP-based approach; to calculate the minimum and maximum resource requirements of an activity, the mixed-integer linear programs M-P$_{min}$ and M-P$_{max}$ are solved, respectively. These two programs are solved once for each activity $i \in V^r$.

The program M-P$_{min}$ is based on the following formulation.

$$
(\text{M-P}_{min}) \quad \begin{cases} \text{Min } R_{i^*k^*1} & \\ \text{s.t. } (3.2)\text{–}(3.10), (3.12), (3.13) & \\ \qquad X_{i^*1} = 1 & (3.14) \\ \qquad R_{i^*kt} \in \mathbb{Z}_{\geq 0} & (k \in \mathcal{R}_{i^*};\ t \in T_{i^*}) \\ \qquad X_{i^*t} \in \{0,1\} & (t \in T_{i^*}) \\ \qquad D_{i^*t} \in \{0,1\} & (t \in T_{i^*}) \end{cases}
$$

The objective function minimizes the resource usage of the work-content resource k^* by the activity under consideration i^* in period $t = 1$. From the formulation of Baumann and Trautmann (2013), the constraints (3.2) to (3.10), (3.12), and (3.13) are included. These constraints are formulated only for activity i^*. Furthermore, the resource-capacity constraints, i.e., constraints (3.13), consider the resource usage of only activity i^*. Since the makespan and the precedence relationships do not have to be considered at this point, constraints (3.1) and (3.11) are omitted. Constraint (3.14) ensures that the activity is processed in the first period. The program M-P$_{max}$ is based on the same formulation as M-P$_{min}$, except that the objective function maximizes the resource usage in the first

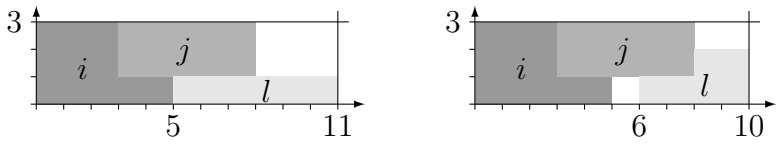


Figure 3.5: Makespan reduction by delaying the start of activity l

period.

The set of relevant time periods for activity i^* is determined as follows. The earliest start time EST_{i^*} is set to 0, and the latest finish time LFT_{i^*} corresponds to the upper bound for the activity duration, i.e., $LFT_{i^*} = \overline{p}_{i^*}$.

Let $\underline{r}_i^*$ and $\overline{r}_i^*$ denote the tightened lower and upper bounds for the usage of the work-content resource as determined by M-P$_{min}$ and M-P$_{max}$, respectively. Constraints (3.4) and (3.5) are replaced by constraints (3.4*) and (3.5*), respectively.

$$\underline{r}_i^* X_{it} \leq R_{ik^*t} \qquad (i \in V^r;\ t \in T_i) \qquad (3.4^*)$$

$$\overline{r}_i^* X_{it} \geq R_{ik^*t} \qquad (i \in V^r;\ t \in T_i) \qquad (3.5^*)$$

3.4.2 Scheduling a single activity

Baumann et al. (2015) show that, due to the minimum-block-length constraint, it may be optimal to delay the start of an activity rather than scheduling it as early as possible. Figure 3.5 presents an example with three activities, i, j, and l; a work-content resource with capacity 3; and a minimum block length of 2 periods. By delaying the start of activity l, the makespan is reduced by one period. Hence, when scheduling a single activity, we set the minimization of its completion time as the objective.

The MIP formulation (M-SI) for scheduling a single activity i^* reads as follows.

$$
\text{(M-SI)} \quad \begin{cases}
\text{Min. } C_{i^*} + \dfrac{1}{LFT_{i^*} w_{i^*}} \displaystyle\sum_{t \in T_{i^*}} t R_{i^* k^* t} & \\
\text{s.t. } (3.2), (3.3), (3.6)\text{–}(3.10), (3.12), (3.13) & \\
\quad (3.4^*), (3.5^*) & \\
\quad C_{i^*} \geq (t-1)(X_{i^*,t-1} - X_{i^*t}) & (t \in T_{i^*} :\ t > EST_{i^*} + 1) \qquad (3.15) \\
\quad R_{i^* kt} \in \mathbb{Z}_{\geq 0} & (k \in \mathcal{R}_{i^*};\ t \in T_{i^*}) \\
\quad X_{i^* t} \in \{0,1\} & (t \in T_{i^*}) \\
\quad D_{i^* t} \in \{0,1\} & (t \in T_{i^*})
\end{cases}
$$

The objective function minimizes the completion time of the activity under consideration i^*. The second term of the objective function minimizes the activity's usage of the work-content resource in the later periods of its execution. An upper bound for the maximum value of the second term is given by multiplying the entire work content by the latest finish time of an activity. The second term is used to reduce the resource usage at the end of the partial schedule. From the formulation of Baumann and Trautmann (2013), the constraints (3.2), (3.3), (3.6) to (3.10), (3.12), and (3.13) are included, and constraints (3.4*) and (3.5*) are defined as described in Section 3.4.1. Similar to the MIP formulations M-P$_{min}$ and M-P$_{max}$, these constraints are formulated only for activity i^*, constraints (3.1) and (3.11) are omitted, and constraints (3.13) consider the resource usage of only activity i^*. Constraints (3.15) determine the completion time of activity i^*.

The set of relevant time periods for activity i^* is determined as follows. The earliest start time of activity i^* is given by the maximum completion time over all its predecessors, i.e., $EST_{i^*} = \max_{j \in P_{i^*}} C_j$. If $P_{i^*} = \emptyset$, then we set $EST_{i^*} = 0$. The latest finish time LFT_{i^*} corresponds to the makespan of the current partial schedule plus the upper bound for the activity duration, i.e., $LFT_{i^*} = C + \overline{p}_{i^*}$.

Once an activity has been scheduled, the resource capacities R_{kt} are updated. If $C_{i^*} > C$,

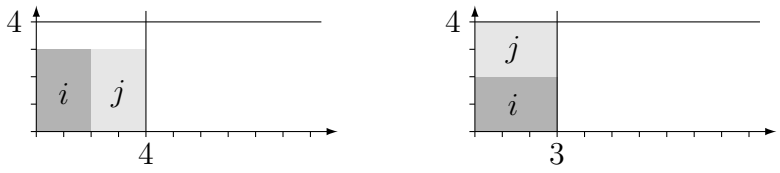


Figure 3.6: Makespan reduction by decreasing the resource usage of activities i and j

then the makespan of the partial schedule is set to $C = C_{i^*}$.

3.4.3 Rescheduling a subset of activities

Fündeling and Trautmann (2010) show that it may be optimal to reduce the resource usage of an activity by increasing its duration because it allows more activities to be executed in parallel. Figure 3.6 depicts an example with two activities, i and j; a work-content resource with capacity 4; and a minimum block length of 2 periods. During the execution of both activities, either 2 or 3 units of the work-content resource must be used. By decreasing the resource usage of activities i and j, both activities can be executed in parallel rather than sequentially, and the makespan is reduced by one period. The inefficient resource usage depicted in Figure 3.6 may result from minimizing the completion times of individual activities, as described in Section 3.4.2. Hence, a rescheduling step is employed that aims to reduce such inefficiencies by considering multiple activities simultaneously.

To determine whether a rescheduling step is performed, the following procedure is applied. Let i^* be the activity that has been scheduled in the current iteration, and let V^C be the set of activities that have been scheduled thus far. We define C' as the makespan of the partial schedule before activity i^* has been scheduled, i.e., $C' = \max_{j \in V^C \setminus \{i^*\}} C_j$, and we define a time window $\overline{T}$ that consists of the periods $\overline{T} = \{\max(1, C' - \overline{p}_{i^*} + 1), \ldots, C'\}$.

The rescheduling step is performed each time the following three conditions are met.

1. By scheduling activity i^*, the makespan of the partial schedule is increased, i.e., $C_{i^*} > C'$.

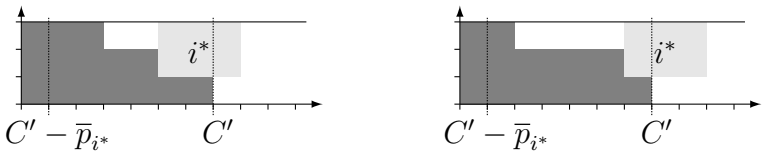


Figure 3.7: Partial schedules that meet the first two conditions (left) and all three conditions (right) that are required for performing a rescheduling step

2. Within the time window $\overline{T}$, the sum of the remaining capacity of the work-content resource is larger than zero, i.e., $\sum_{t\in\overline{T}} R_{k^*t} > 0$.

3. Within the time window $\overline{T}$, the usage of the work-content resource k^* by activity i^* is less than or equal to half of the remaining resource capacity before activity i^* has been scheduled, i.e., $\sum_{t\in\overline{T}} R_{i^*k^*t} \leq 0.5 \sum_{t\in\overline{T}}(R_{k^*t} + R_{i^*k^*t})$. In particular, the current resource usage may be inefficient because no more than 50% of the available resource capacity within the time window $\overline{T}$ was used when scheduling activity i^*. In the case that $\sum_{t\in\overline{T}} R_{i^*k^*t} = \sum_{t\in\overline{T}} R_{k^*t} = 0$, i.e., activity i^* does not use any resource capacities in $\overline{T}$ because there are none available, the secondary condition prevents a rescheduling.

Figure 3.7 depicts two partial schedules. The dark-gray area represents the overall resource usage before activity i^* has been scheduled. In both cases, the first and second conditions are met because the scheduling of activity i^* increases the makespan, and there are still resource capacities available within the time window $\overline{T}$ (white area). However, the third condition is only met by the partial schedule on the right. The available resource capacity before activity i^* has been scheduled is 6 units. In the partial schedule on the left, activity i^* uses 4 units, and the third condition is not met because $4 > 0.5 \cdot 6$. In the partial schedule on the right, activity i^* only uses 2 units, and the condition is met because $2 \leq 0.5 \cdot 6$.

To select the activities for rescheduling, the following procedure is applied. Let V_s be the subset of activities that are selected for rescheduling. The number of activities included

in V_s determines the size of the rescheduling problem. If the size of the rescheduling problem becomes too large, then the mixed-integer linear program might be unable to determine a more efficient resource usage in reasonable CPU time. Hence, only a small number of activities should be considered in the rescheduling step. We define γ as the number of activities initially included in V_s. The subset V_s consists of γ activities with the greatest completion times over all activities that have been scheduled thus far. Additionally, all activities that meet the following conditions are included in V_s. Let $\underline{S}$ be the minimum start time over all γ activities initially included in V_s. If an activity j that is not yet included in V_s is completed immediately before or some time after this minimum start time, i.e., $C_j \geq \underline{S}$, then activity j is also included in V_s.

The MIP formulation (M-RE) for rescheduling the activities in the subset V_s reads as follows.

$$
\text{(M-RE)} \quad \begin{cases}
\text{Min.} \displaystyle\sum_{t \in T_{n+1}} t X_{n+1,t} - 1 + \frac{1}{\sum_{i \in V_s} LFT_i} \sum_{i \in V_s} C_i & \\
\text{s.t.} \;\; (3.1)\text{–}(3.3),\ (3.6)\text{–}(3.10),\ (3.12),\ (3.13) & \\
\qquad (3.4^*),\ (3.5^*) & \\
\qquad C_i \geq (t-1)(X_{i,t-1} - X_{it}) & \\
\qquad\qquad\qquad (i \in V_s;\ t \in T_i :\ t > EST_i + 1) & (3.16) \\
\qquad X_{it} \leq \displaystyle\sum_{t' \in T_j : t' < t} \frac{R_{jk^*t'}}{w_j} & \\
\qquad\qquad\qquad (i \in V_s \cup \{n+1\};\ j \in P_i \cap V_s;\ t \in T_i) & (3.17) \\
\qquad R_{ikt} \in \mathbb{Z}_{\geq 0} \qquad (i \in V_s;\ k \in \mathcal{R}_i;\ t \in T_i) & \\
\qquad X_{it} \in \{0,1\} \qquad (i \in V_s \cup \{n+1\};\ t \in T_i) & \\
\qquad D_{it} \in \{0,1\} \qquad (i \in V_s;\ t \in T_i) &
\end{cases}
$$

The objective function minimizes the makespan of the partial schedule. The dummy activity $n+1$ represents the end of the partial schedule because it can only start after all

activities in the partial schedule have been completed. The second term of the objective function minimizes the sum of the completion times of the activities in V_s. An upper bound for the maximum value of the second term is given by the sum of the latest finish times of all activities in V_s. The second term is used to avoid scheduling uncritical activities that do not affect the makespan at the end of the partial schedule. From the MIP formulation of Baumann and Trautmann (2013), the constraints (3.1) to (3.3), (3.6) to (3.10), (3.12), and (3.13) are included, and constraints (3.4*) and (3.5*) are defined as described in Section 3.4.1. These constraints are formulated only for the activities in V_s and, in the case of constraint (3.1), for the dummy activity $n+1$. Constraints (3.16) determine the completion times of all activities in V_s. Constraints (3.17) ensure that any precedence relationships among the activities in V_s and the dummy activity $n+1$ are satisfied.

To determine the set of relevant time periods for the activities in V_s, the following procedure is applied. For the γ activities with the greatest completion times, the set of relevant time periods is determined as follows. The activities' earliest start time and latest finish time are set to the smallest start time and the greatest finish time over all activities in V_s, respectively. The resulting time windows are further increased by reducing the earliest start times as follows. Let p_i be the duration of activity i in the current partial schedule, i.e., $p_i = \sum_{t \in T_i} X_{it}$. Then, $EST_i = EST_i - (\overline{p}_i - p_i)$. Activities for which the value of $\overline{p}_i - p_i$ is large use more resource units in each period during their execution. By increasing their time windows accordingly, this resource usage might be reduced and more activities can be processed in parallel. For any activity i in V_s that does not belong to the γ activities with the greatest completion times, the earliest start time and the latest finish time are set to their current start time minus 5 periods and their current completion time plus 5 periods, respectively. By limiting the time windows of these activities, the size of the rescheduling problem can be reduced. If the earliest start time violates the precedence relationships with the activities not included in V_s, then the earliest start time is set to $EST_i = \max_{j \in P_i \setminus V_s} C_j$.

If the current makespan C cannot be improved during the rescheduling step, then the current solution is retained. Notably, the current solution is also used as an initial solution for M-RE.

3.4.4 Solution for illustrative example

We illustrate how the MIP-based heuristic procedure generates an optimal schedule for the example presented in Section 3.2. Using the so-called longest path following priority rule of Fündeling and Trautmann (2010) (cf. Subsection 3.5.2), the precedence-feasible list $L = (1, 2, 4, 3, 5, 6, 8, 7, 10, 9)$ is generated. We set $\gamma = 4$. Figure 3.8 depicts the construction of the schedule. After the first five iterations, a first rescheduling step is performed because the scheduling of activity 5 increases the makespan of the partial schedule, but none of the available resource capacities before the completion of activity 3 are used. The activities for rescheduling are selected as follows. First, the four activities with the largest completion times are selected, i.e., activities 2 to 5. Then, because activity 1 has a completion time that is greater than the minimum start time over these four activities, activity 1 is also selected. During the rescheduling of activities 1 to 5, the makespan of the partial schedule is reduced by one period. Notably, the resource usage of activity 4 is reduced such that activities 3 and 5 can start earlier. During the next five iterations, activities 6 to 10 are scheduled. After the scheduling of activities 8 and 10, respectively, a rescheduling step is performed. However, these two rescheduling steps do not lead to a reduction of the makespan. After the scheduling of activity 9, a final rescheduling for activities 6 to 10 is performed. Consequently, the resource usages of activities 6 to 8 are adjusted such that activity 9 is completed before the start of activity 10. The resulting makespan of 45 periods corresponds to the optimal solution, and the procedure required 13 seconds of CPU time.

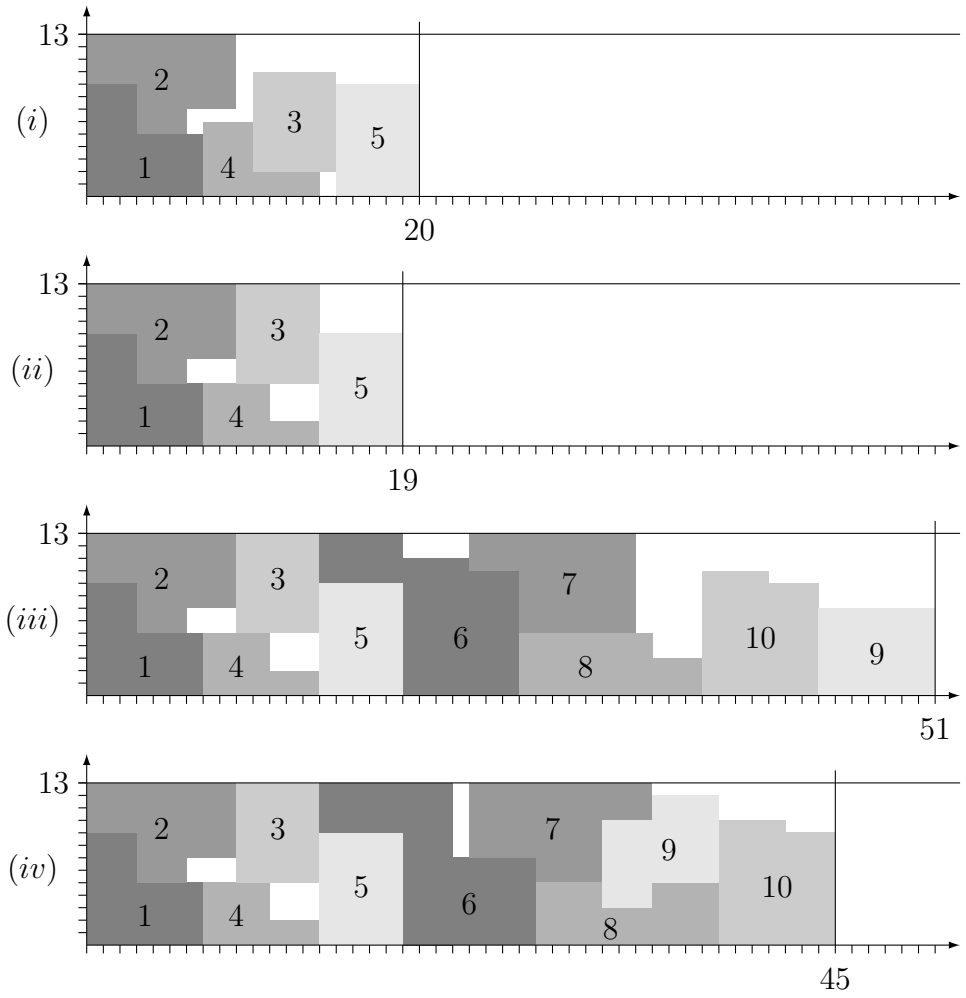


Figure 3.8: Illustrative example: generated (partial) schedule after five iterations (i), after first rescheduling step (ii), after ten iterations (iii), and after final rescheduling step (iv)

3.5 Computational experiment

We applied the MIP-based heuristic procedure presented in Section 3.4 to the set of test instances devised by Fündeling (2006). In Subsection 3.5.1, we describe the test instances. In Subsection 3.5.2, we explain the setting for our computational experiment. In Subsection 3.5.3, we report our numerical results.

3.5.1 Test set

The test set of Fündeling (2006) consists of $5 \cdot 480 = 2400$ instances with 10, 20, 40, 100, and 200 activities. Each instance involves one work-content resource and up to three non-work-content resources. The instances were generated by varying the following complexity factors.

- The order strength OS corresponds to the ratio of the total number of precedence relationships to the maximum number of all possible precedence relationships. The experimental levels $OS \in \{0.25, 0.50, 0.75\}$ were used. For example, $OS = 0.25$ implies that there are few precedence relationships relative to the number of project activities.

- The resource factor RF corresponds to the mean percentage of resources used by an activity. The experimental levels $RF \in \{0.25, 0.50, 0.75, 1.00\}$ were used. For example, $RF = 0.25$ implies that all activities require the work-content resource only.

- The resource strength RS indicates the scarcity of the resources. The experimental levels $RS \in \{0.00, 0.25, 0.50, 0.75\}$ were used. For example, $RS = 0$ implies that for each resource, there exists at least one activity that requires the resource's maximum capacity.

For each combination of complexity factor levels, 10 instances were generated. The minimum block length was randomly chosen from the set $\{2, 3, 4\}$, and the parameter values for w_i, $\overline{r}_{ik^*}$, and $\underline{r}_{ik^*}$ were randomly chosen such that a feasible solution exists for each instance. For further information, we refer to Fündeling (2006).

3.5.2 Experimental design

We implemented our heuristic procedure in AMPL, and we used the Gurobi Optimizer 6.5 to solve M-P$_{min}$, M-P$_{max}$, M-SI, and M-RE. To limit the run time of our heuristic, we set

the CPU time limit for M-RE to 5 seconds. After some preliminary tests, we set the value of γ to 4. By considering a small number of activities simultaneously, M-RE can obtain good feasible solutions within the prescribed time limit. We did not prescribe a time limit for M-P$_{min}$, M-P$_{max}$, and M-SI because the CPU time required to solve these programs is short.

The performance of our heuristic procedure depends on the order of the activities in the precedence-feasible list. Hence, we generated a list with each of the three priority rules proposed by Fündeling and Trautmann (2010). The priority rules are as follows.

- Longest Path Following (LPF) rule: The priority of an activity i is determined by the minimum time lag between its start time and the end of the project. To compute this priority value, the lower bound for the activity duration $\underline{p}_i = \lceil w_i/\overline{r}_{ik^*} \rceil$ is used.
- Most Total Successors (MTS) rule: The priority of an activity i equals the total number of activities that cannot be started before the completion of activity i.
- Most Work content Remaining (MWR) rule: The priority value of an activity i equals its work content plus the work content of all its immediate and indirect successor activities.

If several activities are eligible for the same position in the list, then ties are broken by the lowest activity index. For each list, we applied our heuristic procedure to construct a single schedule. Thus, we obtained three schedules for each instance.

We compare our heuristic procedure to the priority-rule heuristic of Fündeling and Trautmann (2010) and the MIP formulation of Baumann and Trautmann (2013). Fündeling and Trautmann (2010) applied their schedule-generation scheme for each of the three priority rules described above. Furthermore, they employed random priority values in a multi-pass experiment with a CPU time limit of 30 seconds. Their computations were performed on a standard PC with a 3.4GHz Intel Pentium IV CPU. We implemented the MIP of Baumann and Trautmann (2013) in AMPL, and used the Gurobi Optimizer 6.5 as

the solver. Before applying the MIP, we performed the preprocessing routine described in Section 3.4.1. To generate feasible solutions for the instances with more than 10 activities, the CPU time required by the MIP becomes prohibitively long. Hence, we applied the MIP to the 480 instances with 10 activities only. Thereby, we set the solver time limit to 600 seconds. All of our computations were performed on a standard PC with a 2.80GHz Intel i5 CPU and 4GB RAM.

3.5.3 Numerical results

Table 3.3 reports the average results for the test instances with 10 activities. The MIP of Baumann and Trautmann (2013) obtains feasible solutions for 469 instances only. The average gaps for the 469 instances and for all 480 instances are listed in rows [$\varnothing$ ΔLB_{469}] and [$\varnothing$ ΔLB_{480}], respectively. The gap of an instance equals $(MS - LB)/MS$, where MS is the makespan of the schedule generated and LB corresponds to the lower bound proposed by Fündeling and Trautmann (2010). The average CPU time and the number of instances solved to optimality are reported in the last two rows. Columns [BT13] and [FT10] report the results obtained by the MIP of Baumann and Trautmann (2013) and the heuristic of Fündeling and Trautmann (2010), respectively. Columns [LPF], [MTS], and [MWR] present the results obtained by our heuristic procedure using the respective priority rules described in Subsection 3.5.2. Regardless of the priority rule employed, our heuristic generates shorter schedules than FT10 in less CPU time. Furthermore, optimal schedules are obtained for approximately 80% of the instances.

Tables 3.4 to 3.7 report the average results for the test instances with 20, 40, 100, and 200 activities, respectively. On average, for instances with 40 activities or less, our procedure generates schedules with shorter makespans than FT10 in less CPU time. For instances with 100 and 200 activities, our procedure also generates better solutions than FT10. However, the required CPU time is longer.

Table 3.8 reports the effect of the preprocessing routine and the rescheduling step on

Table 3.3: Results for instances with 10 activities

	BT13	FT10	this work		
			LPF	MTS	MWR
⌀ ΔLB_{469} [%]	14.0	16.3	14.4	14.6	14.5
⌀ ΔLB_{480} [%]	–	16.5	14.7	14.9	14.8
⌀ CPU time [sec]	72.0	30.0	4.6	5.2	5.2
# opt.	447	164	391	383	386

Table 3.4: Results for instances with 20 activities

	FT10	this work		
		LPF	MTS	MWR
⌀ ΔLB_{480} [%]	14.6	12.8	13.0	12.9
⌀ CPU time [sec]	30.0	10.4	10.5	10.8

Table 3.5: Results for instances with 40 activities

	FT10	this work		
		LPF	MTS	MWR
⌀ ΔLB_{480} [%]	15.6	12.8	13.1	12.9
⌀ CPU time [sec]	30.0	24.5	23.6	23.9

Table 3.6: Results for instances with 100 activities

	FT10	this work		
		LPF	MTS	MWR
⌀ ΔLB_{480} [%]	16.9	12.6	12.8	12.8
⌀ CPU time [sec]	30.0	68.5	63.8	64.2

Table 3.7: Results for instances with 200 activities

	FT10	this work		
		LPF	MTS	MWR
∅ ΔLB_{480} [%]	16.9	11.6	11.9	11.9
∅ CPU time [sec]	30.0	171.4	155.9	159.0

Table 3.8: Effect of preprocessing and rescheduling

	$n = 10$			$n = 100$		
	complete	no prepro.	no res.	complete	no prepro.	no res.
∅ ΔLB_{480} [%]	14.7	14.9	16.9	12.6	12.8	13.1
∅ CPU time [sec]	4.6	7.7	1.6	68.5	90.8	22.3

the average results for the instances with 10 activities ($n = 10$) and 100 activities ($n = 100$). Because the results are similar for all priority rules, we focus on the LPF rule, which provides the lowest average gaps. Columns [complete], [no prepro.], and [no res.] show the average results when the complete heuristic procedure is applied, when the procedure is applied without the preprocessing routine, and when no rescheduling steps are performed, respectively. In particular, without the preprocessing routine, the required CPU time increases from 4.6 to 7.7 seconds for instances with 10 activities and from 68.5 to 90.8 seconds for instances with 100 activities. The inclusion of the rescheduling steps leads to an increase in the average CPU time, but it also reduces the average gaps from 16.9% to 14.7% for instances with 10 activities and from 13.1% to 12.6% for instances with 100 activities.

3.6 Conclusions and outlook

For the project scheduling problem with work-content constraints, we have devised an MIP-based heuristic procedure that involves two novel MIP formulations. Furthermore, we

presented an MIP-based preprocessing routine that considerably improves the performance of the heuristic. Our computational results for a test set from the literature that consists of instances with up to 200 activities indicate that the heuristic provides very good feasible solutions for small-, medium-, and large-sized instances. For many small-sized instances, the heuristic generates optimal solutions in very short CPU time, and for medium- and large-sized instances, the heuristic provides better solutions than the state-of-the-art method.

In our future research, we will focus on the development of new methods for determining the order in which the activities are scheduled. For example, an MIP-based approach may be used to generate a schedule for a relaxed version of the problem, and the order of the activities may then be determined based on their start times in that schedule. A relaxation of the problem can be achieved by, e.g., omitting the minimum-block-length constraints or by allowing a continuous resource usage. Furthermore, the MIP-based heuristic presented in this work may be applied to similar project scheduling problems in the literature, such as the multi-mode resource-constrained project scheduling problem.

Bibliography

Baumann, P., Fündeling, C.-U., Trautmann, N., 2015. The resource-constrained project scheduling problem with work-content constraints. In: Handbook on Project Management and Scheduling Vol. 1. Springer, Cham, pp. 533–544.

Baumann, P., Trautmann, N., 2013. Optimal scheduling of work-content-constrained projects. In: T. Laosirihongthong, R. Jiao, M. Xie, R. Sirovetnukul (Eds.): Proceedings of the International Conference on Industrial Engineering and Engineering Management. IEEE, Bangkok, pp. 395–399.

Baumann, P., Trautmann, N., 2014. An MILP formulation for scheduling of work-content-constrained projects. In: T. Fliedner, R. Kolisch, A. Naber (Eds.): Proceedings of the International Conference on Project Management and Scheduling. PMS, Munich, pp. 24–27.

Dror, M., Stern, H. I., Lenstra, J. K., 1987. Parallel machine scheduling: processing rates dependent on number of jobs in operation. Management science 33 (8), 1001–1009.

Fırat, M., Hurkens, C. A. J., 2012. An improved MIP-based approach for a multi-skill workforce scheduling problem. Journal of Scheduling 15 (3), 363–380.

Fündeling, C.-U., 2006. Ressourcenbeschränkte Projektplanung bei vorgegebenen Arbeitsvolumina. Gabler, Wiesbaden.

Fündeling, C.-U., Trautmann, N., 2010. A priority-rule method for project scheduling with work-content constraints. European Journal of Operational Research 203 (3), 568–574.

Gary, M. R., Johnson, D. S., 1979. Computers and Intractability: A Guide to the Theory of NP-completeness. WH Freeman and Company, New York.

Hackman, S. T., Leachman, R. C., 1989. An aggregate model of project-oriented production. IEEE Transactions on Systems, Man, and Cybernetics 19 (2), 220–231.

Jozefowska, J., Weglarz, J., 1998. On a methodology for discrete–continuous scheduling. European Journal of Operational Research 107 (2), 338–353.

Kis, T., 2005. A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. Mathematical programming 103 (3), 515–539.

Kovács, A., 2003. A novel approach to aggregate scheduling in project-oriented manufacturing. In: Proceedings of the 13th International Conference on Automated Planning and Scheduling. Doctoral Consortium, Trento, pp. 63–67.

Kuhlmann, A., 2003. Entwicklung eines praxisnahen Project Scheduling Ansatzes auf der Basis von Genetischen Algorithmen. Logos, Berlin.

Márkus, A., Váncza, J., Kis, T., Kovács, A., 2003. Project scheduling approach to production planning. CIRP Annals-Manufacturing Technology 52 (1), 359–362.

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

Ranjbaer, M., Kianfar, F., 2010. Resource-constrained project scheduling problem with flexible work profiles: a genetic algorithm approach. Transaction E: Industrial Engineering 17, 25–35.

Toffolo, T. A. M., Santos, H. G., Carvalho, M. A. M., Soares, J. A., 2016. An integer programming approach to the multimode resource-constrained multiproject scheduling problem. Journal of Scheduling 19 (3), 295–307.

Zimmermann, A., 2016. An MIP-based heuristic for scheduling projects with work-content constraints. In: Proceedings of the International Conference on Industrial Engineering and Engineering Management. IEEE, Bali, to appear.