Models and Algorithms for Scheduling and Resource Allocation

INAUGURALDISSERTATION

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

Nicklas Klein

Betreuer: Prof. Dr. Norbert Trautmann
Professur für Quantitative Methoden der BWL
Departement Betriebswirtschaftslehre
Engehaldenstrasse 4, 3012 Bern

Bern, Dezember 2024

Die Fakultät hat diese Arbeit am 20. Februar 2025 auf Antrag der beiden Gutachter Prof. Dr. Alain Haït und Prof. Dr. Norbert Trautmann als Dissertation angenommen, ohne damit zu den darin ausgesprochenen Auffassungen Stellung nehmen zu wollen.

Contents

Introduction		1
Paper I:	Mixed-integer linear programming for project scheduling under various resource constraints	7
Paper II:	Integer programming for multi-mode resource-constrained project scheduling	36
Paper III:	Flow shops with reentry: The total weighted completion time objective	73

Introduction

Companies today are confronted with an increasingly complex and competitive business environment. Emerging trends such as rising energy costs, stricter sustainability requirements, and a growing shortage of skilled workers have led to an increased scarcity of resources. As a result, it is essential for companies to plan their operations and allocate their resources in the most efficient way. The planning process of allocating scarce resources to operations over time is called scheduling (cf., e.g., Pinedo, 2022). This planning process plays a vital role in many companies across a diverse range of industries, such as construction companies that allocate workers and equipment to various activities within a construction project, software companies that allocate developers and hardware to develop new software, or semiconductor manufacturing companies that allocate the machines in their facility to produce wafers. For many companies, such as the ones described above, optimized scheduling can lead to shorter construction times, faster product launches, and increased output through more efficient use of scarce resources. Optimizing such scheduling and resource-allocation decisions in practice is often highly challenging, i.e., no efficient algorithm or closed-form expression to determine an optimal schedule is known for many such problems. Considerable research effort has been dedicated to addressing these challenges, and many models and algorithms have been developed that either exhibit a strong performance for specific scheduling problems, or offer flexibility to address a wide range of scheduling problems. However, these methods frequently struggle to achieve both of these strengths simultaneously, which remains a major challenge in the literature. Consequently, developing novel models and algorithms for scheduling and resource allocation problems that combine both of these strengths is essential for enhancing cost efficiency and ensuring the profitability of companies.

Two common approaches for solving scheduling problems in practice are mathematical programming (cf., e.g., Bixby & Rothberg, 2007, Achterberg et al., 2020), and priority rules (also referred to as dispatching rules, cf., e.g., Haupt, 1989, Pinedo, 2022). When applying an optimization approach based on mathematical programming, the planning problem is formulated as a mathematical model, which comprises decisions to be made (e.g., when to start an activity within a project), constraints these decisions must comply with (e.g., the availability of workers), and an objective function that should be minimized (e.g., project completion time) or maximized (e.g., factory output). This model is then solved using software known as a mathematical programming solver for given data that is specific to the planning situation at hand. For mathematical models to be effective in practical scheduling, they must be flexible to accommodate the dynamically changing requirements of companies, such as the varying conditions between different construction projects or the software needs of clients in different industries. At the same time, solver performance can vary significantly with the model formulation chosen (cf., e.g., Koné et al., 2011, Vielma, 2015), highlighting the need to develop and evaluate flexible mathematical models with good computational performance for scheduling problems. When applying an optimization approach based on priority rules, the operations are sorted according to a predefined measure and subsequently scheduled in that order. In a semiconductor manufacturing company, the different wafers may be scheduled according to the ratio of their importance to their processing time requirement. The major advantages of this approach are its simplicity and explainability, which make it highly scalable and easily applicable in practice. Despite this simplicity, well-designed priority rules frequently devise good schedules regarding various objectives, highlighting the need to develop, evaluate, and apply novel priority rules to emerging scheduling problems.

This thesis is composed of three papers on modeling and solving challenging scheduling problems. We address two project scheduling problems and one production scheduling problem. For the project scheduling problems, we develop novel mathematical models that combine flexibility to handle different types of activities and resource constraints with simplicity and interpretability in practical project contexts. In extensive computational experiments, these models consistently outperform the state-of-the-art models across a wide range of problem instances. For the production scheduling problem, we propose novel priority rules and dedicated algorithms. Through a theoretical analysis, we show that these priority rules and algorithms are guaranteed to devise optimal or near-optimal schedules for all problem instances. The models and algorithms developed in this thesis have in common that they are flexible, with computational and theoretical analyses demonstrating a relatively good performance across problem instances with varying characteristics.

In the first paper, we consider the scheduling of a project, where the project activities require time and capacities of renewable resources, e.g., personnel or equipment, and of storage resources, e.g., funds, energy, or utilities (cf., e.g., Agha et al., 2010), for execution. The problem comprises determining the start times of all project activities such that the project completion time is minimized, while completion-start precedences between pairs of activities are respected, the availabilities of the renewable resources are never exceeded, and the stock of all storage resources never falls below a given threshold. Such a problem arises, e.g., in a long-term construction project, where sufficient personnel and equipment must be allocated to the project activities (renewable resources), but also project milestones, frequently linked to interim payments, must be achieved regularly to ensure the company's liquidity. For this problem, we propose a novel mathematical model based on a novel type of start-start sequencing variables. The main advantages of this sequence-based model are that its variables can be easily interpreted in the problem context and are sufficient to simultaneously capture the renewable and storage resources, whereas all reference models require additional variables to capture the storage resources. In a computational experiment, the proposed model consistently outperformed the state-of-the-art models for a wide range of benchmark instances. In particular, the model consistently devised project schedules with a relatively short project duration for the most challenging problem instances, for which all reference models struggled to devise resource-feasible project schedules.

In the second paper, we extend the sequence-based model to address a project scheduling problem where the project activities can be executed in multiple modes. These execution modes represent different ways to execute a project activity and can reflect trade-offs between execution time and resource requirements, e.g., a software company can either assign a task to a junior developer or to a senior developer who can complete the task faster but is unavailable for other tasks during that time. In a computational experiment, the sequence-based model outperformed the state-of-the-art models for a wide range of benchmark instances, especially instances comprising a relatively large number of project activities, resources, and execution modes, for which all reference models struggle to devise resource-feasible project schedules with a short project duration. The obtained results demonstrate that the developed sequence-based model provides great flexibility to address a wide range of challenging project scheduling problems, while exhibiting a favorable computational performance compared to the state-of-the-art models.

In the third paper, we consider a production scheduling problem where the products require multiple loops through a flow line. The planning problem is to schedule all required loops of all products while minimizing the total weighted completion time of the products. This problem frequently arises in semiconductor manufacturing, where multiple layers of material are sequentially placed on the same wafer. Since some of the machines that are required to produce these wafers cost more than 100 million US Dollars (cf., e.g., Tembey et al., 2023), it is essential that these machines are utilized in the most efficient way. Therefore, products in different stages of the production cycle must be produced on the same machines, which poses significant challenges. For this problem, we propose a novel and simple priority rule that can be used to devise feasible production schedules in a relatively short time. We show that, under certain conditions, this priority rule can even devise optimal schedules. Furthermore, we show that the theoretical worst-case deviation from the optimum is about 20%. This analysis of the worst-case deviation guarantees manufacturers using this priority rule in practice that the rule will devise near-optimal production schedules regardless of potential variations in product or factory characteristics. In a computational experiment, we observed that despite its simplicity, the total weighted completion time of schedules devised using this rule is, on average, only about 1% larger than the optimum. Finally, we present an exact algorithm and an approximation scheme for this problem, which can be used to devise schedules that are arbitrarily close to the optimum.

While this thesis focuses on three specific scheduling problems, the proposed models and priority rules can be adapted, with minor modifications, to address related scheduling problems. An example of such a related problem is resource-constrained project scheduling with minimal and maximal time lags that generalize the precedence relations between pairs of activities (cf., e.g., de Azevedo et al., 2021). For this problem, the novel start-start sequencing variables can be used to efficiently model positive time lags between pairs of activities, which may enhance the computational performance of solvers. Another related problem is the resource investment problem (cf., e.g., Zhu et al., 2017). In this problem, a project schedule is sought that completes the project within a given deadline while minimizing the investment cost for renewable resources. A third related problem are stochastic flow shops with reentry. In this problem, the number of passes through the flow line is not deterministically known in advance, but each time a product has passed through the flow line, it has a probability of reentry, e.g., to repair a detected defect. For this problem, a variant of the proposed priority rule that takes into account the expected number of passes through the flow line may minimize the total expected completion time of the products.

Bibliography

Achterberg, T., Bixby, R., Gu, Z., Rothberg, E., & Weninger, D. (2020). Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, *32*, 473–506. doi:10.1287/ijoc.2018.0857.

Agha, M. H., Thery, R., Hetreux, G., Haït, A., & Le Lann, J. M. (2010). Integrated production and utility system approach for optimizing industrial unit operations. *Energy*, *35*, 611–627. doi:10.1016/j.energy.2009.10.032.

Bixby, R., & Rothberg, E. (2007). Progress in computational mixed integer programming—A look back from the other side of the tipping point. *Annals of Operations Research*, *149*, 37–41. doi:10.1007/s10479-006-0091-y.

de Azevedo, G. H. I., Pessoa, A. A., & Subramanian, A. (2021). A satisfiability and workload-based exact method for the resource constrained project scheduling problem with generalized precedence constraints. *European Journal of Operational Research*, *289*, 809–824. doi:10.1016/j.ejor.2019.07.056.

Haupt, R. (1989). A survey of priority rule-based scheduling. *OR Spektrum*, *11*, 3–16. doi:10.1007/BF01721162.

Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research*, *38*, 3–13. doi:10.1016/j.cor.2009.12.011.

Pinedo, M. (2022). *Scheduling: Theory, Algorithms, and Systems*. (6th ed.). Cham: Springer International Publishing. doi:10.1007/978-3-031-05921-6.

Tembey, G., Dahik, A., Richard, C., & Rastogi, V. (2023). Navigating the costly economics of chip making. URL: https://www.bcg.com/publications/2023/navigating-the-semiconductor-manufacturing-costs (Accessed: December 2024).

Vielma, J. P. (2015). Mixed integer linear programming formulation techniques. *SIAM Review*, *57*, 3–57. doi:10.1137/130915303.

Zhu, X., Ruiz, R., Li, S., & Li, X. (2017). An effective heuristic for project scheduling with resource availability cost. *European Journal of Operational Research*, *257*, 746–762. doi:10.1016/j.ejor.2016.08.049.

Paper I

Mixed-integer linear programming for project scheduling under various resource constraints[1]

Nicklas Klein Mario Gnägi Norbert Trautmann
Department of Business Administration
University of Bern

Contents

1.1 Introduction **8**
1.2 Planning situation **11**
1.3 MILP models from the literature **14**
1.4 Novel MILP model **16**
1.4.1 Variables used in the novel MILP model 16
1.4.2 Novel MILP model 18
1.4.3 Extended MILP models 20
1.5 Computational results **22**
1.5.1 Experimental design 22
1.5.2 Model configuration 25
1.5.3 Results for set J30-CPR and variants 26
1.5.4 Results for set CV-CPR and variants 29
1.5.5 Scalability 31
1.6 Conclusion and outlook **32**
Bibliography **33**

[1]Published in European Journal of Operational Research 319(1), pp. 79–88 (https://doi.org/10.1016/j.ejor.2024.06.036).

Abstract

Project scheduling is an important management task in many companies across different industries. Generally, projects require resources, such as personnel or funds, whose availabilities are limited, giving rise to the challenging problem of resource-constrained project scheduling. In this paper, we consider the scheduling of a project consisting of precedence-related activities that require time and two types of resources for execution: storage resources representing, e.g., the project budget; and renewable resources representing, e.g., personnel or equipment. Storage resources are consumed by activities at their start or produced upon their completion, while renewable resources are allocated to activities at their start and released upon their completion. The resource-constrained project scheduling problem with consumption and production of resources (RCPSP-CPR) consists of determining a minimum-makespan schedule such that all precedence relations are respected, the demand for each renewable resource never exceeds its capacity, and the stock level of each storage resource never falls below a prescribed minimum. Due to the consideration of storage resources, the feasibility variant of this problem is NP-complete. We propose a novel compact mixed-integer linear programming (MILP) model based on a novel type of sequencing variables. These variables enable us to identify which activities are processed in parallel and whether a sequencing of activities is necessary to respect the resource capacities. Our computational results indicate that our novel model significantly outperforms state-of-the-art MILP models for all considered scarcity settings of the storage resources. Additionally, our results indicate a superior performance for instances of the well-known resource-constrained project scheduling problem (RCPSP).

1.1 Introduction

A vital aspect of project management is project scheduling, i.e., the allocation of resources to the processing of project activities over time (cf., e.g., Tavares, 2002). If the availability of project resources is limited, this task is referred to as resource-constrained project scheduling (cf., e.g., Brucker et al., 1999). A large portion of the resource-constrained project scheduling literature focuses on renewable resources, which are used by activities while they are in progress and released at completion (e.g., personnel or machinery), and nonrenewable resources, which are consumed by the activities but are not restored (e.g., raw materials). In addition, various researchers have considered the consumption and production of resources (cf., e.g., Neumann & Schwindt, 2003, Laborie, 2003, Carlier et al., 2009, Carlier & Moukrim, 2015). This concept generalizes renewable and nonrenewable resources by assuming that the activities consume a certain quantity of resources at their start or produce a possibly different quantity upon completion. These resources are called storage resources or cumulative resources (cf., e.g., Neumann & Schwindt, 2003). At the project start, for each storage resource, an initial quantity is available for consumption. A typical example of such a resource is a project budget from which the activities require an initial investment or which they increase upon completion. Moreover, storage resources are needed to model utilities built and needed in a manufacturing process (cf., e.g., Agha et al., 2010). A technique that has recently become increasingly relevant for solving project scheduling problems is mixed-integer linear programming (MILP) (cf., e.g., Artigues et al., 2015). The motivation for using MILP for project scheduling is threefold. First, the performance of mathematical programming solvers and computer hardware has improved substantially in recent years; e.g., Koch et al. (2022) report a total speedup factor of up to 1,000 for MILP solvers in the past 20 years. While dedicated exact procedures might still yield better performance, modern solvers can currently solve many problem instances within seconds that earlier versions cannot solve within any reasonable time (cf. Koch et al., 2022). Second, off-the-shelf solvers might be one of the only handy optimization tools available to practitioners. Third, MILP models might offer greater flexibility to represent the dynamically changing characteristics of business applications (e.g., changing objectives or additional project-specific constraints) compared to that of tailored algorithms.

The resource-constrained project scheduling problem with consumption and production of resources (RCPSP-CPR) can be described as follows (cf., e.g., Koné et al., 2013). A set of activities interconnected by completion-start precedence relations is given. For execution, the activities require time and capacities of storage and renewable resources. The RCPSP-CPR consists of determining start times for all activities such that the precedence relations and given renewable resource capacities are respected, and the storage resources' stock never falls below a prescribed minimum level; the objective is to minimize the project completion time, i.e., the makespan, which is typical for, e.g., research and development projects, where the time-to-market is essential. Because storage resources must be considered, determining whether a feasible schedule exists represents an NP-complete decision problem (cf. Carlier et al., 2009, Carlier & Moukrim, 2015).

In the literature, several MILP models have been proposed for the RCPSP-CPR. These models can be divided into two categories: discrete-time (DT) and continuous-time (CT) models. DT models rely on splitting the scheduling horizon into intervals of equal length and allowing activities to start only at the beginning of those intervals. In contrast, CT models allow activities to start at any time during the scheduling horizon. To our knowledge, there are two DT and two CT MILP models for the RCPSP-CPR in the literature. These models were proposed by Koné et al. (2013) and are based on well-established MILP models for the resource-constrained project scheduling problem (RCPSP), which comprises renewable resources only. The authors extend these models to the RCPSP-CPR by including additional variables and constraints to model storage resources. Moreover, Koné et al. (2013) propose novel test sets for the RCPSP-CPR as extensions of benchmark instances for the RCPSP. In their extensive computational study, Koné et al. (2013) compare the performance of the various MILP models for these novel test sets. The results of their computational study indicate that room for improvement remains regarding the number of instances of moderate size that can be solved to optimality in a reasonable time. Furthermore, they did not identify a unique model that performs best for all sets of instances, which is desirable when implementing such a model in practice.

In this paper, we propose a novel continuous-time MILP model for the RCPSP-CPR. The novel model is compact, i.e., it includes a polynomial number of variables and constraints, and is based on two kinds of sequencing variables. The first type is a completion-start sequencing variable that has been used by other CT models in the literature (cf., e.g., Artigues et al., 2003, Trautmann et al., 2018). The second type is a novel start-start sequencing variable that, to our knowledge, is used for the first time to address resource-constrained project scheduling problems. We combine these two types of variables to identify which activities are processed in parallel and, therefore, must not exceed the renewable resource capacities. Furthermore, the combination of these two variable types enables us to track the stock changes of storage resources, and, therefore, the model does not use any additional variables to handle the storage resource constraints. Another advantage of this new variable type is that one can easily interpret its meaning in the problem context, which is important when explaining a solution approach to practitioners or when modeling additional project-specific restrictions. To further enhance the performance of the model, we add valid inequalities to our formulation. While some of these inequalities are based on the completion-start sequencing variables and are known to improve the performance of continuous-time RCPSP models (cf., e.g., Gnägi et al., 2018, Trautmann et al., 2018), we also propose three novel valid inequalities based on the start-start sequencing variables. In our computational study, we first compare the performance of various configurations of our novel model and identify the best setting. Then, we compare our novel model with the four MILP models from the literature. To analyze the impact on the performance of all models, we examine three scenarios with different levels of storage resource scarcity. In the first scenario, we study the instances proposed by Koné et al. (2013); in these instances, the initial stock of the storage resources can be considered relatively high. In the second scenario, we consider a variant of these instances, in which we have reduced the initial stock. In the third scenario, we investigate another variant of these instances, in which we consider the renewable resources only but not the storage resources, i.e., the same setting as in the RCPSP. In addition, we consider these three scenarios for correspondingly modified instances of the set CV proposed by Coelho & Vanhoucke (2020); with respect to renewable resources, CV instances are considered more challenging to solve than the instances proposed by Koné et al. (2013). Our computational results indicate that our novel model substantially outperforms all reference models and handles the additional challenges of the storage resources best. For all studied instances and scenarios, the novel model solves substantially more instances to feasibility and to optimality than any reference model; in many cases, the reference models struggle to devise feasible project schedules. Notably, our results also indicate that the novel model outperforms the reference models for benchmark instances of the RCPSP of set J30 (cf. Kolisch & Sprecher, 1996) and of set CV (cf. Coelho & Vanhoucke, 2020). Furthermore, we analyze the scalability of the models by using modified instances of set J60 that comprise a relatively large number of activities (cf. Kolisch & Sprecher, 1996). Our results indicate that our novel model scales well toward these larger instances, finding the most proven optimal and best feasible solutions among all models.

The remainder of this paper is structured as follows. In Section 1.2, we describe the problem setting in detail and provide an illustrative example to highlight the impact of storage resources on the complexity of constructing a feasible project schedule. In Section 1.3, we review the MILP reference models for the RCPSP-CPR from the literature and briefly summarize the different types of variables used. In Section 1.4, we explain the novel variable type and our novel MILP model in detail. In Section 1.5, we present our computational study. In Section 1.6, we provide some conclusions and provide an outlook on possible future research directions.

1.2 Planning situation

In this section, we describe the RCPSP-CPR in detail and provide an illustrative example of the planning situation. In the remainder of this paper, we use the notation for sets and parameters summarized in Table 1.1.

V	Set of all activities ($V = \{0, 1, \ldots, n, n+1\}$)
E	Set of precedence relations $(i, j) \in V \times V$
TE	Transitive closure of E
R	Set of renewable resources
C	Set of storage resources
R_k	Capacity of renewable resource $k \in R$
C_k	Initial stock of storage resource $k \in C$
p_i	Duration of activity $i \in V$
r_{ik}	Demand of activity $i \in V$ of renewable resource $k \in R$
c^-_{ik}	Consumption of activity $i \in V$ of storage resource $k \in C$
c^+_{ik}	Production of activity $i \in V$ of storage resource $k \in C$

Table 1.1: Notation for sets and parameters

In the RCPSP-CPR, we are given a set of n real activities. To model the project start and project completion, we additionally consider the two fictitious activities 0 and $n+1$, respectively. We denote the set of all activities as $V = \{0, 1, \ldots, n, n+1\}$. Each activity $i \in V$ has a duration p_i and requires r_{ik} units of each renewable resource $k \in R$ during its execution; i.e., the activity occupies r_{ik} units while it is in progress and releases them upon its completion. The capacity of renewable resource $k \in R$ is denoted as R_k; i.e., at most R_k units of the renewable resource can be used simultaneously. For each storage resource $k \in C$, the activity consumes c^-_{ik} units at its start and produces c^+_{ik} units upon completion. An initial stock $C_k \geq 0$ that is available at the project start is given for each storage resource $k \in C$, and, without loss of generality, we assume the minimum stock level to be zero. We note that storage resources can be considered a generalization of renewable resources: A renewable resource could be modeled as a storage resource whose initial stock is equal to the capacity of the renewable resource and for which each activity produces the amount that it consumes. However, since this is not computationally more efficient, we model renewable resources as a separate resource type in the remainder of this paper. The two fictitious activities 0 and $n+1$ have a duration of zero and do not require any resources, i.e., $p_i = 0$, $r_{ik} = 0$ $(k \in R)$, and $c^-_{ik} = c^+_{ik} = 0$ $(k \in C)$ for $i \in \{0, n+1\}$. Furthermore, a set of completion-start precedence relations $E \subsetneq V \times V$ is given. For each pair of activities $(i, j) \in E$, this precedence relation prescribes that activity i must be completed before activity j starts. TE denotes the transitive closure of these precedence relations; i.e., if $(i, j) \in E$ and $(j, h) \in E$, then the pair (i, h) is also included in TE. The RCPSP-CPR consists of determining a start time for each activity such that the total project duration is minimized, all precedence relations are considered, for each renewable resource at each point in time, the total requirement does not exceed its capacity, and for each storage resource, the stock level never falls below zero.

The RCPSP-CPR represents an NP-hard optimization problem as it contains the RCPSP as a special case (cf. Blazewicz et al., 1983). When considering renewable resources only, it is easy to devise a feasible solution. However, when storage resources are involved, determining whether a feasible schedule exists represents an NP-complete decision problem (cf. Carlier et al., 2009, Carlier & Moukrim, 2015).

We illustrate the RCPSP-CPR with an example consisting of $n = 4$ real activities that require capacities of one renewable resource $k = 1$ and one storage resource $k = 2$, i.e., $R = \{1\}$ and $C = \{2\}$. The renewable resource has a capacity of $R_1 = 3$ units, and the initial stock of the storage resource is $C_2 = 4$. The project start and the project completion are represented by the two fictitious activities 0 and $n + 1$, respectively. The prescribed precedence relations are visualized in the activity-on-node graph in Figure 1.1. Each node i of this graph represents an activity $i \in V$, and each arc (i, j) in this graph represents a precedence relation $(i, j) \in E$. The activity durations, renewable resource requirements, and storage resource consumptions and productions are displayed in Table 1.2.

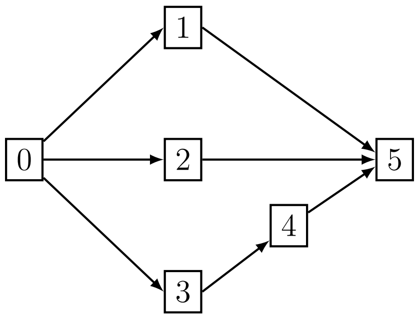


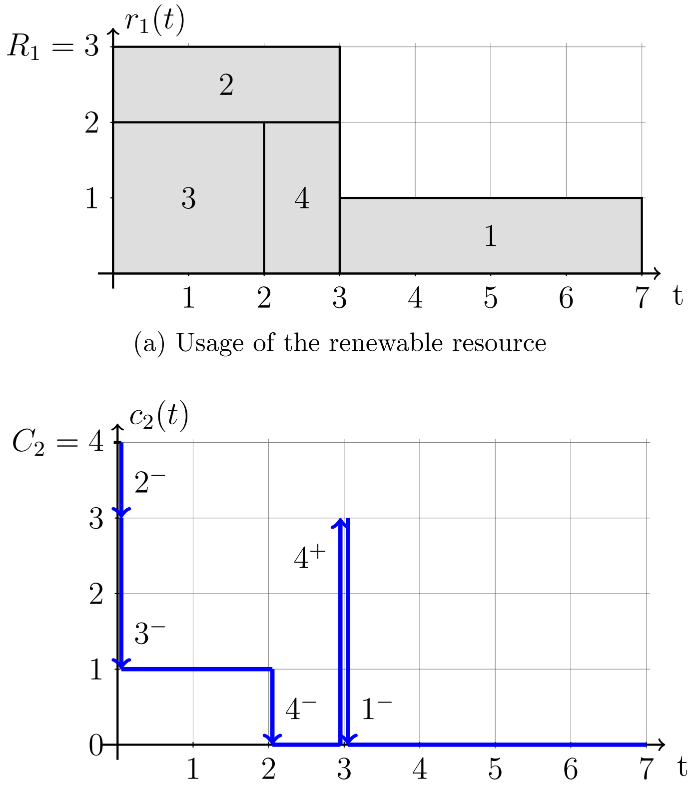
Figure 1.1: Illustrative example — activity-on-node graph

Table 1.2: Illustrative example — duration and resource usages

i	p_i	r_{i1}	c^-_{i2}	c^+_{i2}
1	4	1	3	0
2	3	1	1	0
3	2	2	2	0
4	1	2	1	3

The minimal makespan of this example project is seven; an optimal schedule is depicted in Figure 1.2. For each point in time t, we visualize the usage $r_1(t)$ of renewable resource $k = 1$ and stock level $c_2(t)$ of storage resource $k = 2$ in Figures 1.2a and 1.2b, respectively.

To illustrate how the storage resource constraints can impact the feasibility of a schedule, we describe how the optimal solution changes if we neglect the storage resource constraints. An optimal solution with the corresponding renewable resource usage $\bar{r}_1(t)$ is depicted in Figure 1.3. To achieve the minimal makespan of five, activity one must start at time zero. However, when starting activity one before activity four is completed, there is no feasible solution for the RCPSP-CPR instance. This stands in contrast to the scheduling problem comprising only renewable resources, where any given order leads to a (potentially suboptimal) feasible solution.



(a) Usage of the renewable resource

(b) Stock level of the storage resource

Figure 1.2: Illustrative example — optimal solution with storage resource

1.3 MILP models from the literature

In this section, we summarize the types of variables that are used in the two DT and the two CT MILP models for the RCPSP-CPR proposed by Koné et al. (2013) for the RCPSP-CPR.

Discrete-time models The DT models for the RCPSP-CPR are based on the DT models for the RCPSP that were introduced by Pritsker et al. (1969) and Christofides et al. (1987). We denote the extended models as DT and DDT, respectively. These two models are very similar; they differ only in their formulation of the precedence relation constraints. We therefore provide a combined explanation for them here. To split the scheduling horizon into intervals of equal length, an upper bound T on the minimal makespan is needed. For each activity $i \in V$ and each point in time $t \in \{0, 1, \ldots, T\}$, the models use a binary variable to indicate whether activity i starts at time t. These binary "pulse" variables are used to model the precedence relations and the renewable resource capacities, but they do not provide sufficient information to track the stock changes of the storage resources. Therefore, the models additionally include time-indexed continu-

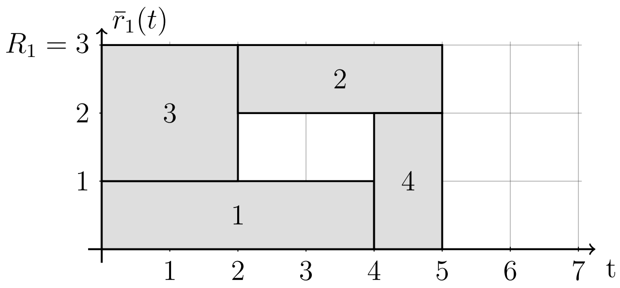


Figure 1.3: Illustrative example — optimal solution without storage resources

ous variables for each storage resource to track their available amounts at each point in time. In total, the models use $\mathcal{O}(T \cdot n)$ binary variables to represent the start times and $\mathcal{O}(T \cdot |C|)$ additional continuous variables to model the storage resources. Consequently, the total number of variables increases considerably when considering the additional storage resources, especially when the activity durations (and thus the scheduling horizon) are long.

Continuous-time models The first CT model is based on the resource flow model from Artigues et al. (2003) and is denoted as FCT in the following. Both renewable and storage resources are represented as flows that might occur between pairs of activities. Specifically, for each activity pair and each resource, there is a continuous flow variable. Each activity has a minimum total inflow (consumption) and a maximum total outflow (production). The respective initial resource capacities are represented by an outflow of fictitious activity 0. To allow a resource flow from activity i to activity j, activity i must be completed before the start of activity j. This condition is ensured through a binary completion-start sequencing variable for each pair of activities $i, j \in V (i \neq j)$ that is equal to one if and only if activity i is completed before the start of activity j. Therefore, the FCT model requires $\mathcal{O}(n^2 \cdot |R|)$ variables for renewable resources and $\mathcal{O}(n^2 \cdot |C|)$ additional variables for storage resources.

The second CT model, denoted by OOE in the following, is based on the concept of events, particularly the on/off event-based RCPSP model proposed by Koné et al. (2011). In this model, the start of one or more activities is considered an event, which makes the number of activities an upper bound to the number of needed events. For each activity and each event, the model includes a binary variable that indicates whether the activity starts or is still in process at that event. Furthermore, a continuous variable indicates the start time for each event. These variables are sufficient to model the renewable resource constraints. However, additional variables are required to represent the storage resources. Koné et al. (2013) introduce two continuous variables for each activity, event, and storage

Model	Number of Variables		Number of Constraints	
	P-RR	add. CPR	P-RR	add. CPR
DT	$\mathcal{O}(T \cdot n)$	$\mathcal{O}(T \cdot \lvert C \rvert)$	$\mathcal{O}(T \cdot \lvert R \rvert + n^2)$	$\mathcal{O}(T \cdot \lvert C \rvert)$
DDT	$\mathcal{O}(T \cdot n)$	$\mathcal{O}(T \cdot \lvert C \rvert)$	$\mathcal{O}(T \cdot \lvert R \rvert + n^2 \cdot T)$	$\mathcal{O}(T \cdot \lvert C \rvert)$
FCT	$\mathcal{O}(n^2 \cdot \lvert R \rvert)$	$\mathcal{O}(n^2 \cdot \lvert C \rvert)$	$\mathcal{O}(n^3 + n^2 \cdot \lvert R \rvert)$	$\mathcal{O}(n^2 \cdot \lvert C \rvert)$
OOE	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2 \cdot \lvert C \rvert)$	$\mathcal{O}(n^3 + n \cdot \lvert R \rvert)$	$\mathcal{O}(n^2 \cdot \lvert C \rvert)$
Novel CT model	$\mathcal{O}(n^2)$	0	$\mathcal{O}(n^2 + n \cdot \lvert R \rvert)$	$\mathcal{O}(n \cdot \lvert C \rvert)$

Table 1.3: Order of magnitude of variables and constraints in the MILP models

resource. These variables correspond to the activity's production (or consumption) of the resource at that event. Moreover, the model includes a continuous variable for each event and each storage resource to track the current stock of the storage resource at the event. Therefore, the OOE model requires $\mathcal{O}(n^2)$ variables for renewable resources and $\mathcal{O}(n^2 \cdot |C|)$ additional variables for storage resources.

The order of magnitude of variables and constraints needed for modeling the precedence relations and renewable resources (P-RR) and the additional consideration of the storage resources (add. CPR) are summarized in Table 1.3. The novel model presented in this paper is the only model that does not require additional variables for the storage resource constraints. Furthermore, the novel model is compact, i.e., the number of variables and constraints can be described by a polynomial in the number of activities and resources.

1.4 Novel MILP model

In this section, we introduce the novel continuous-time MILP model for the RCPSP-CPR. In Subsection 1.4.1, we introduce the variables used, including the novel type of sequencing variables. In Subsection 1.4.2, we present the model in detail. In Subsection 1.4.3, we first transfer some well-known valid inequalities from the literature to the model, and then, we add some valid inequalities that we can formulate by using the novel sequencing variables.

1.4.1 Variables used in the novel MILP model

As is common for CT models, we use a continuous variable S_i for each activity $i \in V$ to indicate the respective start time. To model both types of resource constraints, at the start of each activity, we determine which activities have already been completed and which activities are currently in progress. For this purpose, we introduce two types of binary sequencing variables for certain pairs of activities. The first type is denoted as y_{ij} $(i, j \in V : i \neq j)$, with $y_{ij} = 1$ if activity i is completed at any time before or at the same time as activity j starts and $y_{ij} = 0$ otherwise. This variable is called a completion-start sequencing variable and is also used to model the completion-start precedence relations. The second type is the novel start-start sequencing variable, denoted as z_{ij} $(i, j \in V : i \neq j, (i, j) \notin TE, (j, i) \notin TE)$, with $z_{ij} = 1$ if activity i starts at any time before or at the same time as activity j starts and $z_{ij} = 0$ otherwise. Since the sequencing is prespecified for any pair of activities that have a given precedence relation, we do not include the variables z_{ij} and z_{ji} for any pair of activities $(i, j) \in TE$. At the start of any activity i, we can then determine whether another activity j is already completed $(z_{ji} = y_{ji} = 1)$, is currently in progress $(z_{ji} = 1,\ y_{ji} = 0)$, or has not yet been started $(z_{ji} = y_{ji} = 0)$. We summarize the notations for the variables used in Table 1.4.

Variable	Description
S_i	Start time of activity i
y_{ij}	$\begin{cases} = 1, & \text{if activity } i \text{ is completed before the start of activity } j \\ = 0, & \text{otherwise} \end{cases}$
z_{ij}	$\begin{cases} = 1, & \text{if activity } i \text{ starts before or at the same time as activity } j \\ = 0, & \text{otherwise} \end{cases}$

Table 1.4: Variables used in the novel MILP model

We explain the novel variable type by using the example from Section 1.2; the values of the variables for all real activities in the optimal solution (cf. Figure 1.2) are displayed in Figure 1.4. At the start of, e.g., activity four, for each other activity, we check whether it has already been completed, is currently in progress, or has not yet started. Activity two has already been started ($z_{24} = 1$) but is not completed ($y_{24} = 0$), i.e., it is in progress at the start of activity four. Therefore, we must consider its renewable resource requirement and storage resource consumption in combination with activity four. Since there is a precedence relation between activities three and four ($(3, 4) \in E$), activity three is already completed ($y_{34} = 1$); therefore, we do not have to consider its renewable resource requirement. However, we must account for the difference between storage resource consumption and production to determine the currently available stock of storage resources. Activity one has not yet been started ($z_{14} = 0$) and is hence not completed ($y_{14} = 0$); therefore, we do not have to consider its renewable resource requirements and storage resource consumption or production.

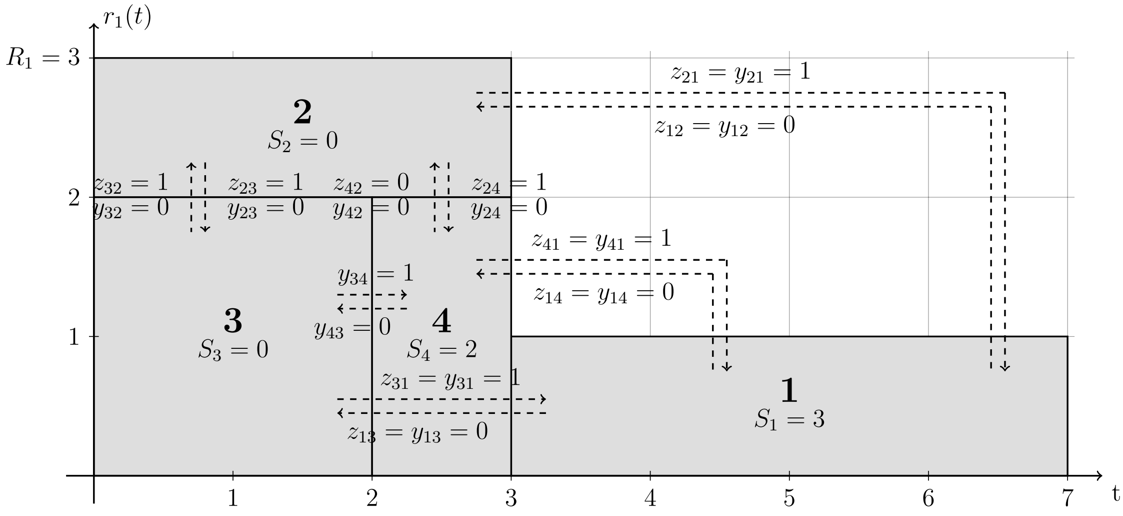


Figure 1.4: Illustrative example — values of the variables in the optimal solution

1.4.2 Novel MILP model

In this subsection, we describe the novel CT model in detail. The objective is to minimize the project makespan, i.e., the start of fictitious activity $n + 1$.

$$\text{min. } S_{n+1} \tag{1.1}$$

Constraints (1.2) ensure that the start of activity j occurs after the completion of activity i if the completion-start sequencing variable y_{ij} is set to one. Hereby, the constant T represents the scheduling horizon, i.e., an upper bound on the minimal makespan; we set $T := \sum_{i \in V} p_i$.

$$S_i + p_i \leq S_j + T(1 - y_{ij}) \qquad (i, j \in V : \; i \neq j) \tag{1.2}$$

Together with constraints (1.2), constraints (1.3) enforce the precedence relations.

$$y_{ij} = 1, \;\; y_{ji} = 0 \qquad ((i, j) \in TE) \tag{1.3}$$

Constraints (1.4) assure that the novel start-start sequencing variables are set according to the timing variables S_i, i.e., z_{ij} being equal to one if $S_i \leq S_j$. Therefore, we need a sufficiently small parameter $\epsilon > 0$ to set both variables z_{ij} and z_{ji} equal to one if activities i and j start at the same time. If all activity durations are integer values, we can set $\epsilon := 1$.

$$T \cdot z_{ij} \geq S_j - S_i + \epsilon \qquad (i, j \in V: \ i \neq j, (i, j) \notin TE, (j, i) \notin TE) \tag{1.4}$$

Constraints (1.5) model the storage resource constraints. At the start of each activity i, we check whether there is sufficient stock remaining for the resource consumption of this activity. The left-hand side of the constraint represents the consumption of activity i plus the total consumption for all activities that are still in progress at the start of activity i. More precisely, we consider the resource consumption of each other activity j that starts at any time before activity i or at the same time as activity i but is not completed before activity i begins. This condition is fulfilled for activity j if and only if the term $z_{ji} - y_{ji}$ is equal to one. The right-hand side of the equation corresponds to the initial stock of the storage resource plus the net production of all completed activities, i.e., $c^+_{jk} - c^-_{jk}$ for each activity j, where y_{ji} is equal to one. This outcome represents the amount of storage resources that are available for consumption for all activities that are still in progress when activity i starts.

$$c^-_{ik} + \sum_{\substack{j \in V, \ j \neq i \\ (i,j),(j,i) \notin TE}} c^-_{jk}(z_{ji} - y_{ji}) \leq C_k + \sum_{j \in V, \ j \neq i} y_{ji}(c^+_{jk} - c^-_{jk}) \quad (i \in V, \ k \in C : c^-_{ik} > 0) \tag{1.5}$$

For renewable resources, only the resource requirement of all activities j being executed in parallel to activity i must be considered; therefore, the analogous constraint reads

$$r_{ik} + \sum_{\substack{j \in V, \ j \neq i \\ (i,j),(j,i) \notin TE}} r_{jk}(z_{ji} - y_{ji}) \leq R_k \qquad (i \in V, \ k \in R : r_{ik} > 0) \tag{1.6}$$

The model for the RCPSP-CPR without extensions (NE) then reads as follows:

$$
(\text{SEQ\_NE}) \begin{cases}
\text{min.} & S_{n+1} \\
\text{s.t.} & (1.2)\text{–}(1.6) \\
& S_i \in \mathbb{R}_{\geq 0} \qquad (i \in V) \\
& y_{ij} \in \{0,1\} \qquad (i,j \in V: \ i \neq j) \\
& z_{ij} \in \{0,1\} \qquad (i,j \in V: \ i \neq j, (i,j) \notin TE, (j,i) \notin TE)
\end{cases}
$$

1.4.3 Extended MILP models

In this subsection, we add various valid inequalities that further enhance the performance of the model. We consider the set

$$F_2 := \{(i,j) \in V : i < j, (i,j), (j,i) \notin TE, \ \exists k \in R : r_{ik} + r_{jk} > R_k\}$$

of tuples of activities that cannot be executed simultaneously because the total requirement of the activities exceeds the capacity of any renewable resource. For each of these tuples, at least one of the corresponding completion-start sequencing variables must be equal to one, i.e.,

$$y_{ij} + y_{ji} \geq 1 \qquad ((i,j) \in F_2) \tag{1.7}$$

If a precedence relation is given between the activities of such a tuple, the corresponding sequencing variable is forced to be one, and therefore these tuples must not be considered. Analogously, for set

$$
\begin{aligned}
F_3 := \{&(i,j,h) \in V : i < j < h, (i,j), (i,h), (j,h) \notin F_2, \\
&\exists (i,j), (j,i), (i,h), (h,i), (j,h), (h,j) \notin TE, \ k \in R : r_{ik} + r_{jk} + r_{hk} > R_k\},
\end{aligned}
$$

the inequality

$$y_{ij} + y_{ji} + y_{ih} + y_{hi} + y_{jh} + y_{hj} \geq 1 \qquad ((i,j,h) \in F_3) \tag{1.8}$$

must be fulfilled.

Additionally, activity i cannot be completed before the start of activity j if activity j is completed before the start of activity i, i.e.,

$$y_{ij} + y_{ji} \leq 1 \qquad (i, j \in V:\ i < j, (i,j) \notin TE, (j,i) \notin TE) \tag{1.9}$$

It is known from the literature that inequalities (1.7), (1.8), and (1.9) improve the performance of continuous-time MILP models for the RCPSP (cf., e.g., Trautmann et al., 2018, Gnägi et al., 2018, Koné et al., 2011). Our novel model with these extensions from the literature (EL) reads as follows:

$$\text{(SEQ\_EL)} \begin{cases} \text{min.} & S_{n+1} & \\ \text{s.t.} & \text{(1.2)–(1.9)} & \\ & S_i \in \mathbb{R}_{\geq 0} & (i \in V) \\ & y_{ij} \in \{0,1\} & (i, j \in V:\ i \neq j) \\ & z_{ij} \in \{0,1\} & (i, j \in V:\ i \neq j, (i,j) \notin TE, (j,i) \notin TE) \end{cases}$$

In addition to these known inequalities, which are based on the completion-start sequencing variables only, we propose three novel valid inequalities that are based on the novel start-start sequencing variables. First, we note that for any pair of activities, at least one activity has to start before the other or both activities must start at the same time, i.e.,

$$z_{ij} + z_{ji} \geq 1 \qquad (i, j \in V:\ i < j, (i,j) \notin TE, (j,i) \notin TE) \tag{1.10}$$

Constraints (1.11) model the fact that if activity i is completed before the start of activity j, activity i also has to start before j.

$$z_{ij} \geq y_{ij} \qquad (i, j \in V:\ i \neq j, (i,j) \notin TE, (j,i) \notin TE) \tag{1.11}$$

If both variables z_{ij} and z_{ji} are equal to one for a pair of activities $i, j \in V$, then these two activities must start at the same time, i.e.,

$$2 - z_{ij} - z_{ji} \geq \frac{1}{T}(S_i - S_j) \qquad (i, j \in V : \ i \neq j, (i, j) \notin TE, (j, i) \notin TE) \tag{1.12}$$

Our fully extended model (EF) then reads as follows:

$$\text{(SEQ\_EF)} \begin{cases} \text{min.} & S_{n+1} \\ \text{s.t.} & \text{(1.2)–(1.12)} \\ & S_i \in \mathbb{R}_{\geq 0} \qquad (i \in V) \\ & y_{ij} \in \{0, 1\} \qquad (i, j \in V : \ i \neq j) \\ & z_{ij} \in \{0, 1\} \qquad (i, j \in V : \ i \neq j, (i, j) \notin TE, (j, i) \notin TE) \end{cases}$$

1.5 Computational results

In this section, we report on our computational analysis. In Subsection 1.5.1, we describe our experimental design. In Subsection 1.5.2, we analyze various configurations of our novel model to study the impact of the model extensions presented in Subsection 1.4.3 and to identify the best configuration. In Subsection 1.5.3, we compare the performance of our novel model to state-of-the-art models from the literature (cf. Table 1.5); thereby, we analyze three scenarios for the storage resources. In the first scenario, we study the instances proposed by Koné et al. (2013) that comprise storage resources with a relatively high initial stock. In the second scenario, we investigate the same instances but with a reduced initial stock of storage resources. Due to this reduced initial stock, the storage resources are scarcer, which makes it more challenging to devise feasible solutions for these instances. In the third scenario, we consider renewable resources only. In Subsection 1.5.4, we consider the same three scenarios for modified instances of set CV that are considered hard to solve regarding renewable resources (cf. Coelho & Vanhoucke, 2020). In Subsection 1.5.5, we analyze the scalability of the models using modified instances of set J60 that comprise a relatively large number of activities (cf. Kolisch & Sprecher, 1996).

1.5.1 Experimental design

In this subsection, we describe our experimental design. In Subsection 1.5.1.1, we outline the test instances. In Subsection 1.5.1.2, we describe the performance measures used. In Subsection 1.5.1.3, we report on the details of the implementation.

Abbreviation	DT	CT	Variable type	Proposed by	Based on
DT	x		Pulse	Koné et al. (2013)	Pritsker et al. (1969)
DDT	x		Pulse	Koné et al. (2013)	Christofides et al. (1987)
FCT		x	Flow	Koné et al. (2013)	Artigues et al. (2003)
OOE		x	Event	Koné et al. (2013)	Koné et al. (2011)
SEQ		x	Sequence	This paper	This paper

Table 1.5: Overview of model abbreviations

1.5.1.1 Test instances

We used two instance sets to analyze the influence of the storage resources on the performance of the models. The first set is called J30-CPR and contains 456 instances for which a feasible solution exists. Koné et al. (2013) generated this set by modifying benchmark instances for the RCPSP from set J30 of the PSPLIB (Kolisch & Sprecher, 1996). Set J30 contains 480 problem instances comprising 30 real activities and four renewable resources. To extend the set to the RCPSP-CPR, Koné et al. (2013) added three storage resources to the instances by assigning randomly generated integer values between zero and ten to each activity for each storage resource. They set the initial stock of each storage resource to a value based on a so-called resource strength indicator $\mathrm{RS}^{\mathrm{CPR}} \in [0, 1]$, with a lower value implying that the storage resources are scarcer. The authors sampled this indicator randomly from the interval $[0.7, 1]$, implying that the initial stock of the storage resource constraints is relatively high. Therefore, the storage resources are not very scarce in this set of instances. To analyze instances that are more challenging regarding the storage resources, we modify the J30-CPR instances by lowering the initial stock of the storage resources. We achieve this by randomly sampling the resource strength indicator from the interval $[0.4, 0.7]$. The resulting set contains 290 instances for which a feasible solution exists; we call it J30-CPR-L.

The second set is called CV-CPR and contains 623 instances for which a feasible solution exists. We generated this set by modifying benchmark instances for the RCPSP from set CV, which was proposed by Coelho & Vanhoucke (2020). Set CV contains 623 problem instances comprising 20 to 30 real activities and one to four renewable resources. The problem instances of this set are considered very hard to solve. To extend the set to the RCPSP-CPR, we added three storage resources to the instances following the methodology proposed by Koné et al. (2013). We randomly sampled the resource strength indicator from the empirical values of set J30-CPR, implying a relatively high initial stock of the storage resource constraints. Again, we modify these instances by reducing the storage resources' initial stock by randomly sampling a new resource strength indicator

Name	RR	CPR	CPR-L	Results
J30-CPR	x	x		Subsection 1.5.3.1
J30-CPR-L	x		x	Subsection 1.5.3.2
J30	x			Subsection 1.5.3.3
CV-CPR	x	x		Subsection 1.5.4.1
CV-CPR-L	x		x	Subsection 1.5.4.2
CV	x			Subsection 1.5.4.3
J60-CPR	x	x		Subsection 1.5.5

Table 1.6: Overview of the sets of instances

from the interval $[0.4, 0.7]$. The resulting set contains 617 instances for which a feasible solution exists; we call it CV-CPR-L.

To analyze the scalability of all models, we use a set called J60-CPR that contains 476 instances for which a feasible solution exists. We generated this set by modifying benchmark instances for the RCPSP from set J60 of the PSPLIB (Kolisch & Sprecher, 1996). Set J60 contains 480 problem instances comprising 60 real activities and four renewable resources. Again, we added three storage resources to the instances following the methodology proposed by Koné et al. (2013). We generated the initial stock using the same resource strength indicators as the corresponding J30-CPR instances, e.g., instance J60_1_1 received the same resource strength indicator as instance J30_1_1.

We summarize the characteristics of all sets of instances in Table 1.6, where we indicate whether the instances comprise renewable resources (RR), storage resources with a relatively high initial stock (CPR), or a relatively low initial stock (CPR-L). Additionally, we indicate in which subsection the results for the respective set of instances are discussed. Note that the results for instances comprising renewable resources only, i.e., instances of the RCPSP, are discussed in Subsection 1.5.3.3 and Subsection 1.5.4.3.

1.5.1.2 Performance measures

For all sets of instances, we report the following key metrics:

- # Feas: Number of instances for which a feasible solution was found
- # Opt: Number of instances for which a proven optimal solution was found
- # Best: Number of instances for which the best feasible solution among all models was found

- Gap^{MIP}: Average relative deviation between the objective function value of the best feasible solution (UB^*) and the best lower bound (LB^*), calculated as $(\text{UB}^* - \text{LB}^*)/\text{UB}^*$
- $\text{Gap}_A^{\text{UB}^* - \text{CPM}}$: Average relative deviation between UB^* and the lower bound based on the critical path method (CPM), calculated as $(\text{UB}^* - \text{CPM})/\text{CPM}$; this average considers the subset of instances for which all models have found a feasible solution.
- $\text{Gap}_M^{\text{UB}^* - \text{CPM}}$: Average relative deviation between UB^* and CPM; this average considers all instances for which the respective model has found a feasible solution.
- $\text{Gap}^{\text{LB}^* - \text{CPM}}$: Average relative deviation between LB^* and CPM, calculated as $(\text{LB}^* - \text{CPM})/\text{CPM}$
- Time: Average time (in seconds) used to build the model and solve an instance

In all following tables, we mark the best values per metric in bold. As the metric $\text{Gap}_M^{\text{UB}^* - \text{CPM}}$ uses different subsets of instances for all models, it is not suitable for a comparison between models, but rather serves as a benchmark for the overall solution quality. Therefore, we do not mark any values in this column in bold.

1.5.1.3 Implementation

We implemented all models in Python 3.8 and used Gurobi version 9.1.2 as the MILP solver. We carried out the computations on a workstation with two 8-core Intel(R) Xeon(R) E5-2687 W CPUs (3.1 GHz) and 128 GB RAM. We set a time limit of 500 seconds per instance and restricted the number of available threads to two. Since the duration of all activities in the considered sets of instances takes integer values, we set $\epsilon := 1$ in constraints (1.4) for our novel model. Additionally, this implies that there is at least one optimal solution where the start times of all activities are integers. This allows us to restrict the start time variables of all CT models to take integer values. We demonstrate the benefit of this restriction in Subsection 1.5.2. Apart from this modification, we implemented all reference models as described in Koné et al. (2013), using $T = \sum_{i \in V} p_i$ as the scheduling horizon.

1.5.2 Model configuration

In this subsection, we analyze the computational performance of various configurations of the novel CT model. Specifically, we analyze the impact of various sets of valid constraints and of restricting the timing variables to take integer values. We list our configurations in Table 1.7.

Abbreviation	Valid constraints	Integer start times
SEQ_NE	None	x
SEQ_EL	(1.7) – (1.9)	x
SEQ_EF	(1.7) – (1.12)	x
SEQ	(1.7) – (1.12)	✓

Table 1.7: Overview of the model configurations

We use the instances of set J30-CPR for this comparison. The results are summarized in Table 1.8.

Model	# Feas	# Opt	# Best	Gap^{MIP}	$\text{Gap}_A^{\text{UB}^*-\text{CPM}}$	$\text{Gap}_M^{\text{UB}^*-\text{CPM}}$	$\text{Gap}^{\text{LB}^*-\text{CPM}}$	Time
SEQ_NE	454	386	403	3.23%	17.90%	17.90%	12.47%	97.20
SEQ_EL	**456**	414	428	1.23%	17.27%	17.67%	15.66%	73.61
SEQ_EF	**456**	438	451	0.64%	17.04%	17.43%	16.31%	40.51
SEQ	**456**	**445**	**455**	**0.38%**	**17.01%**	17.39%	**16.73%**	**22.74**

Table 1.8: Comparison of different model configurations using the instances of set J30-CPR

We observe a positive impact of both sets of valid constraints on the model performance regarding all metrics. In particular, the lower bounds are improved, which is indicated by the larger gap to the critical path-based lower bound. Additionally, restricting the start time variables to take integer values has a positive influence, leading to configuration SEQ performing best regarding all metrics. Therefore, we use the SEQ configuration of our novel model for all of the following experiments. For the CT reference models FCT and OOE, we also restrict the start time variables to take integer values in the following.

1.5.3 Results for set J30-CPR and variants

In this subsection, we present the computational results for the instances of sets J30-CPR (cf. Subsection 1.5.3.1), J30-CPR-L (cf. Subsection 1.5.3.2), and J30 (cf. Subsection 1.5.3.3).

1.5.3.1 Relatively high initial stock

In this subsection, we report our computational results for the instances of set J30-CPR; the results are presented in Table 1.9.

Model	# Feas	# Opt	# Best	Gap^{MIP}	$\text{Gap}_A^{\text{UB}^*-\text{CPM}}$	$\text{Gap}_M^{\text{UB}^*-\text{CPM}}$	$\text{Gap}^{\text{LB}^*-\text{CPM}}$	Time
DT	434	379	392	1.79%	14.53%	17.12%	14.71%	122.12
DDT	432	381	396	1.15%	14.06%	15.22%	15.29%	133.36
FCT	**456**	411	436	1.22%	13.57%	17.54%	15.60%	84.84
OOE	453	0	66	86.38%	23.88%	28.28%	-83.10%	501.87
SEQ	**456**	**445**	**456**	**0.38%**	**13.50%**	17.39%	**16.73%**	**22.74**

Table 1.9: Computational results for the instances of set J30-CPR

We observe that our novel model (SEQ) performs best regarding all comparable metrics. It yields the best feasible solution among all models to all instances and proves optimality for over 97% of them. Additionally, it yields the best upper and lower bounds, on average. Regarding the models from the literature, model FCT performs best, providing a feasible solution to each instance, solving the second-most instances to optimality, and providing a best solution among all models for the second-most instances. This result stands in contrast to the results of Koné et al. (2013), where both DT models provide proven optimal solutions for more instances than the FCT model. This improved performance can be attributed to restricting the start time variables to integer values. The DT models provide the third-highest number of instances solved to optimality. However, they cannot find a feasible solution for 22 and 24 instances, respectively. The OOE model finds feasible solutions to the third-most instances but cannot prove optimality for any instance. A reason for this might be the weaker LP relaxation that the model yields, whose optimal objective function values are smaller than the CPM lower bound for most instances; this is also the reason for the negative value in the $\text{Gap}^{\text{LB}^*\text{-CPM}}$ column.

1.5.3.2 Relatively low initial stock

In this subsection, we report our computational results for the instances of set J30-CPR-L and discuss the influence of the relatively low initial stock on the performance of the models. The results of our analysis for set J30-CPR-L are summarized in Table 1.10.

Model	# Feas	# Opt	# Best	Gap^{MIP}	$\text{Gap}_A^{\text{UB}^*-\text{CPM}}$	$\text{Gap}_M^{\text{UB}^*-\text{CPM}}$	$\text{Gap}^{\text{LB}^*-\text{CPM}}$	Time
DT	205	139	155	4.18%	20.23%	27.58%	22.39%	315.44
DDT	203	138	156	3.48%	21.69%	25.06%	24.19%	338.45
FCT	269	167	208	5.68%	20.09%	31.09%	22.44%	280.41
OOE	261	0	2	88.76%	42.07%	53.36%	-83.25%	502.22
SEQ	**290**	**259**	**287**	**1.66%**	**19.20%**	31.99%	**29.36%**	**94.22**

Table 1.10: Computational results for the instances of set J30-CPR-L

Again, the novel CT model performs best regarding all comparable metrics; it finds the best solution among all models for most instances and proves optimality for over 89% of the instances. Additionally, it yields the best upper and lower bounds, on average. Regarding the models from the literature, again, the FCT model performs best. However, none of the models from the literature finds feasible solutions to all instances. When comparing the results to the previous setting with a relatively high initial stock, the model SEQ devises feasible solutions to all instances in both settings. In contrast, all models from the literature provide feasible and proven optimal solutions to relatively fewer instances than in the previous setting. In summary, it can be stated that the performance advantage of the novel CT model over the reference models from the literature increases with the scarcity of storage resources.

1.5.3.3 No storage resources

In this subsection, we report our computational results for the instances of set J30 and discuss the influence of omitting the storage resources on the performance of the models for these instances. The results of our computational study for set J30 are presented in Table 1.11.

Model	# Feas	# Opt	# Best	Gap^{MIP}	$\text{Gap}_A^{\text{UB}^*-\text{CPM}}$	$\text{Gap}_M^{\text{UB}^*-\text{CPM}}$	$\text{Gap}^{\text{LB}^*-\text{CPM}}$	Time
DT	**480**	443	462	0.77%	13.44%	13.59%	12.21%	54.16
DDT	479	437	451	1.00%	13.66%	13.66%	11.96%	67.89
FCT	**480**	458	473	0.69%	13.30%	13.45%	12.23%	40.21
OOE	**480**	0	302	81.46%	15.35%	15.50%	-79.38%	501.08
SEQ	**480**	**471**	**478**	**0.35%**	**13.28%**	13.43%	**12.80%**	**15.28**

Table 1.11: Computational results for the instances of set J30

We observe that our novel model performs best regarding all metrics, providing proven optimal solutions to over 98% of instances. Regarding the models from the literature, all models provide feasible solutions to all instances, except for DDT, which fails to find a feasible solution to one instance. This stands in contrast to both of the previous settings that include storage resources. Additionally, the average quality of their upper and lower bounds is considerably better for the models from the literature. This highlights that these models are not best suited to tackle the additional challenges the storage resource constraints pose. A potential explanation for this is that the SEQ model does not require additional variables to tackle the storage resource constraints. Nevertheless, these results indicate that the SEQ model also outperforms the reference models for instances without storage resources, i.e., instances of the RCPSP.

1.5.4 Results for set CV-CPR and variants

In this subsection, we report on the computational results for the instances of sets CV-CPR (cf. Subsection 1.5.4.1), CV-CPR-L (cf. Subsection 1.5.4.2), and CV (cf. Subsection 1.5.4.3).

1.5.4.1 Relatively high initial stock

In this subsection, we report our computational results for the instances of set CV-CPR. The results are presented in Table 1.12.

Model	# Feas	# Opt	# Best	Gap^{MIP}	$\text{Gap}_A^{\text{UB}^*-\text{CPM}}$	$\text{Gap}_M^{\text{UB}^*-\text{CPM}}$	$\text{Gap}^{\text{LB}^*-\text{CPM}}$	Time
DT	612	29	284	**13.35%**	149.09%	153.27%	**118.91%**	483.55
DDT	564	22	187	14.94%	155.49%	156.53%	116.14%	492.96
FCT	**623**	16	243	36.55%	149.59%	153.10%	56.54%	497.92
OOE	**623**	0	21	90.90%	163.28%	167.09%	-76.36%	501.54
SEQ	**623**	**62**	**447**	29.54%	**145.92%**	149.23%	70.48%	**472.69**

Table 1.12: Computational results for the instances of set CV-CPR

We observe that our novel model (SEQ) performs best regarding the number of instances solved to optimality and the number of instances for which a best solution among all models has been obtained. Additionally, it yields the best upper bounds, on average. Regarding the models from the literature, FCT and OOE find feasible solutions to all instances, while the models DT and DDT do not find feasible solutions to 11 and 59 instances, respectively. However, the DT and DDT models yield better lower bounds than all CT models. Therefore, they have smaller MIP gaps, on average.

1.5.4.2 Relatively low initial stock

In this subsection, we report on our computational results for the instances of set CV-CPR-L and discuss the influence of the relatively low initial stock on the performance of the models. The results of our computational study for set CV-CPR-L are presented in Table 1.13.

Again, we observe that our novel model performs best regarding the number of instances solved to optimality and the number of instances for which a best feasible solution among all models has been obtained. Additionally, it yields the best upper bounds, on average. Regarding the models from the literature, FCT and OOE find feasible solutions to all instances, while DT and DDT do not find feasible solutions to 12 and 68 instances, respectively. Again, the DT and DDT models yield better lower bounds and therefore have smaller MIP gaps, on average. When comparing the results to the previous setting

Model	# Feas	# Opt	# Best	Gap^{MIP}	$\text{Gap}_A^{\text{UB}^*-\text{CPM}}$	$\text{Gap}_M^{\text{UB}^*-\text{CPM}}$	$\text{Gap}^{\text{LB}^*-\text{CPM}}$	Time
DT	605	25	298	**13.07%**	149.29%	151.92%	**118.90%**	485.55
DDT	549	24	172	15.34%	157.91%	158.31%	115.72%	493.09
FCT	**617**	25	280	35.49%	149.96%	152.26%	58.94%	494.65
OOE	**617**	0	21	92.32%	165.17%	168.11%	-79.68%	501.56
SEQ	**617**	**52**	**404**	30.42%	**147.77%**	149.97%	69.89%	**474.09**

Table 1.13: Computational results for the instances of set CV-CPR-L

with a relatively high initial stock, we observe that the CT models find feasible solutions to all instances in both settings. Surprisingly, the DDT and FCT models solve more instances to optimality than in the previous setting, which stands in contrast to their performance regarding the instances of set J30-CPR-L. However, the novel model still solves more instances to optimality and finds the best feasible solution among all models for more instances.

1.5.4.3 No storage resources

In this subsection, we report our computational results for the instances of set CV and discuss the influence of omitting the storage resources on the models' performance for these instances. The results of our computational study for set CV are presented in Table 1.14.

Model	# Feas	# Opt	# Best	Gap^{MIP}	$\text{Gap}_A^{\text{UB}^*-\text{CPM}}$	$\text{Gap}_M^{\text{UB}^*-\text{CPM}}$	$\text{Gap}^{\text{LB}^*-\text{CPM}}$	Time
DT	**623**	35	**398**	**11.28%**	**147.74%**	147.63%	**119.83%**	477.84
DDT	**623**	5	249	14.00%	152.33%	152.20%	116.13%	499.23
FCT	622	27	249	34.16%	151.10%	151.10%	61.36%	493.82
OOE	**623**	0	47	85.50%	159.95%	159.83%	-62.92%	500.84
SEQ	**623**	**77**	357	29.04%	149.05%	148.94%	72.72%	**465.87**

Table 1.14: Computational results for the instances of set CV

We observe that our novel model performs best regarding the number of instances solved to optimality, providing optimal solutions to more than twice as many instances as the second-best model regarding this metric (DT). Regarding the models from the literature, all models find feasible solutions to all instances, except for FCT, which fails to devise a feasible solution to one instance. The DT model finds the best feasible solution among all models for most instances and, on average, yields the best upper and lower bounds. This stands in contrast to both of the previous settings that include storage resources, where the models DT and DDT models did not devise feasible solutions to all instances.

In sum, the SEQ model performs best regarding the number of instances solved to feasibility and optimality. In the settings without storage resources, the DT model yields better upper bounds, on average, and more instances for which a best solution among all models has been obtained. This result is notable, since the DT model was originally proposed by Pritsker et al. (1969) over 50 years ago and still performs comparably well today, especially when only renewable resources are involved.

1.5.5 Scalability

In this subsection, we analyze the scalability of all models regarding instances that comprise a relatively large number of activities. The results of our computational study for set J60-CPR are presented in Table 1.15.

Model	# Feas	# Opt	# Best	Gap^{MIP}	$\text{Gap}_A^{\text{UB}^*-\text{CPM}}$	$\text{Gap}_M^{\text{UB}^*-\text{CPM}}$	$\text{Gap}^{\text{LB}^*-\text{CPM}}$	Time
DT	385	353	370	1.22%	**1.16%**	4.51%	**5.20%**	152.21
DDT	355	332	333	**0.72%**	1.34%	2.23%	3.65%	238.88
FCT	187	128	132	4.79%	5.32%	8.88%	1.91%	437.10
OOE	**473**	0	27	97.38%	62.54%	81.19%	-95.42%	510.62
SEQ	450	**372**	**438**	3.45%	**1.16%**	9.32%	4.09%	**124.80**

Table 1.15: Computational results for the instances of set J60-CPR

We observe that our novel model (SEQ) provides optimal solutions for most instances, and for over 90% of the instances, it provides the best solution among all models. Regarding the models from the literature, the OOE model solves the most instances to feasibility, missing only three instances. However, the solution quality is considerably worse than that of all other models. The DT and DDT models do not find feasible solutions to 91 and 121 instances, respectively. This also explains their relatively small MIP gaps that are calculated by using only instances for which the respective model found a feasible solution. Notably, the FCT model, which performed second-best in the J30-CPR instances, only provides feasible solutions to less than 40% of the instances. One potential explanation for this is that the model comprises $O(n^3)$ constraints and, therefore, does not scale well to instances comprising a relatively large number of activities.

In sum, the novel model provides many feasible solutions that have a comparably low makespan, while the models from the literature struggle with finding feasible solutions, indicating that our novel model scales best to larger instances.

1.6 Conclusion and outlook

In this paper, we introduced a novel continuous-time MILP model for the RCPSP-CPR based on a novel type of sequencing variable. By combining completion-start sequencing variables with this novel type of start-start sequencing variable, we can efficiently formulate renewable resource and storage resource constraints. The main advantages of our model are that it does not require additional variables to model the storage resource constraints, it is compact, and the variables are easily interpretable. In addition, the novel variable type enables us to add novel valid inequalities to the model that considerably enhance its computational performance. Our computational results indicate that our model outperforms all reference MILP models from the literature and is best suited to address the additional challenges the storage resource constraints pose.

For future research, we propose to develop a matheuristic solution procedure, i.e., a mathematical programming-based heuristic, that utilizes the new MILP model. A promising direction may be to initially relax the binary restriction on the sequencing variables, and afterwards iteratively restore it (cf., e.g., Bigler et al., 2024). Furthermore, how the new modeling approach can be adapted to related project scheduling problems, i.e., variants of the RCPSP that have been proposed in the literature (cf., e.g., Hartmann & Briskorn, 2022 for an overview), should be investigated.

Bibliography

Agha, M. H., Thery, R., Hetreux, G., Hait, A., & Le Lann, J. M. (2010). Integrated production and utility system approach for optimizing industrial unit operations. *Energy*, *35*, 611–627. doi:10.1016/j.energy.2009.10.032.

Artigues, C., Koné, O., Lopez, P., & Mongeau, M. (2015). Mixed-integer linear programming formulations. In C. Schwindt, & J. Zimmermann (Eds.), *Handbook on Project Management and Scheduling Vol.1* (pp. 17–41). Cham: Springer International Publishing. doi:10.1007/978-3-319-05443-8_2.

Artigues, C., Michelon, P., & Reusser, S. (2003). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, *149*, 249–267. doi:10.1016/S0377-2217(02)00758-0.

Bigler, T., Gnägi, M., & Trautmann, N. (2024). MIP-based solution approaches for multi-site resource-constrained project scheduling. *Annals of Operations Research*, *337*, 627–647. doi:10.1007/s10479-022-05109-0.

Blazewicz, J., Lenstra, J., & Kan, A. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, *5*, 11–24. doi:10.1016/0166-218X(83)90012-4.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, *112*, 3–41. doi:10.1016/S0377-2217(98)00204-5.

Carlier, J., & Moukrim, A. (2015). Storage resources. In C. Schwindt, & J. Zimmermann (Eds.), *Handbook on Project Management and Scheduling Vol.1* (pp. 177–189). Cham: Springer International Publishing. doi:10.1007/978-3-319-05443-8_9.

Carlier, J., Moukrim, A., & Xu, H. (2009). The project scheduling problem with production and consumption of resources: A list-scheduling based algorithm. *Discrete Applied*

Mathematics, *157*, 3631–3642. doi:10.1016/j.dam.2009.02.012.

Christofides, N., Alvarez-Valdes, R., & Tamarit, J. (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, *29*, 262–273. doi:10.1016/0377-2217(87)90240-2.

Coelho, J., & Vanhoucke, M. (2020). Going to the core of hard resource-constrained project scheduling instances. *Computers & Operations Research*, *121*, 104976. doi:10.1016/j.cor.2020.104976.

Gnägi, M., Zimmermann, A., & Trautmann, N. (2018). A continuous-time unit-based MILP formulation for the resource-constrained project scheduling problem. In T. Laosirihongthong, K. H. Chai, R. Jiao, & M. Xie (Eds.), *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (pp. 432–436). doi:10.1109/IEEM.2018.8607337.

Hartmann, S., & Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *297*, 1–14. doi:10.1016/j.ejor.2021.05.004.

Koch, T., Berthold, T., Pedersen, J., & Vanaret, C. (2022). Progress in mathematical programming solvers from 2001 to 2020. *EURO Journal on Computational Optimization*, *10*, 100031. doi:10.1016/j.ejco.2022.100031.

Kolisch, R., & Sprecher, A. (1996). PSPLIB — A project scheduling problem library. *European Journal of Operational Research*, *96*, 205–216. doi:10.1016/S0377-2217(96)00170-1.

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*, 25–47. doi:10.1007/s10696-012-9152-5.

Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research*, *38*, 3–13. doi:10.1016/j.cor.2009.12.011.

Laborie, P. (2003). Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, *143*, 151–188. doi:10.1016/S0004-3702(02)00362-4.

Neumann, K., & Schwindt, C. (2003). Project scheduling with inventory constraints.

Mathematical Methods of Operations Research, *56*, 513–533. doi:10.1007/s001860200251.

Pritsker, A. A. B., Waiters, L. J., & Wolfe, P. M. (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, *16*, 93–108. doi:10.1287/mnsc.16.1.93.

Tavares, L. (2002). A review of the contribution of operational research to project management. *European Journal of Operational Research*, *136*, 1–18. doi:10.1016/S0377-2217(01)00097-2.

Trautmann, N., Rihm, T., Saner, N., & Zimmermann, A. (2018). A continuous-time assignment-based MILP formulation for the resource-constrained project scheduling problem. In M. Caramia, L. Bianco, & S. Giordani (Eds.), *Proceedings of the 16th international conference on project management and scheduling* (pp. 242–245). URL: http://www.pms2018.ing.uniroma2.it/.

Paper II

Integer programming for multi-mode resource-constrained project scheduling[2]

Nicklas Klein

Department of Business Administration
University of Bern

Contents

2.1 Introduction **37**
2.2 Problem setting **40**
2.3 Discrete-time models **43**
2.3.1 Models using pulse variables 44
2.3.2 Models using step variables 46
2.3.3 Models using on/off variables 48
2.3.4 Models using on/off and pulse variables 50
2.4 Continuous-time models **51**
2.4.1 Sequence-based model 51
2.4.2 Continuous-time models from the literature 54
2.5 Comparison of models **55**
2.5.1 Experimental design 55
2.5.2 Comparison of the LP relaxations 56
2.5.3 Computational performance analysis 60
2.6 Conclusions **66**
Bibliography **69**

[2] As of December 2024, this version of the paper was under review, submitted to Annals of Operations Research. The paper has since been published with minor revisions in Annals of Operations Research: `https://doi.org/10.1007/s10479-025-06572-1`.

Abstract

Project scheduling is a vital management task in many organizations across various industries. A project typically consists of activities that require time and scarce resources for execution. In many projects, there are trade-offs between the resource requirements and the duration of activities. These trade-offs can be represented via multiple execution modes of activities, which are considered in the multi-mode resource-constrained project scheduling problem (MRCPSP). The MRCPSP comprises determining the activities' start times and execution modes to minimize the project completion time while respecting precedence relations and availabilities of renewable and nonrenewable resources. In this paper, we propose several discrete-time (DT) and one continuous-time (CT) mixed-integer linear programming (MILP) model for the MRCPSP based on models for the single-mode resource-constrained project scheduling problem (RCPSP). We first analyze the models' LP relaxations and show an equivalence between DT models. Then, we compare the computational performance of the models on benchmark instances. Our results indicate that the proposed CT model outperforms the state-of-the-art models on large problem instances and problem instances with relatively long activity durations, while it is competitive on smaller problem instances. A possible explanation for this outperformance is that the LP relaxations of the proposed CT model can be solved considerably faster than the LP relaxations of the state-of-the-art models.

2.1 Introduction

Managing projects efficiently has become increasingly important for companies across various industries. Typically, projects are composed of activities that require time and scarce resources for execution. In many projects, the durations and resource requirements of activities may not be fixed but involve trade-offs that can be represented via alternative execution modes of the activities. An essential component of managing projects with limited resources is the challenging task of resource-constrained project scheduling, which comprises allocating resources to activities over time (cf., e.g., Brucker et al., 1999). One technique that has become increasingly important for addressing this challenging scheduling task is mixed-integer linear programming (MILP) (cf., e.g., Artigues et al., 2015, Trautmann et al., 2018). One reason for this trend is the greater flexibility in terms of different objective functions, additional constraints, and other needs of business applications that MILP models offer compared to specialized algorithms. Furthermore, advances in computer hardware and algorithms have significantly enhanced the performance of mathematical programming solvers in recent years; this is demonstrated, e.g., by Koch et al. (2022), who report a total speedup factor of up to 1,000 for MILP problems in the past 20 years. Despite this speedup, however, the MILP model selected strongly affects the computational performance of MILP-based solution methods, highlighting the need for the development and evaluation of MILP modeling approaches (cf., e.g., Koné et al., 2011).

We consider the multi-mode resource-constrained project scheduling problem (MRCPSP). In the MRCPSP, we are given a set of project activities that are connected via completion-start precedence relations. For execution, the activities require time and two kinds of resources: renewable resources, which are utilized by an activity while it is in progress and released upon completion (e.g., personnel), and nonrenewable resources, which are consumed at an activity's start but not replenished (e.g., raw materials). Additionally, each activity has a finite set of execution modes, influencing its duration and resource requirements. The problem comprises determining the start times and execution modes of the activities to minimize the project completion time (makespan) while respecting the renewable and nonrenewable resource availabilities and the precedence relations (cf., e.g., Mika et al., 2015). The MRCPSP is an extension of the well-known single-mode resource-constrained project scheduling problem (RCPSP) and is, thus, an NP-hard combinatorial optimization problem (cf. Blazewicz et al., 1983). Additionally, even the problem of devising a feasible solution is NP-hard for the MRCPSP if two or more nonrenewable resources are considered (cf. Kolisch & Drexl, 1997). Practical applications of the MRCPSP can be found, e.g., in R&D projects in the automotive industry (cf. Bartels & Zimmermann, 2009) or in construction projects (cf. Xu & Zeng, 2015).

Multiple MILP models for project scheduling problems have been proposed in the literature (cf., e.g., Artigues et al., 2015 for an overview). These models can be categorized into discrete-time (DT) and continuous-time (CT) models. In DT models, the scheduling horizon is divided into intervals of equal length, and each activity can only start at the beginning of those intervals. This leads to a relatively large number of variables and constraints if the activity durations are large or if their accuracy requires a fine discretization of the scheduling horizon. Such scenarios arise, e.g., in semiconductor manufacturing (cf. Pfund et al., 2006), or in plastic compounding plants (cf. Pinto & Grossmann, 1995). In CT models, on the other hand, activities are allowed to start at any time during the scheduling horizon. To the best of our knowledge, the current state-of-the-art DT and CT models have been proposed by Talbot (1982) and Sayah (2023), respectively. The DT model includes binary "pulse" variables that indicate whether an activity starts at a specific time in a specific mode; it is an extension of the well-known RCPSP model proposed by Pritsker et al. (1969). The CT model represents units of the renewable resources as flows between pairs of activities; it is an extension of the RCPSP model proposed by Artigues et al. (2003). Several other MILP modeling approaches for the RCPSP have been proposed in the literature (cf., e.g., Klein, 2000, Kopanos et al., 2014, Klein et al., 2024). However, these approaches have not been used to address the MRCPSP, to the best of our knowledge.

In this paper, we propose several DT and CT MILP models for the MRCPSP that are based on models for the single-mode RCPSP. First, we extend three DT models that are based on DT variable types, which are known to yield superior computational performance for the RCPSP (cf., e.g., Kopanos et al., 2014). Additionally, we consider a disaggregation of the precedence constraints in each model, which is known to yield stronger LP relaxations for the RCPSP (cf., e.g., Christofides et al., 1987, Artigues, 2017). Second, we extend a CT model that is based on a combination of two kinds of sequencing variables, which are used to identify activities that are executed in parallel (cf. Klein et al., 2024). This sequence-based model is compact, i.e., it contains a polynomial number of variables and constraints. To evaluate the extended models, we compare them to state-of-the-art models on six sets of established benchmark instances. These sets contain well-known instances from the PSPLIB (cf. Kolisch & Sprecher, 1996), modifications of these instances that comprise relatively large activity durations (cf. Gnägi et al., 2019), and relatively large instances from the MMLIB (cf. Van Peteghem & Vanhoucke, 2014). We first analyze the (root-node) LP relaxations of all models. We show, using variable transformations, that the LP relaxations of the extended DT models are equivalent to those of the models based on pulse variables. These results and transformations are consistent with corresponding equivalences for the single-mode RCPSP reported by Artigues (2017). Regarding the CT models, we observe that their LP relaxations are weaker than those of the DT models, i.e., they have smaller optimal objective function values, on average. However, the LP relaxations of the sequence-based CT model can be solved considerably faster than those of the fastest DT model. Therefore, we observe a trade-off between the quality of the LP relaxations and the time required to solve them. This trade-off is a distinct property of the proposed models and cannot be observed with the current state-of-the-art DT and CT models. Then, we evaluate the computational performance of all models. Our results indicate that the sequence-based CT model outperforms the state-of-the-art for large problem instances and for problem instances with relatively long activity durations, while it is competitive for smaller problem instances. In particular, it outperforms the current state-of-the-art CT model for all six sets of instances for all considered metrics. Regarding the DT models, we observe that a DT model based on pulse variables (cf. Talbot, 1982) remains state-of-the-art among DT models. This finding differs from the computational performance of the corresponding models for the single-mode RCPSP (cf., e.g., Koné et al., 2011, Kopanos et al., 2014). Compared to this DT model, the proposed CT model yielded weaker lower bounds, possibly due to its weaker LP relaxations. However, the proposed CT model devised considerably better upper bounds, on average. In addition, the DT model did not devise feasible solutions for a considerable number of large problem instances and for problem instances with relatively long activity durations. Two possible explanations for the superior performance of the sequence-based CT model are the smaller total number of variables and constraints that this model contains, and the considerably smaller CPU time required to solve its LP relaxations. Finally, our results indicate that the MILP solver is able to devise good-quality feasible solutions for a large number of problem instances using the sequence-based CT model. This opens up promising directions for future research, e.g., the development of matheuristics that combine mathematical programming and (meta-) heuristic techniques to devise good-quality feasible solutions within reasonable running time (cf., e.g., Fernandes & de Souza, 2021, Bigler et al., 2024).

The remainder of this paper is structured as follows. In Section 2.2, we describe the MRCPSP in detail and present an illustrative example of the planning problem. In Section 2.3, we present the different types of DT variables using our illustrative example and introduce the extended DT models. In Section 2.4, we present the extended sequence-based CT model and relate the formulation to CT models from the literature. In Section 2.5, we report on the design and results of our comparison between the extended models and the state-of-the-art. In Section 2.6, we conclude the paper and provide outlooks for future research.

2.2 Problem setting

In this section, we describe the planning problem and provide an illustrative example of the MRCPSP. In what follows, we use the notation for sets and parameters summarized in Table 2.1.

In the MRCPSP, we are given a set $\bar{V}$ of n real activities to be executed throughout the project. To model the project's start and completion, we additionally consider two

$\bar{V}$	Set of real activities ($\bar{V} \coloneqq \{1, \ldots, n\}$)
V	Set of all activities ($V \coloneqq \{0, 1, \ldots, n, n+1\}$)
T	Scheduling horizon
H	Discretization of the scheduling horizon ($H \coloneqq \{0, 1, \ldots, T\}$)
M_i	Set of modes for activity $i \in \bar{V}$
E	Set of completion-start precedence relations $(i, j) \in V \times V$
TE	Transitive closure of E
V_p	Pairs of activities that are not precedence-related
R	Set of renewable resources
N	Set of nonrenewable resources
R_k	Availability of renewable resource $k \in R$
W_k	Capacity of nonrenewable resource $k \in N$
p_i^m	Duration of activity $i \in V$ in mode $m \in M_i$
r_{ik}^m	Requirement of activity $i \in V$ of renewable resource $k \in R$ in mode $m \in M_i$
w_{ik}^m	Consumption of activity $i \in V$ of nonrenewable resource $k \in N$ in mode $m \in M_i$
ES_i	Earliest start time of activity $i \in V$
LS_i	Latest start time of activity $i \in V$

Table 2.1: Notation for sets and parameters

fictitious activities with indices 0 and $n+1$, and we denote the set of all activities as $V \coloneqq \{0, 1, \ldots, n, n+1\}$. Each activity $i \in \bar{V}$ must be executed in precisely one of the execution modes $m \in M_i$, with M_i representing the set of possible execution modes for this activity. If activity $i \in \bar{V}$ is executed in mode $m \in M_i$, it has a duration of p_i^m. Moreover, the activity requires r_{ik}^m units of each renewable resource $k \in R$ during execution, i.e., r_{ik}^m units of renewable resource k are occupied while the activity is in progress and are then released upon completion. We denote the availability of renewable resource $k \in R$ as R_k, i.e., the project activities can occupy at most R_k units of this renewable resource simultaneously. Furthermore, the activity consumes w_{ik}^m units of each nonrenewable resource $k \in N$ at its start. We denote the capacity of nonrenewable resource $k \in N$ as W_k, i.e., the activities can consume at most W_k units of this nonrenewable resource throughout the entire project. The two fictitious activities 0 and $n+1$ have a single fictitious mode, and a duration and resource requirement of zero. To simplify the notation, we define $M_i = \{1\}$, $p_i^1 = 0$, $r_{ik}^1 = 0$ $(k \in R)$, and $w_{ik}^1 = 0$ $(k \in N)$ for $i \in \{0, n+1\}$. The project has a scheduling horizon T, which is calculated as the sum of the maximal activity durations, i.e., $T \coloneqq \sum_{i \in \bar{V}} \max_{m \in M_i} p_i^m$. To simplify the notation, we define a discretization of the scheduling horizon as set $H \coloneqq \{0, 1, \ldots, T\}$. Furthermore, we are given a set of completion-start precedence relations $E \subsetneq V \times V$. For each pair of activities with such a precedence relation $(i, j) \in E$, the first activity i must be completed before the latter activity j begins. We denote the transitive closure of the precedence relations as TE, i.e., if $(i,j) \in E$ and $(j,l) \in E$, then the pair (i,l) is included in TE. To simplify the notation, we define the set of all pairs of activities that are not related through a precedence relation from TE as $V_p := \{(i,j) \in V \times V : i \neq j, (i,j) \notin TE, (j,i) \notin TE\}$. Based on these precedence relations, for each activity $i \in V$, we can calculate the earliest start time (ES_i) and the latest start time (LS_i) via forward and backward pass calculations (cf., e.g., Demeulemeester & Herroelen, 2002), for which the execution mode with the shortest duration is used (cf. Talbot, 1982). Note that ES_{n+1} is a lower bound on the minimal project makespan; we denote this lower bound as CPM. The problem comprises determining a start time and an execution mode for each activity such that the project makespan is minimized. Thereby, the precedence relations, renewable resource availabilities and nonrenewable resource capacities must be respected.

We illustrate the problem setting of the MRCPSP using an exemplary project comprising $n = 4$ real activities, with the two fictitious activities $i = 0$ and $i = 5$ representing the project start and completion, respectively. Each real activity i can be executed in two different modes, i.e., $M_i = \{1, 2\}$ $(i \in \bar{V})$, and the scheduling horizon is $T = 13$. The activities require capacities of one renewable resource ($R = \{1\}$), with an availability of $R_1 = 3$, and of one nonrenewable resource ($N = \{2\}$), with a capacity of $W_2 = 7$. The precedence relations of the considered example are displayed in the activity-on-node graph in Figure 2.1. Furthermore, the graph contains the activities' durations and resource requirements in the two possible execution modes below their corresponding nodes. We display the earliest and latest start times of all activities $i \in V$ in Table 2.2.

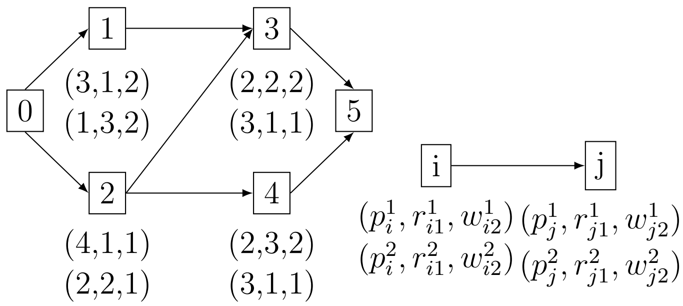


Figure 2.1: Illustrative example - Precedence relations, durations and resource requirements

i	0	1	2	3	4	5
ES_i	0	0	0	2	2	4
LS_i	9	10	9	11	11	13

Table 2.2: Illustrative example - Earliest and latest start times

An optimal project schedule for the described exemplary instance is illustrated in Figure 2.2. We display the renewable resource usage $r_1(t)$ as a function of time and the nonrenewable resource consumption w_2 as bars. The minimal makespan of this exemplary instance is five.

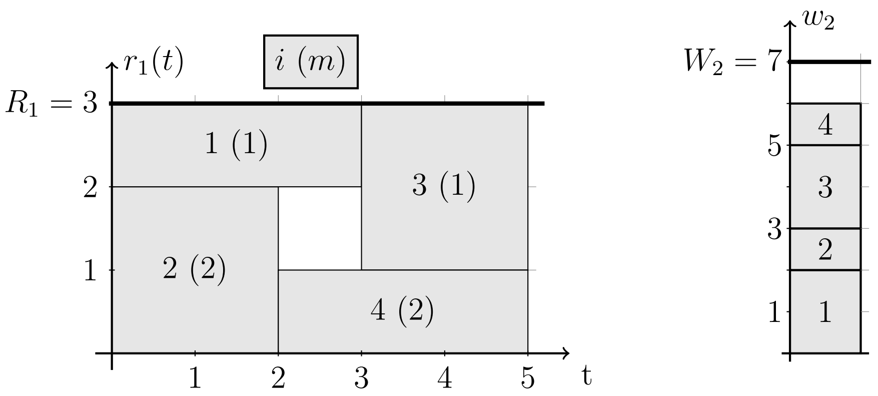


Figure 2.2: Illustrative example - Optimal solution

2.3 Discrete-time models

In this section, we present the extended DT models. Several types of binary DT variables have been presented for RCPSP models in the literature (cf., e.g., Artigues et al., 2015 for an overview); among them are pulse, step, and on/off variables. For the RCPSP, these variables are defined for each activity $i \in V$ and each point in time $t \in H$. To adapt these variable types for the MRCPSP, they must additionally be defined for each possible execution mode of the activity. We illustrate the variables used in each of the DT models using the optimal solution to our illustrative example. Additionally, for each of these variable types, we disaggregate the precedence constraints in the respective DT models, which yields stronger LP relaxations for the RCPSP (cf., e.g., Christofides et al., 1987, Artigues, 2017).

In Subsection 2.3.1, we present the aggregated model based on pulse variables (cf. Talbot, 1982) and a model using a disaggregated version of the precedence constraints. In Subsection 2.3.2, we present an aggregated and a disaggregated model based on step variables. In Subsection 2.3.3, we present a model that uses on/off variables and disaggregated precedence constraints. In Subsection 2.3.4, we present an aggregated and a disaggregated version of a DT model that combines pulse and on/off variables. We summarize the properties of all DT models in Table 2.3.

Abbreviation	Pulse	Step	On/off	Dis.	Single-mode	Multi-mode
PDT	x				Pritsker et al. (1969)	Talbot (1982)
PDDT	x			x	Christofides et al. (1987)	This paper
SDT		x			Klein (2000)	This paper
SDDT		x		x	Klein (2000)	This paper
OODDT			x	x	Klein (2000)	This paper
OOPDT	x		x		Kopanos et al. (2014)	This paper
OOPDDT	x		x	x	Kopanos et al. (2014)	This paper

Table 2.3: Summary of discrete-time models

2.3.1 Models using pulse variables

Pulse variables are binary variables indicating, for each point in time $t \in H$, whether an activity starts at that point in time. They were used to address the RCPSP for the first time by Pritsker et al. (1969). For the MRCPSP, the variables additionally indicate a possible execution mode for the activity. We denote pulse variables as x_{it}^m $(i \in V, t \in H, m \in M_i)$, with $x_{it}^m = 1$ if activity i starts at time t in execution mode m and $x_{it}^m = 0$ otherwise. We illustrate the pulse variables using the optimal solution to our illustrative example in Table 2.4, with a gray cell indicating that the corresponding variable is equal to one.

i	m	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1														
1	1														
	2														
2	1														
	2														
3	1														
	2														
4	1														
	2														
5	1														

Table 2.4: Illustration of pulse variables

Talbot (1982) proposes an integer linear programming (ILP) model using pulse variables and aggregated precedence constraints to address the MRCPSP. The model, which we denote as PDT, reads as follows:

$$\min \sum_{t \in H} t \cdot x^1_{n+1,t} \tag{2.1}$$

$$\text{s.t.} \sum_{t \in H} \sum_{m \in M_i} x^m_{it} = 1 \qquad \forall\, i \in V \tag{2.2}$$

$$\sum_{t \in H} \sum_{m \in M_i} (t + p^m_i) \cdot x^m_{it} \leq \sum_{t \in H} \sum_{m \in M_j} t \cdot x^m_{jt} \qquad \forall\, (i,j) \in E \tag{2.3}$$

$$\sum_{i \in V} \sum_{m \in M_i} \sum_{\tau = t - p^m_i + 1}^{t} r^m_{ik} \cdot x^m_{i\tau} \leq R_k \qquad \forall\, t \in H,\ k \in R \tag{2.4}$$

$$\sum_{i \in V} \sum_{t \in H} \sum_{m \in M_i} w^m_{ik} \cdot x^m_{it} \leq W_k \qquad \forall\, k \in N \tag{2.5}$$

$$x^m_{it} = 0 \qquad \forall\, i \in V,\ m \in M_i,\ t \in H \setminus \{ES_i, \ldots, LS_i\} \tag{2.6}$$

$$x^m_{it} \in \{0,1\} \qquad \forall\, i \in V,\ t \in H,\ m \in M_i \tag{2.7}$$

The objective function (2.1) aims to minimize the makespan, i.e., the start time of fictitious activity $n+1$. Constraints (2.2) ensure that each activity starts at exactly one point in time and is executed in exactly one mode. Constraints (2.3) assure that for each precedence relation $(i,j) \in E$, activity j starts after activity i is completed. Note that we formulate one constraint for each precedence relation, which is the aggregated version of the precedence constraints. Constraints (2.4) ensure that the total renewable resource requirements of all activities that are in progress at time $t \in H$ do not exceed the availability R_k of each renewable resource $k \in R$. To present the models more clearly and simplify the notation, we assume variables with a negative index (e.g., $x^m_{i\tau}$ if $t - p^m_i + 1 < 0$ in constraints (2.4)) to represent value zero in the remainder of this section. Constraints (2.5) ensure that the total nonrenewable resource consumption of all activities does not exceed the availability W_k of each nonrenewable resource $k \in W$. Constraints (2.6) set the earliest and latest start times of the activities.

Christofides et al. (1987) propose a disaggregated version of the precedence constraints for the RCPSP. They formulate a constraint for each precedence relation and each point in time. For the MRCPSP model with pulse variables, the disaggregated version reads as follows:

$$\sum_{m \in M_i} \sum_{\tau=0}^{t-p_i^m} x_{i\tau}^m \geq \sum_{m \in M_j} \sum_{\tau=0}^{t} x_{j\tau}^m \qquad \forall\ (i,j) \in E,\ t \in H \quad (2.8)$$

Constraints (2.8) state that for each precedence relation $(i,j) \in E$ and each point in time $t \in H$, activity j can only start before or at time t if activity i starts in mode $m \in M_i$ before or at time $t - p_i^m$. Our disaggregated model with pulse variables results if we replace (2.3) with (2.8); we denote it as PDDT.

2.3.2 Models using step variables

Step variables are binary variables indicating, for each point in time $t \in H$, whether an activity has started at that point in time or before. They have been used to address the RCPSP by Klein (2000). We denote step variables as z_{it}^m $(i \in V, t \in H, m \in M_i)$, with $z_{it}^m = 1$ if activity i starts at time t or before in execution mode m and $z_{it}^m = 0$ otherwise. Note that activity i starts at time t in mode m if and only if $z_{it}^m - z_{i,t-1}^m = 1$ holds. We illustrate the step variables using the optimal solution to our illustrative example in Table 2.5, with a gray cell indicating that the corresponding variable is equal to one.

i	m	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1														
1	1														
	2														
2	1														
	2														
3	1														
	2														
4	1														
	2														
5	1														

Table 2.5: Illustration of step variables

Our model using step variables and aggregated precedence constraints, which we denote as SDT, reads as follows:

$$\min \quad \sum_{t=1}^{T} t \cdot (z^1_{n+1,t} - z^1_{n+1,t-1}) \tag{2.9}$$

$$\text{s.t.} \quad \sum_{m \in M_i} z^m_{i,LS_i} = 1 \qquad \forall\, i \in V \tag{2.10}$$

$$z^m_{i,t-1} \leq z^m_{it} \qquad \forall\, i \in V,\ m \in M_i,\ t \in H : t > 0 \tag{2.11}$$

$$\sum_{t \in H} \sum_{m \in M_i} (t + p^m_i) \cdot (z^m_{it} - z^m_{i,t-1}) \leq \sum_{t \in H} \sum_{m \in M_j} t \cdot (z^m_{jt} - z^m_{j,t-1}) \qquad \forall\, (i,j) \in E \tag{2.12}$$

$$\sum_{i \in V} \sum_{m \in M_i} r^m_{ik} \cdot (z^m_{it} - z^m_{i,t-p^m_i}) \leq R_k \qquad \forall\, t \in H,\ k \in R \tag{2.13}$$

$$\sum_{i \in V} \sum_{m \in M_i} w^m_{ik} \cdot z^m_{iT} \leq W_k \qquad \forall\, k \in N \tag{2.14}$$

$$z^m_{it} = 0 \qquad \forall\, i \in V,\ m \in M_i,\ t \in \{0, \ldots, ES_i - 1\} \tag{2.15}$$

$$z^m_{it} \in \{0,1\} \qquad \forall\, i \in V,\ t \in H,\ m \in M_i \tag{2.16}$$

Constraints (2.10) state that each activity $i \in V$ must be performed in exactly one mode and must be started before or at its latest start time LS_i. Constraints (2.11) ensure that if an activity $i \in V$ has been started at or before time $t-1$ in mode $m \in M_i$, it has also started before time t in the same mode, which is particularly important for modeling the renewable resource constraints. Constraints (2.12) state that activity j can only start after activity i is completed for each precedence relation $(i,j) \in E$. Constraints (2.13) ensure the renewable resource availabilities are not exceeded. For this purpose, the left-hand side determines, for each point in time $t \in H$, which activities $i \in V$ are currently in progress, i.e., start at time t or earlier in mode $m \in M_i$ but have not been started in mode m at time $t - p^m_i$ or earlier. If $t - p^m_i < 0$ holds, the corresponding variable in this constraint should again represent value zero. Constraints (2.14) ensure that the nonrenewable resource capacities are not exceeded. Constraints (2.15) set the earliest and latest start times of the activities.

The disaggregated version of precedence constraints (2.12) for the MRCPSP model with step variables reads as follows:

$$\sum_{m \in M_i} z^m_{i,t-p^m_i} \geq \sum_{m \in M_j} z^m_{jt} \qquad \forall\, (i,j) \in E,\ t \in H \tag{2.17}$$

Constraints (2.17) state that for each precedence relation $(i, j) \in E$ and each point in time $t \in H$, activity j can only start at time t or before if activity i starts at time $t - p_i^m$ or before in mode $m \in M_i$. Our disaggregated model with step variables results if we replace (2.12) with (2.17) and is denoted as SDDT.

2.3.3 Models using on/off variables

On/off variables are binary variables indicating whether an activity is currently in progress for each point in time $t \in H$. They have been used to address the RCPSP by Klein (2000). We denote on/off variables as y_{it}^m $(i \in V, t \in H, m \in M_i)$, with $y_{it}^m = 1$ if activity i is in progress at time t in mode m and $y_{it}^m = 0$ otherwise. Note that for fictitious activities with duration zero, the on/off variables can be interpreted as pulse variables. We illustrate the on/off variables using the optimal solution to our illustrative example in Table 2.6, with a gray cell indicating that the corresponding variable is equal to one.

i	m	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1														
1	1														
	2														
2	1														
	2														
3	1														
	2														
4	1														
	2														
5	1														

Table 2.6: Illustration of on/off variables

Our model using on/off variables and disaggregated precedence constraints, which we denote as OODDT, reads as follows:

$$\min \quad \sum_{t \in H} t \cdot y^1_{n+1,t} \tag{2.18}$$

$$\text{s.t.} \quad \sum_{t \in H} \sum_{m \in M_i} \frac{1}{p^m_i} \cdot y^m_{it} = 1 \qquad \forall\, i \in \bar{V} \tag{2.19}$$

$$\sum_{t \in H} y^1_{i,t} = 1 \qquad \forall\, i \in \{0, n+1\} \tag{2.20}$$

$$p^m_i \cdot (y^m_{it} - y^m_{i,t+1}) - \sum_{\tau = t - p^m_i + 1}^{t-1} y^m_{i\tau} \leq 1 \qquad \forall\, i \in V,\ m \in M_i,\ t \in H : t < T \tag{2.21}$$

$$\sum_{\tau=0}^{t-1} \sum_{m \in M_i : p^m_i > 0} \frac{1}{p^m_i} \cdot y^m_{i\tau} \geq \sum_{m \in M_j} y^m_{jt} \qquad \forall\, (i,j) \in E,\ t \in H \tag{2.22}$$

$$\sum_{i \in V} \sum_{m \in M_i} r^m_{ik} \cdot y^m_{it} \leq R_k \qquad \forall\, t \in H,\ k \in R \tag{2.23}$$

$$\sum_{i \in \bar{V}} \sum_{t \in H} \sum_{m \in M_i} \frac{w^m_{ik}}{p^m_i} \cdot y^m_{it} \leq W_k \qquad \forall\, k \in N \tag{2.24}$$

$$y^m_{it} = 0 \qquad \forall\, i \in V,\ m \in M_i, t \in H \setminus \{ES_i, \ldots, LS_i + p^m_i\} \tag{2.25}$$

$$y^m_{it} \in \{0,1\} \qquad \forall\, i \in V,\ t \in H,\ m \in M_i \tag{2.26}$$

Constraints (2.19) and (2.21) ensure that exactly p^m_i consecutive on/off variables corresponding to the chosen execution mode $m \in M_i$ are equal to one for each activity $i \in \bar{V}$. Thereby, it is again important that the same execution mode is chosen for all points in time. Analogously, constraints (2.20) ensure that fictitious activities 0 and $n+1$ start at exactly one point in time. Constraints (2.22) ensure that if activity j is in progress at time $t \in H$, activity i is completed at or before time t, i.e., p^m_i on/off variables are equal to one for points in time $\tau = 0, \ldots, t-1$. Since the on/off variables indicate, for each point in time, whether an activity is currently in progress in a certain mode, it is straightforward to formulate the renewable resource constraints (2.23). For the nonrenewable resource constraints (2.24), we must additionally divide by the duration to ensure that we take each activity into account only once. The earliest and latest start times of the activities are set by constraints (2.25).

Note that the number of on/off variables that are equal to one depends on the execution mode selected for each activity. This dependency prevents us from an efficient aggregation of the precedence constraints. Therefore, we consider only a disaggregated model for on/off variables.

2.3.4 Models using on/off and pulse variables

Kopanos et al. (2014) propose a model for the RCPSP that utilizes both pulse and on/off variables.

Our extended model using pulse and on/off variables with aggregated precedence constraints, which we denote as OOPDT, reads as follows:

$$\min \sum_{t\in H} t \cdot x^1_{n+1,t} \tag{2.27}$$

$$\text{s.t.} \sum_{t\in H}\sum_{m\in M_i} x^m_{it} = 1 \qquad \forall\, i \in V \tag{2.28}$$

$$y^m_{it} = \sum_{\tau=t-p^m_i+1}^{t} x^m_{i\tau} \qquad \forall\, i \in V,\ t \in H,\ m \in M_i \tag{2.29}$$

$$\sum_{t\in H} y^m_{it} = \sum_{t\in H} p^m_i \cdot x^m_{it} \qquad \forall\, i \in V,\ m \in M_i \tag{2.30}$$

$$\sum_{t\in H}\sum_{m\in M_i} (t + p^m_i) \cdot x^m_{it} \le \sum_{t\in H}\sum_{m\in M_j} t \cdot x^m_{jt} \qquad \forall\, (i,j) \in E \tag{2.31}$$

$$\sum_{i\in V}\sum_{m\in M_i} r^m_{ik} \cdot y^m_{it} \le R_k \qquad \forall\, t \in H,\ k \in R \tag{2.32}$$

$$\sum_{i\in V}\sum_{t\in H}\sum_{m\in M_i} w^m_{ik} \cdot x^m_{it} \le W_k \qquad \forall\, k \in N \tag{2.33}$$

$$x^m_{it} = 0 \qquad \forall\, i \in V,\ m \in M_i,\ t \in H \setminus \{ES_i, \dots, LS_i\} \tag{2.34}$$

$$y^m_{it} = 0 \qquad \forall\, i \in V,\ m \in M_i,\ t \in H \setminus \{ES_i, \dots, LS_i + p^m_i\} \tag{2.35}$$

$$x^m_{it},\ y^m_{it} \in \{0,1\} \qquad \forall\, i \in V,\ t \in H,\ m \in M_i \tag{2.36}$$

Constraints (2.29) and (2.30) ensure that both variable types correspond to the same start times and execution modes for the activities. The renewable resource constraints are modeled using the on/off variables, while all other constraints are modeled via the pulse variables.

Since the model uses pulse variables to represent the precedence constraints, we can disaggregate them in the same way as described in Subsection 2.3.1. The disaggregated model with on/off and pulse variables results if we replace (2.31) with (2.8) and is denoted as OOPDDT.

Abbreviation	Ass.	Flow	Seq.	Single-mode	Multi-mode
MCTAB_E	x			Trautmann et al. (2018)	Gnägi et al. (2019)
FCT-S		x		Artigues et al. (2003)	Sayah (2023)
MSEQCT			x	Klein et al. (2024)	This paper

Table 2.7: Summary of continuous-time models

2.4 Continuous-time models

In this section, we discuss CT models for the MRCPSP. In Subsection 2.4.1, we present the sequence-based CT model. In Subsection 2.4.2, we compare this model to existing assignment-based and flow-based CT models from the literature. We summarize the properties of all CT models in Table 2.7.

2.4.1 Sequence-based model

The sequence-based CT model for the RCPSP has been proposed by Klein et al. (2024). It includes continuous variables S_i for all activities $i \in V$ that indicate their respective start times. To indicate the chosen execution mode, we use binary variables x_i^m $(i \in V, m \in M_i)$, with $x_i^m = 1$ if activity i is executed in mode m and $x_i^m = 0$ otherwise. To determine which activities are in progress simultaneously, we use two kinds of binary sequencing variables. The first type are completion-start sequencing variables y_{ij} $((i,j) \in V_p)$, with $y_{ij} = 1$ if activity i is completed before or at the same time as activity j starts and $y_{ij} = 0$ otherwise. The second type are start-start sequencing variables z_{ij} $((i,j) \in V_p)$, with $z_{ij} = 1$ if activity i starts either at any time before or at the same time as activity j and $z_{ij} = 0$ otherwise. At the start of any activity i, we can then determine whether another activity j is currently in progress $(z_{ji} = 1, y_{ji} = 0)$, has already been completed $(z_{ji} = y_{ji} = 1)$, or has not yet been started $(z_{ji} = y_{ji} = 0)$. However, to identify the renewable resource requirement of an activity j at the start of activity i, we must combine this information about the sequence with the information about the execution mode chosen for activity j, leading to a nonlinear term in the corresponding constraint. Therefore, we introduce continuous, nonnegative auxiliary variables u_{ji}^k $((j,i) \in V_p, k \in R)$, representing the renewable resource requirement of activity j at the start of activity i. As $r_{jk}^{\max} := \max_{m \in M_j} r_{jk}^m$ is an upper bound on the renewable resource requirement of activity j, we can bound $u_{ji}^k \in [0, r_{jk}^{\max}]$ $((j,i) \in V_p, k \in R)$. In addition, this implies that we can omit variables u_{ji}^k for all $i \in V$ if $r_{jk}^{\max}$ is zero. We illustrate the variables used in the sequence-based model in Figure 2.3.

We demonstrate how the auxiliary variables are used, taking activity four as an exam-

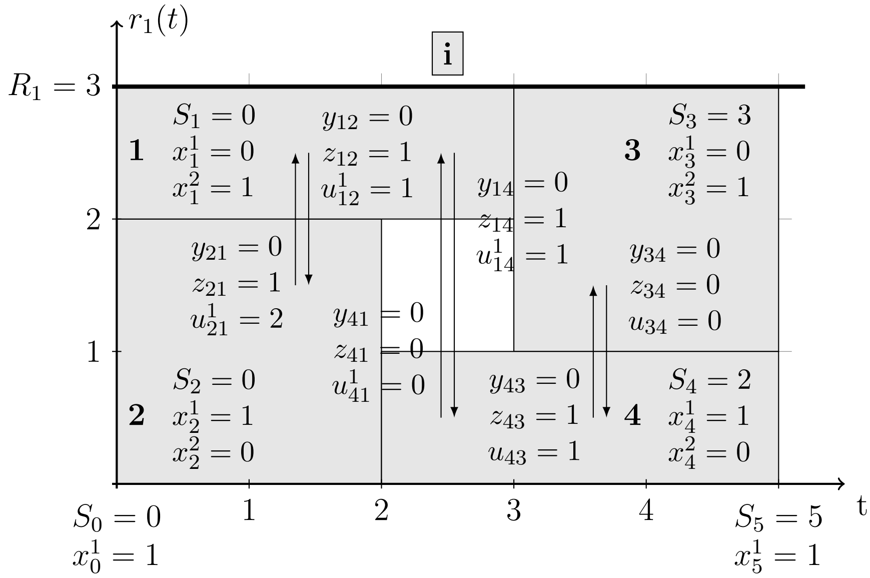


Figure 2.3: Illustration of the variables in the sequence-based model

ple. Activity one has already been started ($z_{14} = 1$) but is not yet completed ($y_{14} = 0$), i.e., it is in progress at the start of activity four. Therefore, we must consider its renewable resource requirement of $u_{14}^1 = 1$ in combination with activity four. Activity three has not yet been started and is hence not completed ($z_{34} = y_{34} = 0$). Therefore, we do not have to consider its renewable resource requirements. We do not have to consider activity two due to the precedence relation $(2, 4) \in TE$.

Our sequence-based model, which we denote as MSEQCT, reads as follows:

$$\min S_{n+1} \tag{2.37}$$

$$\text{s.t. } S_i + \sum_{m \in M_i} p_i^m x_i^m \leq S_j + T(1 - y_{ij}) \qquad \forall\ (i,j) \in V_p \tag{2.38}$$

$$S_i + \sum_{m \in M_i} p_i^m x_i^m \leq S_j \qquad \forall\ (i,j) \in TE \tag{2.39}$$

$$T \cdot z_{ij} \geq S_j - S_i + \epsilon \qquad \forall\ (i,j) \in V_p \tag{2.40}$$

$$\sum_{m \in M_i} x_i^m = 1 \qquad \forall\ i \in V \tag{2.41}$$

$$\sum_{i \in V} \sum_{m \in M_i} w_{ik}^m x_i^m \leq W_k \qquad \forall\ k \in N \tag{2.42}$$

$$\sum_{m \in M_j} r_{jk}^m x_j^m - r_{jk}^{\max} \cdot (1 - z_{ji} + y_{ji}) \leq u_{ji}^k \qquad \forall\ (j,i) \in V_p,\ k \in R \tag{2.43}$$

$$\sum_{m \in M_i} r_{ik}^m x_i^m + \sum_{j \in V:(j,i) \in V_p} u_{ji}^k \leq R_k \qquad \forall\ i \in V,\ k \in R \tag{2.44}$$

$$S_i \in [ES_i, LS_i] \qquad \forall\ i \in V \tag{2.45}$$

$$u_{ji}^k \in [0, r_{jk}^{\max}] \qquad \forall\ (j,i) \in V_p,\ k \in R \tag{2.46}$$

$$x_i^m \in \{0,1\} \qquad \forall\ i \in V,\ m \in M_i \tag{2.47}$$

$$y_{ij}, z_{ij} \in \{0,1\} \qquad \forall\ (i,j) \in V_p \tag{2.48}$$

The objective function (2.37) of minimizing the makespan is modeled as the start time of fictitious activity $n+1$. Constraints (2.38) connect the start-time variables with the completion-start sequencing variables, ensuring that if $y_{ij} = 1$ for a pair of activities $(i,j) \in V_p$, activity j starts after activity i is completed. Constraints (2.39) ensure the completion-start precedence relations. Constraints (2.40) connect the start-start sequencing variables and the timing variables, i.e., z_{ij} is equal to one if $S_i \leq S_j$. Thereby, $\epsilon > 0$ is a sufficiently small parameter to set both variables z_{ij} and z_{ji} equal to one if activities i and j start simultaneously. If all activity durations are integer values, we can set $\epsilon := 1$. Constraints (2.41) enforce that each activity is executed in exactly one mode, and constraints (2.42) ensure that the nonrenewable resource capacities are respected. Constraints (2.43) ensure that auxiliary variables $u_{ji}^k((j,i) \in V_p, k \in R)$ represent the renewable resource requirement of renewable resource k of activity j if it is still in progress at the start of activity i, i.e., $z_{ji} = 1$, and $y_{ji} = 0$. Otherwise, $r_{jk}^{\max}$ serves as a big-M parameter, and the variable can be set to zero. Constraints (2.44) ensure that the renewable resource availabilities are respected using the auxiliary variables. The left-hand side of the constraints represents the requirement of activity $i \in V$, plus the total requirements of all activities that are still in progress at the start of activity i in their respective execution modes, while the right-hand side represents the availability of renewable resource $k \in R$. Finally, bounds (2.45) set the earliest and latest start times of the activities, while bounds (2.46) set the domains of the auxiliary variables.

2.4.2 Continuous-time models from the literature

We compare the sequence-based CT model to the assignment-based CT model of Gnägi et al. (2019) (MCTAB_E) and the best-performing flow-based model of Sayah (2023) (FCT-S).

All four models use continuous variables for all activities to represent their start times. Also, the models use binary variables for each activity and each possible execution mode to indicate whether the activity is executed in that mode. In addition, the models include binary completion-start sequencing variables for pairs of activities to indicate whether one of the activities is completed before the other. The order of magnitude of these variables is $\mathcal{O}(n^2 + \bar{m})$, with $\bar{m} = \sum_{i \in \bar{V}} |M_i|$, and they are sufficient to reflect the precedence relations and the nonrenewable resource constraints. However, the models take different approaches to formulating the renewable resource constraints.

The assignment-based model of Gnägi et al. (2019) uses binary variables for each activity and each renewable resource unit, indicating whether an activity uses the particular unit. The order of magnitude of these variables is $\mathcal{O}(n \cdot \bar{R})$, with $\bar{R} = \sum_{k \in R} R_k$. This implies that the number of variables in this model depends on the resource availabilities, i.e., the model is pseudo-polynomial. The order of magnitude of the constraints in their model is $\mathcal{O}(n^2 + n^2 \cdot \bar{R})$. To improve the performance of their model, Gnägi et al. (2019) additionally include $\mathcal{O}(n^3)$ redundant constraints.

The flow-based model of Sayah (2023) uses continuous flow variables for each pair of activities and each renewable resource. The order of magnitude of these variables is $\mathcal{O}(n^2 \cdot |R|)$. The model comprises $\mathcal{O}(n^3 + n^2 \cdot |R|)$ nonlinear constraints that are replaced with linear constraints of the same order of magnitude.

Our sequence-based model uses binary start-start sequencing variables and continuous auxiliary variables to model the renewable resource constraints. The order of magnitude of these variables is $\mathcal{O}((n^2 - |TE|) \cdot |R|)$. The order of magnitude of the constraints is $\mathcal{O}((n^2 - |TE|) \cdot |R|)$.

Model	DT	CT	Source
PDT	x		Talbot (1982)
PDDT	x		This paper
SDT	x		This paper
SDDT	x		This paper
OODDT	x		This paper
OOPDT	x		This paper
OOPDDT	x		This paper
MCTAB_E		x	Gnägi et al. (2019)
FCT-S		x	Sayah (2023)
MSEQCT		x	This paper

Table 2.8: Overview of the models tested

2.5 Comparison of models

In this section, we compare the extended MILP models to state-of-the-art models. In Subsection 2.5.1, we describe the sets of instances and the software and hardware we used. In Subsection 2.5.2, we report on the comparison of the LP relaxations of all models. In Subsection 2.5.3, we report on the computational performance analysis.

2.5.1 Experimental design

We compared the extended models for the MRCPSP to the DT model introduced by Talbot (1982) and the CT models introduced by Gnägi et al. (2019) and Sayah (2023). All models tested are listed in Table 2.8.

We analyzed the computational performance on six sets of benchmark instances. The first two sets are sets J20 and J30 from the PSPLIB (Kolisch & Sprecher, 1996). These sets contain 554 and 552 problem instances comprising 20 and 30 real activities, respectively. The activities can be executed in three modes; two renewable and two nonrenewable resources are considered. In these problem instances, the activity durations are relatively short. To analyze the impact of longer durations on the performance of the models, we additionally considered two sets of problem instances proposed by Gnägi et al. (2019), called D20 and D30. Sets D20 and D30 arise from sets J20 and J30 by applying a modification procedure proposed for the single-mode RCPSP by Koné et al. (2011). In this modification, the durations of all modes are increased by a random factor for half of the activities. Thus, the scheduling horizon in these problem instances is relatively large. In addition to these problem instances that comprise 30 or fewer activities, we consider problem instances comprising 50 activities from the MMLIB (Van Peteghem & Vanhoucke, 2014), called MMLIB50 and MMLIB+. Set MMLIB50 contains 540 instances, and we considered the 1,620 instances from set MMLIB+ which comprise 50 project activities. In the problem instances from set MMLIB50, two renewable and two nonrenewable resources are considered, and the activities can be executed in three modes. In the problem instances from set MMLIB+, two or four renewable and two or four nonrenewable resources are considered; the activities can be executed in three, six, or nine modes.

We implemented all models in Python 3.8 and used Gurobi version 9.5.1 as the MILP solver. We carried out the computations on a workstation with two 8-core Intel(R) Xeon(R) E5-2687 W CPUs (3.1 GHz) and 128 GB RAM. We set a time limit of 500 seconds per instance and restricted the number of available threads to two. Since the durations of all activities take integer values in the problem instances we considered, the minimal makespan of these problem instances is also an integer value. We utilize this fact in the computational performance analysis for all CT models by restricting the start time of fictitious activity S_{n+1} to take integer values and prohibit the solver from branching on this variable. This enables the solver to implicitly round up the lower bounds found during the branch-and-bound process to the next integer, which enhances the performance (cf., e.g., Klein et al., 2024). Note that this approach is implicitly utilized in the DT models. Additionally, we set $\epsilon := 1$ in constraints (2.40) for the model MSEQCT.

2.5.2 Comparison of the LP relaxations

We compared the LP relaxations of the models using the sets of instances with relatively short activity durations, i.e., J20, J30, MMLIB50, and MMLIB+. We report the following metrics regarding the LP relaxations; the best value per metric is marked in bold in all following tables:

- OFV^{LPR}: Average optimal objective function value (OFV) of the LP relaxations
- Gap^{LPR}: Average relative deviation between the optimal OFV of the LP relaxation (LPR^*) and the CPM lower bound, calculated as $(\text{LPR}^* - \text{CPM})/\text{CPM}$
- Time: Average time (in seconds) used to solve the LP relaxation of an instance
- # Vars.: Average number of variables the model comprises
- # Cons.: Average number of constraints the model comprises

The results regarding the LP relaxations for the instances of set J20 are presented in Table 2.9.

Model	OFV$^{\text{LPR}}$	Gap$^{\text{LPR}}$	Time	# Vars.	# Cons.
PDT	25.41	6.97%	0.05	8,539.9	**379.6**
PDDT	**25.89**	**9.08%**	0.58	8,539.9	5,520.5
SDT	25.41	6.97%	0.56	9,907.4	10,229.0
SDDT	**25.89**	**9.08%**	4.08	9,907.4	15,941.6
OODDT	23.90	0.14%	0.70	9,747.6	15,782.8
OOPDT	25.41	6.97%	0.09	17,273.5	9,111.1
OOPDDT	**25.89**	**9.08%**	0.75	17,273.5	14,252.1
MCTAB_E	25.20	5.99%	0.21	1,227.4	6,168.3
FCT-S	25.16	5.82%	0.81	1,536.0	11,316.5
MSEQCT	25.20	5.99%	**0.00**	**800.0**	818.0

Table 2.9: Comparison of LP relaxations for the instances of set J20

Regarding the DT models, we observe that the LP relaxations of the models with aggregated precedence relations (PDT, SDT, and OOPDT) have the same optimal OFVs for all instances. The same holds for the LP relaxations of the corresponding models with disaggregated precedence relations (PDDT, SDDT, and OOPDDT). In what follows, we show, using variable transformations, that the LP relaxations of these models are equivalent, i.e., they have the same optimal OFV for any instance of the MRCPSP. These results match the equivalences of the LP relaxations of DT models for the single-mode RCPSP presented by Artigues (2017), who uses similar transformations.

Consider the following transformations of pulse and step variables:

$$x_{it}^m = z_{it}^m - z_{i,t-1}^m \quad \forall\, i \in V,\ m \in M_i,\ t \in H \tag{2.49}$$

$$z_{it}^m = \sum_{\tau=0}^{t} x_{i\tau}^m \quad \forall\, i \in V,\ m \in M_i,\ t \in H \tag{2.50}$$

These transformations imply that any feasible solution for a model using pulse variables can be transformed into a feasible solution for a model using step variables, and vice versa. Therefore, the LP relaxations of the models PDT and SDT, as well as the LP relaxations of models PDDT and SDDT have the same optimal objective function value.

Consider the following representation of on/off variables as pulse variables (cf. constraints (2.29)):

$$y_{it}^m = \sum_{\tau=t-p_i^m+1}^{t} x_{i\tau}^m \quad \forall\, i \in V,\ m \in M_i,\ t \in H \tag{2.51}$$

This transformation implies that any feasible solution for a model using pulse variables can be transformed into a feasible solution for a model using on/off variables. Therefore, using on/off variables in addition to pulse variables does not strengthen the LP relaxation of a model. It follows that the LP relaxations of models PDT and OOPDT, as well as the LP relaxations of models PDDT and OOPDDT have the same optimal objective function value.

Regarding the model OODDT, we observe that the LP relaxations are considerably weaker than those of any other model. This implies that no corresponding representation of a pulse variable as on/off variables can be obtained. One potential explanation for this is that the number of on/off variables that are equal to one depends on the chosen execution mode of an activity.

Regarding the CT models, we observe that the LP relaxations of the assignment-based model (MCTAB_E) and the sequence-based model (MSEQCT) have the same optimal OFVs for all instances, while the LP relaxations of the state-of-the-art flow-based model (FCT-S) are slightly weaker, on average. In addition, we observe that the optimal OFVs of the LP relaxations of MSEQCT remained unchanged after removing constraints (2.43). This suggests that the optimal OFVs were only determined by the precedence relations and the nonrenewable resource constraints, but not affected by the renewable resource constraints. The slightly lower average OFV of model FCT-S may be caused by infeasible modes, i.e., modes for which the renewable resource requirement of an activity exceeds the availability, which are present in some instances of set J20.

When comparing the LP relaxations of the DT and CT models, we observe that, on average, the optimal OFVs of the DT models are larger than those of the CT models. In particular, the optimal OFVs of the aggregated DT models are about 0.21 or 0.8% larger and the optimal OFVs of the disaggregated DT models are about 0.69 or 2.7% larger than the optimal OFVs of the CT models. In addition to this weaker quality, the LP relaxations of the CT models MCTAB_E and FCT-S require more time to be solved than those of PDT. However, the LP relaxations of MSEQCT are solved more than ten times faster than all other models (0.0039 seconds, on average). Therefore, we observe a trade-off between the quality of LP relaxations and the CPU time required to solve them. This trade-off arises due to the proposed disaggregated DT models, whose LP relaxations have the largest optimal OFVs, but also require the most CPU time to be solved, and due to the sequence-based CT model, whose LP relaxations have smaller optimal OFVs compared to PDT, but can be solved considerably faster.

The results regarding the LP relaxations for the instances of set J30, MMLIB50, and MMLIB+ are presented in Tables 2.10, 2.11, and 2.12, respectively.

The trade-off we observed for the instances of set J20 can also be observed for the

Model	OFV^{LPR}	Gap^{LPR}	Time	# Vars.	# Cons.
PDT	31.50	5.60%	0.13	19,648.9	**568.2**
PDDT	**31.98**	**7.34%**	2.32	19,648.9	12,378.8
SDT	31.50	5.60%	1.82	22,090.8	22,571.1
SDDT	**31.98**	**7.34%**	15.99	22,090.8	35,479.4
OODDT	29.95	0.05%	2.37	21,850.7	35,240.3
OOPDT	31.50	5.60%	0.32	39,580.8	20,498.2
OOPDDT	**31.98**	**7.34%**	3.27	39,580.8	32,308.8
MCTAB_E	31.22	4.53%	1.52	2,332.0	18,653.7
FCT-S	31.22	4.53%	3.47	3,196.0	34,012.2
MSEQCT	31.22	4.53%	**0.01**	**1,966.2**	1,992.2

Table 2.10: Comparison of LP relaxations for the instances of set J30

Model	OFV^{LPR}	Gap^{LPR}	Time	# Vars.	# Cons.
PDT	33.25	12.25%	0.87	56,976.1	**1,053.0**
PDDT	**34.15**	**15.62%**	74.76	56,976.1	75,274.3
SDT	33.25	12.25%	13.79	61,051.6	61,964.3
OOPDT	33.25	12.25%	2.59	114,403.2	58,478.1
MCTAB_E	32.60	9.60%	2.63	5,617.2	58,625.9
FCT-S	32.60	9.60%	44.36	8,316.0	143,730.0
MSEQCT	32.60	9.60%	**0.07**	**4,955.6**	5,103.2

Table 2.11: Comparison of LP relaxations for the instances of set MMLIB50

Model	OFV^{LPR}	Gap^{LPR}	Time	# Vars.	# Cons.
PDT	**65.09**	**44.13%**	50.16	346,075.1	**3,700.1**
MCTAB_E	61.41	34.43%	35.77	10,208.9	116,104.3
FCT-S	61.41	34.43%	96.36	11,170.0	146,438.0
MSEQCT	61.41	34.43%	**0.15**	**6,481.0**	6,528.2

Table 2.12: Comparison of LP relaxations for the instances of set MMLIB+

instances comprising 30 or more activities. For the instances of set J30, the OFVs of the DT models are about 0.28 (0.9%) and 0.76 (2.4%) larger than the OFVs of the CT models, while the MSEQCT model is solved about 13 and 232 times faster. For the instances of set MMLIB50, the DT models using disaggregated precedence constraints exceeded the time limit of 500 seconds for many instances. Therefore, we only report results for the model PDDT as representative for all disaggregated models. We observe that the OFVs of the DT models are about 0.65 (2%) and 1.55 (4.8%) larger than the OFVs of the CT models. For this set of instances, the model MSEQCT is solved about 12 times faster than the model PDT. Regarding the model PDDT, we observe a significant increase in CPU time (over 70 seconds, on average) which leads to the model MSEQCT being solved more than 1,000 times faster. For the instances of set MMLIB+, we only report on the LP relaxations of the model PDT as representative for all DT models. The OFVs of the model PDT are about 3.68 (6%) larger than the OFVs of the CT models, on average. For this set of instances, the CPU time required is considerably larger for the model PDT (over 50 seconds, on average). Therefore, the model MSEQCT can be solved more than 330 times faster, on average. Notably, the LP relaxations of the assignment-based model (MCTAB_E) are solved faster than those of PDT, on average, which stands in contrast to the three other sets of instances considered.

2.5.3 Computational performance analysis

For all six sets of instances, we report the following key metrics:

- #Feas: Number of instances for which a feasible solution was found
- #Opt: Number of instances for which a proven optimal solution was found
- #Best: Number of instances for which the best feasible solution among all models was found
- Gap^{MIP}: Average relative deviation between the objective function value of the best feasible solution (UB^*) and the best lower bound (LB^*), calculated as $(\text{UB}^* - \text{LB}^*)/\text{UB}^*$
- Gap^{UB^*}: Average relative deviation between UB^* and the CPM lower bound, calculated as $(\text{UB}^* - \text{CPM})/\text{CPM}$; note that we consider only instances for which all models have found a feasible solution; for set D20, we consider only instances for which the CT models and models PDT, PDDT, SDDT, OODDT, and OOPDT have

found a feasible solution; for set D30, we consider only instances for which the CT models and models PDT, PDDT, and OOPDT have found a feasible solution.

- Gap^{LB^*}: Average relative deviation between LB^* and CPM, calculated as $(\text{LB}^* - \text{CPM})/\text{CPM}$
- Time: Average time (in seconds) used to build the model and solve an instance

2.5.3.1 PSPLIB instances

First, we report on the results for sets J20 and J30 from the PSPLIB. The results are presented in Tables 2.13 and 2.14, respectively.

For set J20, the DT models based on pulse variables (PDT and PDDT) outperform the other DT models, devising more proven optimal and best feasible solutions. The second-best performing DT models are OOPDT and OOPDDT, which devise only slightly less proven optimal and best feasible solutions. The model PDDT with disaggregated precedence constraints has a smaller MIP gap, which is justified by the better lower bounds it yields. Despite its slightly worse quality of LP relaxations, the model PDT produces better lower bounds than all other disaggregated models (SDDT, OODDT, and OOPDDT). One potential explanation for this is that the LP relaxations of PDT are solved considerably faster.

From the CT models, the sequence-based CT model (MSEQCT) outperforms the other CT models regarding all metrics considered. It devises proven optimal solutions to 45 more instances, has a smaller MIP gap, and is, on average, more than four times faster than the current state-of-the-art CT model (FCT-S). This flow-based model performs second-best, devising more proven optimal and best feasible solutions than the model MCTAB_E. However, its lower bounds are, on average, smaller. One potential explanation for this is that the LP relaxations of MCTAB_E are solved faster.

Compared to the DT models, the model MSEQCT devises more proven optimal and best feasible solutions and is approximately 40% faster, on average. However, the lower bounds found by MSEQCT are slightly smaller than those of PDT and PDDT, which explains its larger MIP gap.

For set J30, the model PDT outperforms the other DT models regarding all metrics considered, including the lower bounds. A potential explanation for this is that the higher quality of the LP relaxations of the disaggregated models is offset by the smaller CPU time required to solve the LP relaxations of PDT. In addition, the disaggregated models cannot devise feasible solutions to all instances, which can be explained by the larger number of constraints in these models. The second-best performing models are PDDT,

Model	#Feas	#Opt	#Best	Gap^{MIP}	Gap^{UB^*}	Gap^{LB^*}	Time
PDT	**554**	542	550	0.15%	**15.15%**	16.75%	22.2
PDDT	**554**	542	550	**0.10%**	**15.15%**	**16.84%**	40.2
SDT	539	435	460	3.81%	17.19%	11.99%	143.4
SDDT	549	474	510	1.60%	15.82%	14.92%	147.2
OODDT	545	395	434	5.47%	18.69%	10.44%	184.1
OOPDT	**554**	531	545	0.33%	17.04%	16.52%	45.2
OOPDDT	**554**	527	546	0.26%	15.19%	16.63%	61.1
MCTAB_E	**554**	497	531	1.14%	15.37%	15.42%	64.7
FCT-S	**554**	504	533	1.61%	15.34%	14.57%	57.7
MSEQCT	**554**	**549**	**551**	0.19%	**15.15%**	16.66%	**12.5**

Table 2.13: Computational results for the instances of set J20

OOPDT, and OOPDDT, which all include pulse variables. In combination with the results for set J20, this suggests that a modeling based on pulse variables has an edge over other DT variable types for the MRCPSP, with the aggregated DT model (PDT) performing best. This stands in contrast to the performance of the corresponding models for the single-mode RCPSP (cf., e.g., Koné et al., 2011, Kopanos et al., 2014). A possible explanation is the step constraints, which ensure that the same execution mode must be chosen for an activity at all points in time.

Among the CT models, MSEQCT outperforms the current state-of-the-art CT model, devising considerably more optimal and best feasible solutions in, on average, about 40% less CPU time. It also devises better lower bounds, which can be explained by the smaller CPU time required to solve the LP relaxations. The second-best performing CT model is, again, FCT-S.

Compared to the model PDT, the model MSEQCT devises worse lower bounds and, thus, cannot prove optimality for as many instances. However, it devises feasible solutions of comparable quality, supported by the average gap from the best-found upper bound to the CPM-based lower bound being only 0.02 percentage points larger.

2.5.3.2 Impact of higher durations

Second, we report the results for sets D20 and D30, which have comparably long activity durations. The results for sets D20 and D30 are presented in Tables 2.15 and 2.16, respectively.

For set D20, the DT models cannot devise feasible solutions to all instances. A possible explanation is the large number of variables and constraints in the models. This is

Model	#Feas	#Opt	#Best	Gap^{MIP}	Gap^{UB^*}	Gap^{LB^*}	Time
PDT	**552**	**491**	**529**	**1.04%**	**4.82%**	**11.25%**	**71.7**
PDDT	549	476	501	3.69%	7.01%	11.06%	131.8
SDT	439	341	356	3.13%	10.45%	7.06%	233.5
SDDT	441	337	346	9.27%	68.99%	7.14%	395.7
OODDT	533	353	363	10.14%	8.58%	2.86%	228.9
OOPDT	547	473	487	2.70%	4.91%	10.30%	103.0
OOPDDT	549	471	494	2.53%	4.86%	10.88%	136.3
MCTAB_E	**552**	428	448	4.54%	5.06%	7.48%	124.6
FCT-S	**552**	432	465	4.41%	4.94%	6.92%	125.0
MSEQCT	**552**	481	514	2.74%	4.84%	8.58%	74.5

Table 2.14: Computational results for the instances of set J30

supported by the fact that the models with aggregated precedence constraints outperform their disaggregated counterparts. Among the DT models, the model PDT performs best, devising the most proven optimal and best feasible solutions.

Among the CT models, all models devise feasible solutions to all instances. The model MSEQCT even devises a proven optimal solution to all instances in, on average, less than one second per instance. It is, therefore, over ten times faster than the second-fastest model and outperforms all DT and CT models considered regarding all metrics.

Model	#Feas	#Opt	#Best	Gap^{MIP}	Gap^{UB^*}	Gap^{LB^*}	Time
PDT	509	446	447	5.36%	3.13%	7.19%	192.9
PDDT	489	143	143	53.81%	179.98%	-	657.8
SDT	263	230	232	1.95%	-	2.64%	393.5
SDDT	409	0	0	74.38%	311.53%	1.57%	507.6
OODDT	438	100	105	56.70%	12.24%	-	799.2
OOPDT	409	372	378	2.45%	2.97%	5.56%	230.9
OOPDDT	129	42	43	55.65%	-	-	642.4
MCTAB_E	**554**	546	**554**	0.03%	**1.20%**	10.32%	12.6
FCT-S	**554**	542	551	0.22%	**1.20%**	10.00%	27.3
MSEQCT	**554**	**554**	**554**	**0.00%**	**1.20%**	**10.38%**	**0.7**

Table 2.15: Computational results for the instances of set D20

For set D30, the DT models also cannot devise a feasible solution to all instances. Among these models, the aggregated model based on pulse variables (PDT) performs best, similar to the performance for the instances of set D20. Among the CT models, again, the model MSEQCT performs best regarding all metrics considered. It devises the best feasible solution among all models to all instances and proven optimal solutions to over 99% of them.

Model	#Feas	#Opt	#Best	Gap^{MIP}	Gap^{UB^*}	Gap^{LB^*}	Time
PDT	507	467	468	3.22%	2.29%	2.64%	172.5
PDDT	484	16	17	78.73%	316.11%	-	795.5
SDT	134	123	125	1.23%	-	0.85%	480.0
SDDT	375	0	0	75.33%	332.45%	0.12%	511.6
OODDT	211	43	43	73.64%	-	-	973.4
OOPDT	437	418	418	1.31%	3.78%	1.71%	203.1
OOPDDT	16	1	1	87.74%	-	-	766.5
MCTAB_E	**552**	515	527	0.47%	**0.55%**	4.09%	45.5
FCT-S	**552**	478	505	1.84%	0.56%	2.44%	82.0
MSEQCT	**552**	**548**	**552**	**0.01%**	**0.55%**	**4.60%**	**8.8**

Table 2.16: Computational results for the instances of set D30

2.5.3.3 MMLIB instances

Third, we report on the results for sets MMLIB50 and MMLIB+ that comprise 50 activities per problem instance. Due to their larger size, we consider only three benchmark models, PDT, MCTAB_E, and FCT-S, and the best-performing extended model, MSEQCT. The results are presented in Tables 2.17 and 2.18, respectively.

For set MMLIB50, the model PDT devises the most proven optimal and best feasible solutions and further devises the best lower bounds, on average. However, it does not devise a feasible solution to 27 instances. The CT model MSEQCT devises a feasible solution to all instances and, on average, devises significantly better feasible solutions than do all reference models, which is highlighted by the relatively small gap of the best found upper bound to the CPM lower bound. Moreover, it is the fastest model, on average. For instances for which all models devised the optimal solution, MSEQCT was able to prove optimality in, on average, 5.7 seconds, which is more than five times faster than all other models. The relatively small average MIP gap of the model FCT-S can be explained by the relatively small number of feasible solutions it produces, since the average for FCT-S considers 457 instances, while the average for the other three models considers 513, 539, and 540 instances, respectively. When considering only those instances for which all four models found a feasible solution, MSEQCT has the smallest average MIP gap of 2.08%.

For set MMLIB+, the model PDT again yields the best lower bounds, on average.

Model	#Feas	#Opt	#Best	Gap$^{\text{MIP}}$	Gap$^{\text{UB}^*}$	Gap$^{\text{LB}^*}$	Time
PDT	513	**416**	**479**	4.97%	43.21%	**19.85%**	160.6
MCTAB_E	539	325	342	11.51%	20.71%	13.23%	224.9
FCT-S	457	351	382	**4.75%**	30.73%	13.19%	208.2
MSEQCT	**540**	389	444	6.96%	**16.16%**	13.76%	**156.4**

Table 2.17: Computational results for the instances of set MMLIB50

However, it fails to devise a feasible solution to over 50% of the instances. While the model FCT-S also fails to devise feasible solutions to over 290 instances, MCTAB_E and MSEQCT yield feasible solutions for over 98% of the instances. Notably, the model FCT-S produces more proven optimal solutions and a smaller average MIP gap than the models PDT and MCTAB_E. However, the model MSEQCT devises the most proven optimal solutions among all four models. It also devises the best feasible solution to over 75% of the instances and, on average, yields approximately 15 times better feasible solutions than PDT, three times better solutions than MCTAB_E, and over five times better solutions than FCT-S. This is particularly important when these models are embedded in heuristics that utilize mathematical programming (cf., e.g., Fernandes & de Souza, 2021, Bigler et al., 2024). One potential explanation for the superior performance of the model MSEQCT is that its LP relaxations, which are repeatedly solved throughout the branch-and-bound process, can be solved in less than one second while all other models require considerably more time.

Model	#Feas	#Opt	#Best	Gap$^{\text{MIP}}$	Gap$^{\text{UB}^*}$	Gap$^{\text{LB}^*}$	Time
PDT	803	321	455	35.85%	700.56%	**49.61%**	478.6
MCTAB_E	**1619**	223	305	39.26%	148.09%	40.74%	464.5
FCT-S	1328	334	600	24.12%	261.87%	41.60%	421.5
MSEQCT	1598	**348**	**1251**	**22.29%**	**45.26%**	41.82%	**410.1**

Table 2.18: Computational results for the instances of set MMLIB+

2.5.3.4 Summary of results

In summary, the results of our computational study indicate that, in contrast to the single-mode RCPSP, the DT models based on pulse variables remain state-of-the-art among the DT models. For small instances, the disaggregation of precedence constraints slightly improves the lower bounds found and, therefore, reduces the MIP gap. For large problem instances, the model PDT still yields good-quality lower bounds but cannot devise feasible solutions for a considerable number of instances. Among the CT models, the sequence-based model (MSEQCT) outperforms the flow-based and assignment-based reference models considerably, devising better feasible solutions and better lower bounds, on average, for all sets of instances considered.

When comparing the PDT and MSEQCT models, i.e., the best-performing DT and CT models, we observe that they yield proven optimal solutions to 61% and 66% of problem instances, respectively, within 500 seconds of CPU time. Thereby, the model PDT yields better lower bounds and can prove optimality for more instances of sets J30 and MMLIB50 than MSEQCT. However, the model MSEQCT outperforms all models considered on instances with relatively large activity durations and yields significantly better feasible solutions for large instances, especially those of set MMLIB+ that comprise relatively many resources and possible execution modes. A potential explanation for this is that the LP relaxations of model MSEQCT can be solved considerably faster than those of all other models. Additionally, the model comprises the fewest combined variables and constraints, which may make it easier for a solver to devise high-quality feasible solutions. Finally, we note that we observed a similar outperformance of the model MSEQCT when setting different time limits (60 and 300 seconds per instance), and when using a non-commercial MILP solver (SCIP, cf. Bolusani et al., 2024).

2.6 Conclusions

In this paper, we proposed six discrete-time (DT) and one continuous-time (CT) MILP model for the MRCPSP, based on models for the single-mode RCPSP. The DT models utilize pulse, step, and on/off variables, as well as the disaggregation of precedence constraints, which are known to yield superior computational performance or LP relaxations for the RCPSP. The CT model utilizes a combination of two kinds of sequencing variables to detect which activities are in progress simultaneously, and auxiliary variables to avoid nonlinearity. We compared the LP relaxations of the models and showed that the LP relaxations of the three aggregated DT models are equivalent, while the LP relaxations of the three disaggregated versions are also equivalent. Although the LP relaxations of all CT models are weaker than those of the DT models, the LP relaxations of the proposed sequence-based CT model can be solved considerably faster than those of the other models. Therefore, there is a trade-off between the optimal objective function values of the LP relaxations and the CPU time required to solve them. In a computational study, we compared the computational performance of the models on six sets of benchmark instances with varying characteristics. We observed that the current state-of-the-art DT model, which is based on pulse variables, remains state-of-the-art among the DT models, which stands in contrast to the computational performance for the single-mode RCPSP. Among the CT models, the proposed sequence-based model outperforms the current state-of-the-art CT model on all sets of problem instances considered. It also outperforms all DT and CT models considered on instances with relatively large activity durations. Furthermore, it yields good-quality feasible solutions for large instances from the MMLIB for which the DT model based on pulse variables cannot devise a feasible solution for a considerable number of instances. A possible explanation for this is that the LP relaxations of the sequence-based model can be solved considerably faster. In addition, it comprises fewer variables than all other models.

Based on our results, we can derive managerial implications for practitioners facing resource-constrained project scheduling problems involving multiple modes. In particular, the computational performance analysis suggests that DT models based on pulse variables consistently provide the best lower bounds. In addition, they devise feasible solutions of good quality for instances comprising 30 or fewer activities with relatively short durations. Therefore, these models are useful for instances that satisfy these properties, and for establishing lower bounds on the optimal project makespan. However, if the problem instances practitioners face are relatively large or comprise activities with comparably long durations, the sequence-based CT model devises better feasible solutions, on average. This can be observed especially for instances from set MMLIB+, which include activities with up to nine modes, and four renewable and nonrenewable resources. Therefore, the sequence-based CT model can be useful for practitioners who are mainly interested in feasible solutions of good quality for a wide range of problem instances. In addition, the considerable number of problem instances that can be solved to optimality using off-the-shelf MILP solvers is encouraging, as MILP-based solution methods allow practitioners to conveniently tailor the models to their individual needs, e.g., to take into account a different objective function or additional business-specific constraints.

One direction for future research is analyzing the potential of adding novel valid inequalities to the sequence-based CT model to further enhance its computational performance. Specifically, improving the lower bounds, e.g., via energetic reasoning (cf., e.g., Haouari et al., 2012), could be a promising strategy. Furthermore, the considerable number of problem instances that can be solved to optimality and the good-quality feasible solutions the sequence-based CT model yields are a promising foundation for the development of matheuristics for the MRCPSP. We propose to integrate the sequence-based model in a matheuristic solution procedure, e.g., using the relax-optimize-and-fix strategy (cf. Bigler et al., 2024). Another promising research direction could be adapting the proposed models to other variants of the RCPSP (cf. Hartmann & Briskorn, 2022 for an overview), e.g., involving minimal and maximal time lags or resources with multiple skills. Also, adapting the models to address project scheduling problems under uncertainty, e.g., the robust MRCPSP (cf. Bold & Goerigk, 2022), could be a promising direction for future research.

Bibliography

Artigues, C. (2017). On the strength of time-indexed formulations for the resource-constrained project scheduling problem. *Operations Research Letters*, *45*, 154–159.

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

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

Bartels, J.-H., & Zimmermann, J. (2009). Scheduling tests in automotive R&D projects. *European Journal of Operational Research*, *193*, 805–819.

Bigler, T., Gnägi, M., & Trautmann, N. (2024). MIP-based solution approaches for multi-site resource-constrained project scheduling. *Annals of Operations Research*, *337*, 627–647.

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

Bold, M., & Goerigk, M. (2022). A faster exact method for solving the robust multi-mode resource-constrained project scheduling problem. *Operations Research Letters*, *50*, 581–587.

Bolusani, S., Besançon, M., Bestuzheva, K., Chmiela, A., Dionísio, J., Donkiewicz, T., van Doornmalen, J., Eifler, L., Ghannam, M., Gleixner, A., Graczyk, C., Halbig, K., Hedtke, I., Hoen, A., Hojny, C., van der Hulst, R., Kamp, D., Koch, T., Kofler, K., Lentz, J., Manns, J., Mexi, G., Mühmer, E., Pfetsch, M. E., Schlösser, F., Serrano, F., Shinano, Y., Turner, M., Vigerske, S., Weninger, D., & Xu, L. (2024). *The SCIP Optimization Suite 9.0*. ZIB-Report 24-02-29 Zuse Institute Berlin. URL: `https:`

//nbn-resolving.org/urn:nbn:de:0297-zib-95528.

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

Christofides, N., Alvarez-Valdes, R., & Tamarit, J. (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, *29*, 262–273.

Demeulemeester, E. L., & Herroelen, W. S. (2002). *Project Scheduling: A Research Handbook*. Boston: Kluwer Academic Publishers.

Fernandes, G. A., & de Souza, S. R. (2021). A matheuristic approach to the multi-mode resource constrained project scheduling problem. *Computers & Industrial Engineering*, *162*, 107592.

Gnägi, M., Rihm, T., Zimmermann, A., & Trautmann, N. (2019). Two continuous-time assignment-based models for the multi-mode resource-constrained project scheduling problem. *Computers & Industrial Engineering*, *129*, 346–353.

Haouari, M., Kooli, A., & Néron, E. (2012). Enhanced energetic reasoning-based lower bounds for the resource constrained project scheduling problem. *Computers & Operations Research*, *39*, 1187–1194.

Hartmann, S., & Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *297*, 1–14.

Klein, N., Gnägi, M., & Trautmann, N. (2024). Mixed-integer linear programming for project scheduling under various resource constraints. *European Journal of Operational Research*, *319*, 79–88.

Klein, R. (2000). *Scheduling of Resource-Constrained Projects*. Springer New York.

Koch, T., Berthold, T., Pedersen, J., & Vanaret, C. (2022). Progress in mathematical programming solvers from 2001 to 2020. *EURO Journal on Computational Optimization*, *10*, 100031.

Kolisch, R., & Drexl, A. (1997). Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, *29*, 987–999.

Kolisch, R., & Sprecher, A. (1996). PSPLIB — A project scheduling library. *European*

Journal of Operational Research, *96*, 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*, 3–13.

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.

Mika, M., Waligóra, G., & Węglarz, J. (2015). Overview and state of the art. In C. Schwindt, & J. Zimmermann (Eds.), *Handbook on Project Management and Scheduling Vol. 1* (pp. 445–490). Cham: Springer International Publishing.

Pfund, M. E., Mason, S. J., & Fowler, J. W. (2006). Semiconductor manufacturing scheduling and dispatching. In J. W. Herrmann (Ed.), *Handbook of Production Scheduling* (pp. 213–241). Boston, MA: Springer US.

Pinto, J., & Grossmann, I. (1995). A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants. *Industrial & Engineering Chemistry Research*, *34*, 3037–3051.

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

Sayah, D. (2023). Continuous-time formulations for multi-mode project scheduling. *Computers & Operations Research*, *152*, 106147.

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

Trautmann, N., Rihm, T., Saner, N., & Zimmermann, A. (2018). A continuous-time assignment-based MILP formulation for the resource-constrained project scheduling problem. In M. Caramia, L. Bianco, & S. Giordani (Eds.), *Proceedings of the 16th international conference on project management and scheduling* (pp. 242–245). URL: http://www.pms2018.ing.uniroma2.it/.

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

Xu, J., & Zeng, Z. (2015). Multi-criteria multi-modal fuzzy project scheduling in construction industry. In C. Schwindt, & J. Zimmermann (Eds.), *Handbook on Project Management and Scheduling Vol. 2* (pp. 1307–1335). Cham: Springer International Publishing.

Paper III

Flow shops with reentry: The total weighted completion time objective[3]

Maximilian von Aspern* Felix Buld* Nicklas Klein**[4] Michael Pinedo***

* Chair of Operations Research, Technical University of Munich
** Department of Business Administration, University of Bern
*** Leonard N. Stern School of Business, New York University

Contents

3.1 Introduction **74**
3.2 Problem description and notation **77**
3.3 Preliminaries and non-interruptive schedules **78**
3.4 Complexity and priority rules **82**
3.4.1 Complexity 83
3.4.2 Least Remaining Loops priority rule 84
3.4.3 Weighted Least Remaining Loops priority rule 87
3.5 Fixed-size flow shops **92**
3.5.1 Weak NP-hardness 93
3.5.2 A fully polynomial-time approximation scheme 96
3.6 Conclusion and outlook **98**
Bibliography **100**

[3]As of December 2024, this version of the paper was under review, submitted to Journal of Scheduling. The paper has since been published with minor revisions in Journal of Scheduling: `https://doi.org/10.1007/s10951-025-00847-y`.

[4]Nicklas Klein expresses his gratitude to the Hasler Foundation for supporting a research visit during which this paper was initiated.

Abstract

Flow shops are widely studied machine environments in which all jobs must visit all machines in the same order. While conventional flow shops assume that each job traverses the shop only once, many industrial environments require jobs to loop through the shop multiple times before completion. This means that after traversing the shop and completing its processing on the last machine, a job must return to the first machine and traverse the shop again until it has completed all its required loops. Such a setting, referred to as a flow shop with reentry, has numerous applications in industry, e.g., semiconductor manufacturing. The planning problem is to schedule all loops of all jobs while minimizing the total weighted completion time. In this paper, we consider reentrant flow shops with unit processing times. We show that this problem is strongly NP-hard if the number of machines is part of the input. We propose the Least Remaining Loops First (LRL) priority rule and show that it minimizes the total unweighted completion time. Then, we analyze the Weighted Least Remaining Loops First (WLRL) priority rule and show that it has a worst-case performance ratio of $(1+\sqrt{2})/2$ *(about 1.2). Additionally, we present a fully polynomial time approximation scheme (FPTAS) and a pseudo-polynomial time algorithm if the number of machines in the flow shop is fixed.*

3.1 Introduction

Flow shops are widely studied machine environments in which all jobs must visit all machines in the same order. In a conventional flow shop with m machines in series and n jobs, it is assumed that each job traverses the shop only once. However, many industrial environments require the jobs to undergo the same set of sequential processing steps multiple times. Such an environment is called a flow shop with reentry and can be described as follows. Each job must be processed first on machine 1, then on machine 2, and so on until it is processed on machine m. After completing its processing on machine m, the job must return to machine 1 and traverse the shop again until it has completed all its required loops. The planning problem is to schedule all loops of all jobs while adhering to precedence relations between consecutive loops of the same job. This means that a loop of a job must complete its processing on machine m before the next loop of that job can start its processing on machine 1.

In this paper, we consider the problems of minimizing the total completion time and total weighted completion time for reentrant flow shops. Minimizing these objective functions is of interest, e.g., if there is limited storage capacity or if jobs must undergo subsequent processing steps. It is well known that minimizing the total completion time for conventional flow shops with $m \geq 2$ machines is NP-hard. We show that for reentrant flow shops with $m \geq 2$ machines, minimizing the total weighted completion time is strongly NP-hard even if the jobs have unit processing times. Then, we analyze two priority rules. First, we analyze the "Least Remaining Loops First" (LRL) priority rule. This priority rule schedules, whenever machine 1 becomes available, a loop of a job with the least remaining loops among all available jobs. We show that this priority rule minimizes the total unweighted completion time as well as the total weighted completion time if the weights and numbers of loops are "agreeable", i.e., jobs with high weights also have small numbers of loops. Second, we analyze the "Weighted Least Remaining Loops First" (WLRL) priority rule. This priority rule schedules a loop of a job with the highest ratio of weight to the number of remaining loops among all available jobs whenever machine 1 becomes available. While it is not guaranteed that this priority rule generates optimal schedules, we show that it has a theoretical worst-case performance ratio of $\frac{1}{2}(1+\sqrt{2})$ (≈ 1.2). In practice, the WLRL rule seems to perform even better. In a numerical study, we observe that the average relative deviation of the total weighted completion time under WLRL from that under the optimal schedule is about 1%. Finally, we analyze reentrant flow shops with a fixed number of machines. We show that this problem is weakly NP-hard and propose a dynamic programming algorithm that solves the problem in pseudo-polynomial time. Additionally, we present a fully polynomial-time approximation scheme (FPTAS), which produces arbitrarily good approximate solutions.

Reentrant flow shops can be found in many manufacturing facilities across various industries. For example, the repeated dyeing and drying process in textile manufacturing and the coating and drying process in mirror manufacturing can be modeled as reentrant flow shops (Choi & Kim, 2007). Also, printed circuit board manufacturing lines require job recirculation (Pan & Chen, 2003). One of the most frequently discussed applications of job recirculation is in semiconductor wafer fabs (Pfund et al., 2006; Mönch et al., 2011). In these manufacturing facilities, wafers undergo a series of sequential processing steps that are repeated multiple times. Danping & Lee (2011) survey research methodologies for reentrant shops with a particular focus on semiconductor wafer manufacturing. Wu et al. (2022) model the so-called furnace (a part of the semiconductor manufacturing process) as a reentrant shop with additional constraints; they develop a scheduling heuristic and describe an industry implementation of their algorithm in a real-world wafer fab. In addition to manufacturing environments that involve planned repetitive processing, job recirculation may also occur in any production system if a job has to undergo repair or rework after a defect has been detected (Danping & Lee, 2011).

Flow shops with reentry were first examined by Graves et al. (1983). Since then, they have received considerable attention in the literature. Most publications focus on minimizing the makespan (Lev & Adiri, 1984; Wang et al., 1997; Pan & Chen, 2003; Chen, 2006; Choi & Kim, 2007, 2008; Shufan et al., 2023; Zheng et al., 2023). These papers discuss various formats of job reentry. For example, V-shops where the machine route is $1 \to 2 \to \ldots \to m-1 \to m \to m-1 \to \ldots \to 1$ (Lev & Adiri, 1984) or chain-reentrant shops in which only the first machine is revisited (Wang et al., 1997). For an overview of the various different formats of job reentry, see, e.g., Emmons & Vairaktarakis (2012). The reentrant flow shop with full loops is the most frequently examined machine route. In such an environment, each job must visit the machines in each loop in the order $1 \to 2 \to \ldots \to m$ (Pan & Chen, 2003; Chen, 2006; Choi & Kim, 2007, 2008). Yu & Pinedo (2020) investigate machine-ordered and proportionate flow shops with full loop reentry. In these settings, the processing times of jobs depend only on either the machine (machine-ordered) or the job (proportionate). The authors identify a class of schedules that minimize the makespan if all jobs have the same number of loops. Additionally, they propose the "Most Remaining Loops First" (MRL) priority rule and show that it minimizes the makespan under certain conditions, which include the case with unit processing times. The literature concerned with the total completion time objective is more scarce. Kubiak et al. (1996) examine the total completion time objective for a reentrant flow shop with a "hub" machine. In this setting, the jobs must alternate between the hub machine and the $m-1$ other machines, i.e., $1 \to 2 \to 1 \to 3 \to \ldots \to m \to 1$. Jing et al. (2011) propose a k-insertion heuristic for minimizing the total completion time in a reentrant flow shop with full loops. However, to the best of our knowledge, there is no analysis of optimal scheduling policies for the total completion time objective in the literature.

The remainder of this paper is organized as follows. Section 3.2 describes the planning problem in detail and provides an example. In Section 3.3, we introduce two concepts which we call progressions and non-interruptive schedules. We show that there is at least one optimal schedule that is non-interruptive. In Section 3.4, we show that minimizing the total weighted completion time is strongly NP-hard if the number of machines is part of the input. We analyze the LRL rule and show that it minimizes the total completion time. Then, we present the WLRL rule and show its performance guarantee. In Section 3.5, we show that the problem remains NP-hard for a fixed number of machines, and present the pseudo-polynomial time algorithm and the FPTAS. In Section 3.6, we conclude the paper and provide an outlook on future research.

3.2 Problem description and notation

Consider a flow shop with m machines in series and n jobs. Job j traverses the shop $\mathcal{L}_j$ times. Let p_{ijk} be the processing time of job j on machine i in loop k, $k = 1, \ldots, \mathcal{L}_j$ and let $w_j > 0$ denote the weight of job j. Let ℓ_{jk} refer to loop k of job j. For a schedule σ, let S_{jk} denote the start time of job j in loop k on machine 1 and C_{jk} the completion time on machine m. If the schedule is not clear from the context, we may instead write $S_{jk}(\sigma)$ and $C_{jk}(\sigma)$, respectively. Job j is completed once its last loop is completed, i.e., $C_j = C_{j\mathcal{L}_j}$. In what follows, we focus our analysis on permutation schedules, which are defined by considering each loop k of job j as a sub-job of job j (Yu & Pinedo, 2020). In total, $\sum_{j=1}^{n} \mathcal{L}_j$ loops must be sequenced while adhering to precedence constraints between successive loops of the same job. These precedence constraints take the form $S_{jk} \geq C_{j,k-1}$ for all jobs $j = 1, \ldots, n$ and all loops $k = 2, \ldots, \mathcal{L}_j$. This means the next loop of a job can start on machine 1 either immediately upon completion of the job's previous loop on machine m or at some later point in time. Our objective is to determine a schedule that minimizes the total weighted completion time $\sum_{j=1}^{n} w_j C_j$.

We assume throughout this paper that $p_{ijk} = p = 1$, motivated, e.g., by some underlying takt time in manufacturing. This implies that after completing its processing on machine 1, a job immediately continues its processing on machine 2, then on machine 3, and so on, until it completes its processing on machine m. It also implies that an optimal schedule does not allow for unforced idleness, i.e., in an optimal schedule, machine 1 is only left idle if no loop of any job is available to begin its processing on machine 1. We denote this scheduling problem as $F|reentry,\ p_{ijk} = 1|\sum w_j C_j$ using the standard three-field scheduling notation (Graham et al., 1979).

We illustrate the scheduling problem through the following example.

Example 3.2.1. *Consider a reentrant flow shop with $m = 3$ machines and $n = 5$ jobs. The jobs require $\mathcal{L} = (2, 2, 2, 3, 4)$ loops. The jobs have weights $w = (2, 1, 1, 3, 4)$. The Gantt chart for permutation sequence $\sigma = [\ell_{51}; \ell_{41}; \ell_{11}; \ell_{21}; \ell_{31}; \ell_{42}; \ell_{22}; \ell_{32}; \ell_{52}; \ell_{12}; \ell_{43}; \ell_{53}; \ell_{54}]$ is displayed in Figure 3.1. We observe that machine 1 is idle for two time units as loop ℓ_{54} can only start on machine 1 after loop ℓ_{53} is completed on machine 3. The total weighted completion time of this schedule is $\sum_{j=1}^{5} w_j C_j = 150$.*

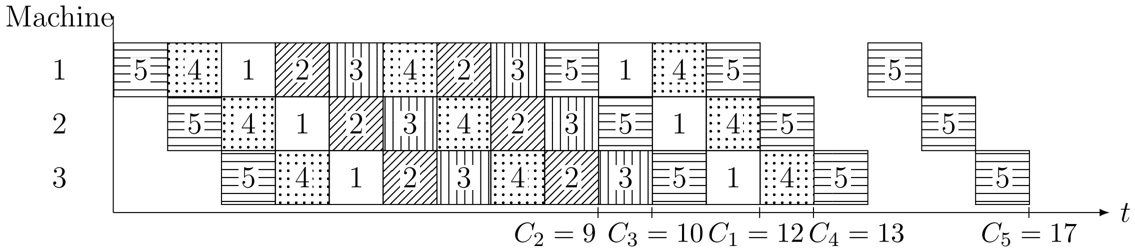


Figure 3.1: Gantt chart for the schedule of Example 3.2.1

3.3 Preliminaries and non-interruptive schedules

In this section, we introduce some preliminary definitions and lemmas. First, we introduce two concepts we refer to as progressions and progression interchanges. Utilizing these concepts, we introduce so-called non-interruptive schedules and show that there is a non-interruptive schedule that minimizes the total weighted completion time.

A progression can be described as an ordered set of loops of one or more jobs that are processed consecutively. This means that the next loop in the progression starts on machine 1 as soon as the previous loop completes on machine m. For a given schedule, a progression is initiated and uniquely defined by its first loop. A progression ends as soon as there is an idle time on machine 1 at the completion of the last loop in the progression.

Definition 3.3.1. *A progression initiated by loop ℓ_{jk} is defined as the following ordered set of loops:*

$$\mathcal{P}\left(\ell_{jk}\right) \coloneqq \{\ell_{jk}\} \cup \{\ell_{j'k'} : S_{j'k'} = C_{jk}\} \cup \{\ell_{j''k''} : S_{j''k''} = C_{j'k'}\} \cup \dots$$

Note that we could also characterize a progression via loops that have a difference in start times of m: $\mathcal{P}\left(\ell_{jk}\right) \coloneqq \{\ell_{j'k'} : S_{j'k'} = S_{jk} + rm, r \in \mathbb{N}_0\}$. If the loop that initiates the progression is not relevant or clear from the context, we simply denote the progression as $\mathcal{P}$. Consider the set of jobs that have their last loop in progression $\mathcal{P}$, denoted as $\{j : \ell_{j\mathcal{L}_j} \in \mathcal{P}\}$. We denote the total weight of these jobs as $\mathcal{W}(\mathcal{P}) \coloneqq \sum_{j:\ell_{j\mathcal{L}_j}\in\mathcal{P}} w_j$.

Note that the loops within one progression may belong to different jobs. Also, loops of the same job may be in different progressions. The following example demonstrates what the pattern of a progression may look like.

Example 3.3.2. *Consider the same schedule as in Example 3.2.1. The progression initiated by loop ℓ_{41} is $\mathcal{P}(\ell_{41}) = \{\ell_{41}, \ell_{31}, \ell_{32}, \ell_{43}\}$ with $\mathcal{W}\big(\mathcal{P}(\ell_{41})\big) = 4$. We display the progression in Figure 3.2.*

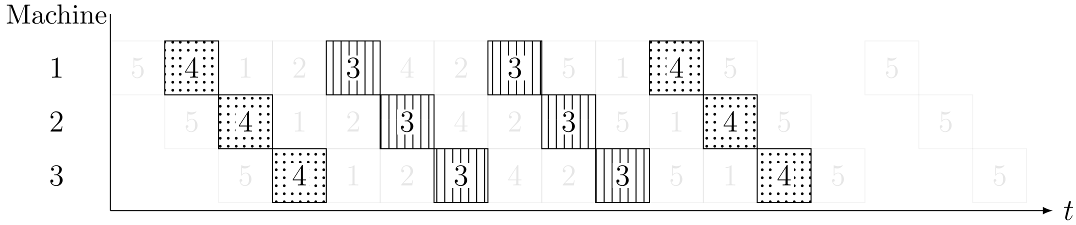


Figure 3.2: Gantt chart for progression $\mathcal{P}(\ell_{41})$

We now introduce progression interchanges. Informally, a progression interchange generates a new schedule in which all loops of one progression are interchanged with all loops of another progression. All loops that are not part of the interchange remain untouched.

Definition 3.3.3. *Consider two progressions $\mathcal{P}(\ell_{j_1k_1})$ and $\mathcal{P}(\ell_{j_2k_2})$ in the schedule σ with $S_{j_1k_1} < S_{j_2k_2}$ and $\ell_{j_2k_2} \notin \mathcal{P}(\ell_{j_1k_1})$. Denote the difference in start times as $\delta = S_{j_2k_2} - S_{j_1k_1}$. The new schedule σ' obtained through the progression interchange is defined by the following start times of loops:*

$$S_{jk}(\sigma') = \begin{cases} S_{jk} + \delta, & \text{if } \ell_{jk} \in \mathcal{P}(\ell_{j_1k_1}) \\ S_{jk} - \delta, & \text{if } \ell_{jk} \in \mathcal{P}(\ell_{j_2k_2}) \\ S_{jk}, & \text{otherwise.} \end{cases}$$

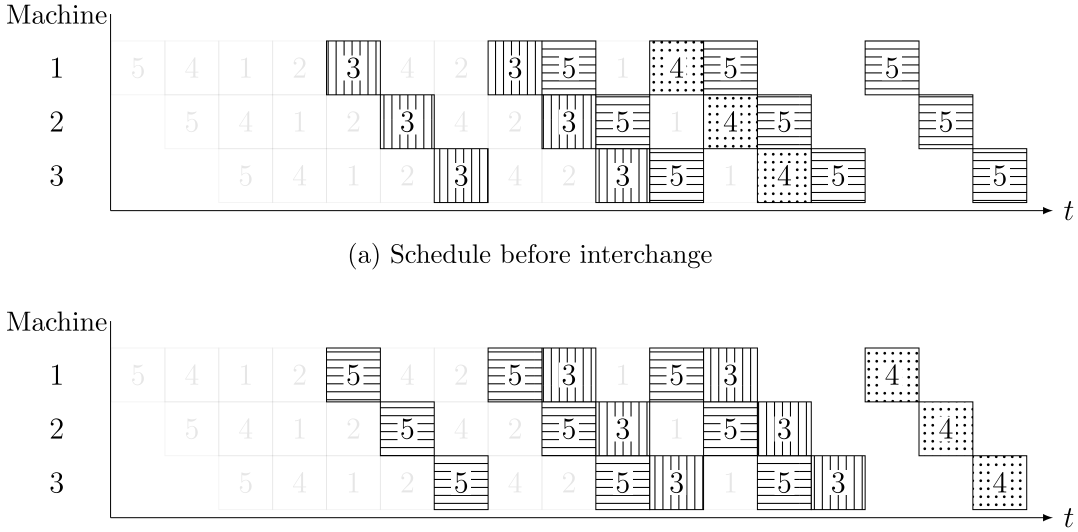
The total weighted completion time of schedule σ' is:

$$\sum_{j=1}^{n} w_j C_j(\sigma') = \sum_{j=1}^{n} w_j C_j(\sigma) + \delta\big[\mathcal{W}\big(\mathcal{P}(\ell_{j_1k_1})\big) - \mathcal{W}\big(\mathcal{P}(\ell_{j_2k_2})\big)\big].$$

We demonstrate what a progression interchange may look like in the following example.

Example 3.3.4. *Consider the same schedule as in Example 3.2.1. The new schedule obtained by interchanging progressions $\mathcal{P}(\ell_{31})$ and $\mathcal{P}(\ell_{52})$ is displayed in Figure 3.3. Note that the new schedule contains unforced idleness between loops ℓ_{32} and ℓ_{43}, since loop ℓ_{43} could actually start earlier on machine 1. However, as $w_3 + w_4 = w_5 = 4$, the total weighted completion time is unchanged.*

We now introduce so-called non-interruptive schedules. We say that a job is interrupted if there is a waiting time between the completion of a loop of that job on machine m and the start of the job's next loop on machine 1. In a non-interruptive schedule, no job has such an interruption.



(a) Schedule before interchange

(b) Schedule after interchange

Figure 3.3: Illustration of a progression interchange

Definition 3.3.5. *A schedule is called non-interruptive if the next loop of any job starts on machine* 1 *as soon as its previous loop completes on machine* m*, i.e.,*

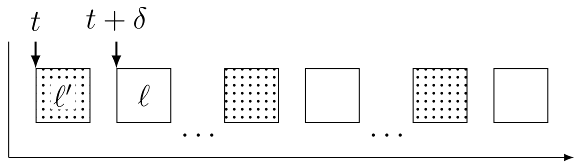
$$S_{jk} = C_{j,k-1}, \qquad \forall\ j = 1, \dots, n,\ k = 2, \dots, \mathcal{L}_j.$$

Note that in a non-interruptive schedule, all loops of a job are in the same progression. This means that a progression starting with a loop of a particular job will finish all loops of that job before it can schedule the first loop of a different job. This stands in contrast to arbitrary (interruptive) schedules, where a waiting time between two consecutive loops of the same job could occur.

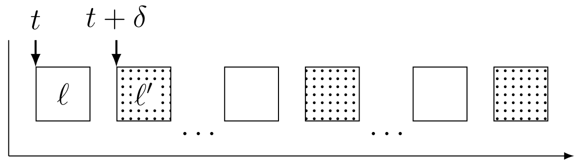
The following theorem states that among the schedules with minimal total weighted completion time, there is at least one that is non-interruptive. This structural result implies that we can focus our analysis of total weighted completion time optimal schedules on non-interruptive schedules. Note that this distinguishes the total weighted completion time objective from other regular objective functions, such as the makespan, for which instances can be created where optimal schedules must contain interruptions (Yu & Pinedo, 2020).

Theorem 3.3.6. *There is a non-interruptive schedule that minimizes the total weighted completion time.*

Proof. The proof is by contradiction. Suppose that no non-interruptive schedule has minimal total weighted completion time. Then, an optimal schedule must have at least



(a) Initial schedule



(b) New schedule

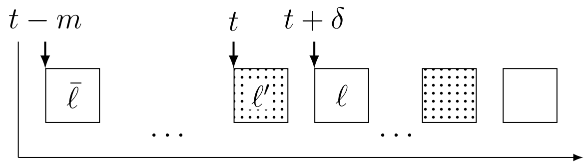
Figure 3.4: Demonstration of the progression interchange in case (i) in the proof of Theorem 3.3.6

one interruption. Consider an optimal schedule with the least interruptions among the optimal ones. We consider the last interruption, i.e., the last time t at which a loop $\bar{\ell} := \ell_{jk}$ with $k < \mathcal{L}_j$ completes on machine m, but loop $\ell' := \ell_{j'k'}$ with $j' \neq j$ starts on machine 1. As t is the last interruption, there are no interruptions from time $t+1$ onward. Let $\ell := \ell_{j,k+1}$ denote the next loop of job j that is scheduled on machine 1 at time $t+\delta > t$ and $\bar{\ell}' := \ell_{j',k'-1}$ the previous loop of job j' that is scheduled on machine 1 before time $t-m$. Let $\mathcal{P}' := \mathcal{P}(\ell')$ and $\mathcal{P} := \mathcal{P}(\ell)$ denote the two progressions initiated by loops ℓ' and ℓ, respectively. We consider three cases:

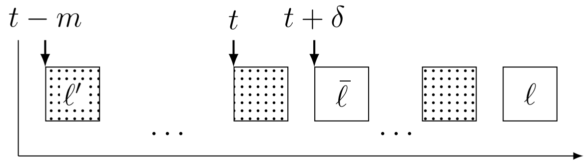
(i) $\boldsymbol{\mathcal{W}(\mathcal{P}') \leq \mathcal{W}(\mathcal{P})}$. We create a schedule without interruption at time t by interchanging the disjoint progressions $\mathcal{P}'$ and $\mathcal{P}$, see Figure 3.4. The change in the objective equals $\delta \cdot \big[\mathcal{W}(\mathcal{P}') - \mathcal{W}(\mathcal{P})\big]$. As $\mathcal{W}(\mathcal{P}') \leq \mathcal{W}(\mathcal{P})$, the total weighted completion time of this new schedule is either less than or equal to the original schedule. The new schedule also has one interruption less than the original schedule. This contradicts our assumptions.

(ii) $\boldsymbol{\mathcal{W}(\mathcal{P}') > \mathcal{W}(\mathcal{P})}$ **and** $\boldsymbol{C_{\bar{\ell}'} \leq t-m}$. We show that this is a contradiction to the original schedule being optimal. To see this, consider a new schedule that adds loop $\bar{\ell}$ as the first element to $\mathcal{P}$ and starts $\mathcal{P}'$ at time $t-m$, which corresponds to the start time of loop $\bar{\ell}$ in the old schedule, see Figure 3.5. As $C_{\bar{\ell}'} \leq t-m$, this new schedule is feasible. The change in the objective equals $m \cdot \big[\mathcal{W}(\mathcal{P}) - \mathcal{W}(\mathcal{P}')\big]$. Thus, the new schedule has a lower total weighted completion time. This contradicts our assumptions.

(iii) $\boldsymbol{\mathcal{W}(\mathcal{P}') > \mathcal{W}(\mathcal{P})}$ **and** $\boldsymbol{C_{\bar{\ell}'} > t-m}$. Similar to case (ii), we show that this is a contradiction to the original schedule being optimal. As the previous loop of job j', $\bar{\ell}'$, is not yet completed at time $t-m$, the new schedule we obtained in case (ii) is infeasible. Consider the $m-1$ loops scheduled on machine 1 from $t-m+1$ to $t-1$. Denote these



(a) Initial schedule



(b) New schedule

Figure 3.5: Demonstration of the progression interchange in case (ii) in the proof of Theorem 3.3.6

loops as $\tilde{\ell}_{m-1}, \ldots, \tilde{\ell}_1$ and their corresponding progressions as $\tilde{\mathcal{P}}_{m-1}, \ldots, \tilde{\mathcal{P}}_1$. Assume $\bar{\ell}'$ completes on machine m at time $t-q$. This implies that loop ℓ' can start processing on machine 1 at time $t-q$ or later. It follows that $\mathcal{W}(\tilde{\mathcal{P}}_q) \geq \mathcal{W}(\mathcal{P}')$ must hold; otherwise, an interchange of $\tilde{\mathcal{P}}_q$ and $\mathcal{P}'$ in the original schedule would have resulted in a feasible schedule with a lower total weighted completion time, which contradicts our assumptions.

Consider now a new schedule that adds loop $\bar{\ell}$ as the first element to $\mathcal{P}$, starts $\mathcal{P}'$ at time $t-q$ and $\tilde{\mathcal{P}}_q$ at time $t-m$, see Figure 3.6. As $\mathcal{W}(\tilde{\mathcal{P}}_q) \geq \mathcal{W}(\mathcal{P}') > \mathcal{W}(\mathcal{P})$ holds, this schedule has a lower total weighted completion time than the original schedule. If this schedule is feasible, this contradicts our assumptions. If this schedule is infeasible, one can apply a similar interchange involving $\tilde{\mathcal{P}}_q$ and one of the progressions $\tilde{\mathcal{P}}_{m-1}, \ldots, \tilde{\mathcal{P}}_{q-1}$. Iteratively, this results in a feasible schedule with a lower total weighted completion time, which contradicts our assumptions.

□

3.4 Complexity and priority rules

In this section, we analyze the complexity of $(F|reentry,\ p_{ijk}=1|\sum w_j C_j)$ and present two priority rules. In Subsection 3.4.1, we show that the problem is strongly NP-hard by a reduction from 3-PARTITION. In Subsection 3.4.2, we present the LRL priority rule and show that it minimizes the total completion time as well as the total weighted completion time if the weights are agreeable. In Subsection 3.4.3, we present the WLRL priority rule and show its worst-case performance ratio.

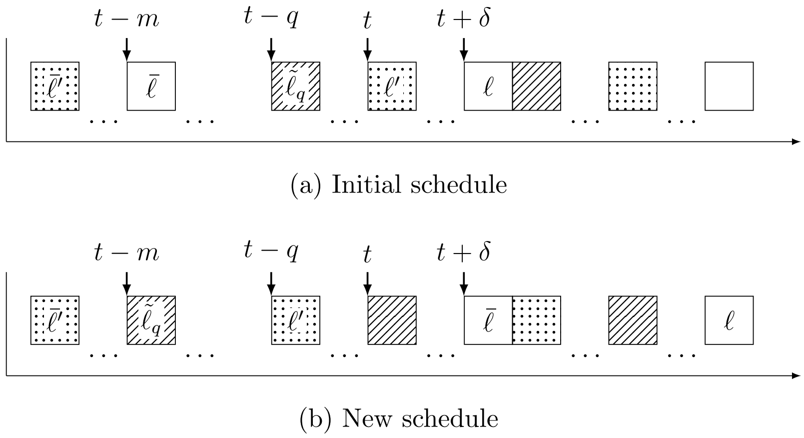


Figure 3.6: Demonstration of the progression interchange in case (iii) in the proof of Theorem 3.3.6

3.4.1 Complexity

The decision problem 3-PARTITION is a well-known strongly NP-hard decision problem.

Definition 3.4.1. *The decision problem 3-PARTITION is defined as follows. Given integers* $a_1, \dots, a_{3q}, b, q$, *such that* $\frac{b}{4} < a_j < \frac{b}{2}$ *and* $qb = \sum_{j=1}^{3q} a_j$, *is there a partition of the first* $3q$ *integers into* q *disjoint triplets* $\mathcal{S}_1, \dots, \mathcal{S}_q$ *with* $\sum_{a_j \in S_t} a_j = b$ *for all* $t = 1, \dots, q$*?*

Before we present the reduction, we provide another formulation of the objective function. We denote the progression that starts at times $t = 0, 1, \dots, m-1$ by $\mathcal{P}_t$. Next, we rewrite the objective for the case $w_j = \mathcal{L}_j$ for all jobs j. The result is similar to (Eastman et al., 1964, page 270) for a parallel machine setting.

Lemma 3.4.2. *Given* m *machines and weights* $w_j = \mathcal{L}_j$, *we obtain for any non-interruptive schedule without intermediate idle times:*

$$\sum_{j=1}^{n} \mathcal{L}_j C_j = \frac{m}{2} \cdot \sum_{t=1}^{m} \left(|\mathcal{P}_t|^2 + |\mathcal{P}_t| \frac{2 \cdot (t-1)}{m} \right) + \frac{m}{2} \cdot \sum_{j=1}^{n} \mathcal{L}_j^2.$$

Proof. Let $1, \dots, n_t$ be the jobs assigned to progression $\mathcal{P}_t$. The objective for these jobs can be rewritten:

$$\sum_{j=1}^{n_t} \mathcal{L}_j C_j = \sum_{j=1}^{n_t} \mathcal{L}_j \left(t - 1 + \sum_{k=1}^{j} \mathcal{L}_k \cdot m \right) = \frac{m}{2} \cdot \left(|\mathcal{P}_t|^2 + \sum_{j=1}^{n_t} \mathcal{L}_j^2 \right) + |\mathcal{P}_t| \cdot (t-1) \,.$$

Summing the terms for all progressions $\mathcal{P}_1, \dots, \mathcal{P}_t$ yields the claimed formula. □

The above calculation also shows that the objective is independent of the ordering of jobs within a progression if $w_j = \mathcal{L}_j$ holds for all jobs. Using Lemma 3.4.2, it can be shown that it is optimal to balance the number of loops in the progressions as much as possible.

Theorem 3.4.3. *The scheduling problem* $F|reentry,\ p_{ijk} = 1|\sum w_j C_j$ *is strongly NP-hard.*

Proof. Let $(b, q, a_1, \ldots, a_{3q})$ be an instance of 3-PARTITION. For each item a_k, $k = 1, \ldots, 3q$, we define a job k with $w_k = \mathcal{L}_k = a_k$. Further, we choose the number of machines to be equal to the number of triplets, i.e., $m = q$. By Lemma 3.4.2, the cost of a non-interruptive schedule without intermediate idle times on this instance is given as $\frac{m}{2}\left(\sum_{t=1}^{m}\left(|\mathcal{P}_t|^2 + |\mathcal{P}_t|\frac{2\cdot(t-1)}{m}\right) + \sum_{k=1}^{3q}\mathcal{L}_k^2\right)$. Notice that the last term does not depend on the assignment of jobs to progressions. We show that this scheduling instance has an objective value of at most $\frac{m}{2}\left(mb^2 + (m-1)b + \sum_{k=1}^{3q}\mathcal{L}_k^2\right)$ if and only if $(b, q, a_1, \ldots, a_{3q})$ is a YES-instance of 3-PARTITION.

Let the instance be a YES-instance of 3-PARTITION and the jobs assigned into triplets $S_1, S_2, \ldots, S_q$ such that $\sum_{a_j \in S_t} a_j = b$ for all $t = 1, \ldots, q$. We can assign the jobs from S_t to progression $\mathcal{P}_t$ to get the above objective value.

Reversely, assume there exists a schedule of value at most the above value. If $|\mathcal{P}_1| = |\mathcal{P}_2| = \ldots = |\mathcal{P}_m|$, then we are done. Otherwise, two progressions $\mathcal{P}_i, \mathcal{P}_l$ must exist such that $|\mathcal{P}_i| \geq |\mathcal{P}_l|+2$. Since the numbers of loops in the progressions are not perfectly balanced, we can use $\sum_{t=1}^{m}|\mathcal{P}_t| = q \cdot b$ to show $\sum_{t=1}^{m}\left(|\mathcal{P}_t|^2 + |\mathcal{P}_t|\frac{2\cdot(t-1)}{m}\right) > mb^2 + (m-1)b$. This result contradicts our assumption, which concludes the proof. □

3.4.2 Least Remaining Loops priority rule

In this subsection, we introduce the "Least Remaining Loops first" (LRL) priority rule. We show that this rule minimizes the total (unweighted) completion time, i.e., $w_j = 1$ for all jobs $j = 1, \ldots, n$. Also, we show that it minimizes the total weighted completion time if the weights are agreeable.

Definition 3.4.4. *The LRL priority rule selects, whenever machine* 1 *becomes available, the next loop of a job with the least remaining loops among all available jobs.*

The following example displays an LRL schedule.

Example 3.4.5. *Consider the same instance as in Example 3.2.1 with* $n = 5$ *and* $w_j = 1$ *for all* $j = 1, \ldots, 5$. *The LRL rule generates the permutation sequence* $\sigma = [\ell_{11}; \ell_{21}; \ell_{31}; \ell_{12}; \ell_{22}; \ell_{32}; \ell_{41}; \ell_{51}; \ell_{42}; \ell_{52}; \ell_{43}; \ell_{53}; \ell_{54}]$. *The Gantt chart for this schedule is shown in Figure 3.7. The total completion time of this schedule is* $C_1 + C_2 + C_3 + C_4 + C_5 = 55$.

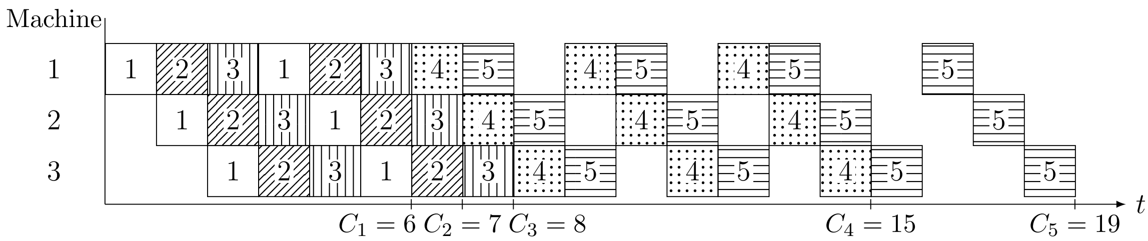


Figure 3.7: Schedule generated via priority rule LRL

After a loop of a job has been scheduled, that job will have one less loop remaining the next time it is available to be processed on machine 1. Thus, the job's priority is even higher than before. This implies that the next loop of that job will be scheduled immediately without interruption.

Lemma 3.4.6. *Schedules generated via the LRL rule are non-interruptive.*

Proof. If a loop of a job is scheduled at time t, it has priority over all jobs that are available at time t. The next time a loop of this job is available, it has one less loop than before. One can inductively show that this job is the unique one with the lowest number of remaining loops left among all jobs available at time $t + m$. Therefore, it again has priority at $t + m$ according to the LRL rule. $\square$

We now show that LRL minimizes the total completion time. The proof uses a similar interchange argument as in Theorem 3.3.6. However, it focuses on the first interruption rather than the last.

Theorem 3.4.7. *LRL minimizes the total completion time.*

Proof. The proof is by contradiction. Suppose LRL does not minimize the total completion time. This means that every optimal schedule deviates from LRL at least once. According to Theorem 3.3.6, we can assume that this optimal schedule has no interruptions. Let time t be the first time the optimal schedule does not act according to LRL with loop $\ell' := \ell_{j'k'}$ being scheduled while loop $\ell := \ell_{jk}$ would have had priority under LRL. Since the schedule has no interruptions, both ℓ and ℓ' must be the first loop of their respective jobs j and j'. Define $\mathcal{P}' := \mathcal{P}(\ell')$ and $\mathcal{P} := \mathcal{P}(\ell)$. Note that $\mathcal{W}(\mathcal{P})$ is the number of jobs in this progression, since $w_j = 1$ for all jobs j. We distinguish two cases:

(i) $\mathcal{W}(\mathcal{P}') \leq \mathcal{W}(\mathcal{P})$. We create a schedule that acts according to LRL at time t by interchanging the disjoint progressions $\mathcal{P}'$ and $\mathcal{P}$. As $\mathcal{W}(\mathcal{P}') \leq \mathcal{W}(\mathcal{P})$, the total weighted completion time of this new schedule is either less than or equal to the original schedule. This contradicts our assumptions.

(ii) $\mathcal{W}(\mathcal{P}') > \mathcal{W}(\mathcal{P})$. We create a schedule that acts according to LRL at time t by interchanging all loops of jobs j' and j. As $\mathcal{L}_j < \mathcal{L}_{j'}$, all subsequent jobs in $\mathcal{P}'$ have an earlier completion time, while all subsequent jobs in $\mathcal{P}$ have a later completion time. As $\mathcal{W}(\mathcal{P}') > \mathcal{W}(\mathcal{P})$, the total completion time of the new schedule is lower. This contradicts our assumptions. Note that this interchange also works if the progressions are not disjoint, i.e., $\mathcal{P} \subset \mathcal{P}'$.

□

The previous theorem implies that any non-interruptive schedule minimizes the total completion time in case all jobs have the same number of loops, i.e., $\mathcal{L}_j = \mathcal{L}$ for all $j = 1, \ldots, n$. The class of non-interruptive schedules can be seen as a "counterpart" to so-called "Loopwise-Cyclic" schedules that minimize the makespan for such instances (Yu & Pinedo, 2020).

We now consider instances in which the weights may differ but are "agreeable" with the number of loops of the job.

Definition 3.4.8. *The weights and the numbers of loops are "agreeable" if $\mathcal{L}_j \leq \mathcal{L}_{j'} \Leftrightarrow w_j \geq w_{j'}$ holds for any pair of jobs j and j', and at least one of the inequalities is strict.*

The next lemma shows that if the weights and number of loops are agreeable, LRL minimizes the total weighted completion time.

Lemma 3.4.9. *LRL minimizes the total weighted completion time if the weights and number of loops are agreeable and ties are broken by the highest weight.*

Proof. The proof is similar to the proof of Theorem 3.4.7. Let time t be the first time the optimal schedule does not act according to LRL with loop $\ell' := \ell_{j'k'}$ being scheduled while loop $\ell := \ell_{jk}$ would have had priority under LRL. Consider the same two cases:

(i) $\mathcal{W}(\mathcal{P}') \leq \mathcal{W}(\mathcal{P})$. Interchange the disjoint progressions $\mathcal{P}'$ and $\mathcal{P}$ (see proof of Theorem 3.4.7).

(ii) $\mathcal{W}(\mathcal{P}') > \mathcal{W}(\mathcal{P})$. Interchange all loops of jobs j' and j. As $\mathcal{L}_j \leq \mathcal{L}_{j'}$, all subsequent jobs in $\mathcal{P}'$ have a completion time that is not later than in the original schedule, while all subsequent jobs in $\mathcal{P}$ have a completion time that is not earlier. As $\mathcal{W}(\mathcal{P}') > \mathcal{W}(\mathcal{P})$ and $w_j \geq w_{j'}$, the total weighted completion time of the new schedule cannot be higher. As one of the inequalities $\mathcal{L}_j \leq \mathcal{L}_{j'}$ or $w_j \geq w_{j'}$ is strict, the total weighted completion time is strictly lower, which contradicts our assumptions. □

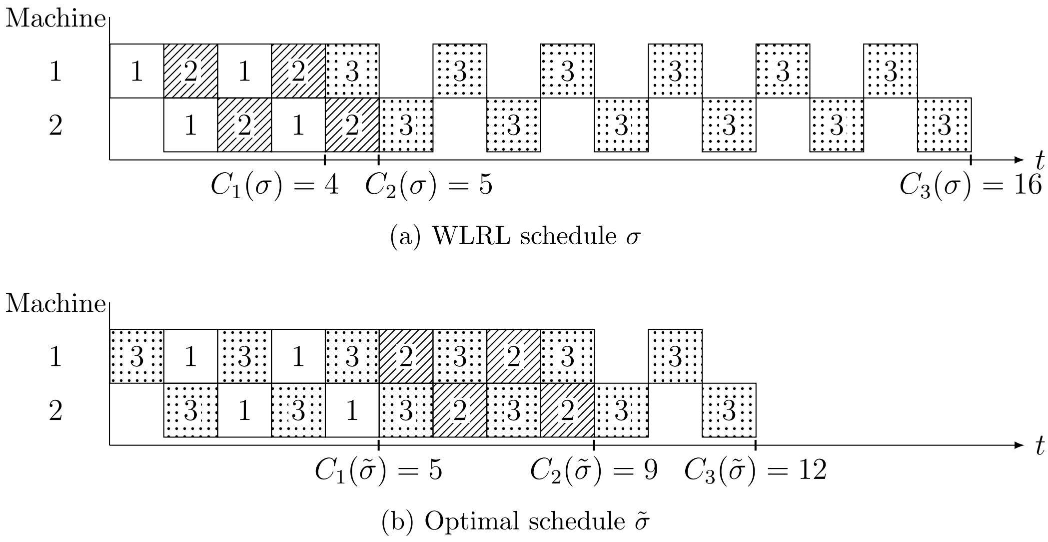


Figure 3.8: Gantt charts for the WLRL schedule and an optimal schedule of Example 3.4.11

3.4.3 Weighted Least Remaining Loops priority rule

In this subsection, we introduce the "Weighted Least Remaining Loops first" (WLRL) rule. We show that it produces schedules whose total weighted completion time exceeds the optimal total weighted completion by a factor of at most $\frac{1}{2}\bigl(1+\sqrt{2}\bigr) \approx 1.207$.

Definition 3.4.10. *The WLRL priority rule selects, whenever machine* 1 *becomes available, the next loop of a job with the highest* $\frac{w_j}{\mathcal{L}_j}$ *ratio among all available jobs.*

The following example displays a WLRL schedule and shows that the rule may not generate an optimal solution.

Example 3.4.11. *Consider a reentrant flow shop with* $m = 2$ *machines and* $n = 3$ *jobs. The jobs require* $\mathcal{L}_1 = 2,\ \mathcal{L}_2 = 2,$ *and* $\mathcal{L}_3 = 6$ *loops, and have weights* $w_1 = 2.2,\ w_2 = 2.1,$ *and* $w_3 = 6$*. The WLRL rule generates the permutation sequence* $\sigma = [\ell_{11}; \ell_{21}; \ell_{12}; \ell_{22}; \ell_{31}; \ell_{32}; \ell_{33}; \ell_{34}; \ell_{35}; \ell_{36}]$*. The Gantt chart for this schedule is displayed in Figure 3.9a. The total weighted completion time of this schedule is* 115.3*.*

The optimal schedule $\tilde{\sigma}$ *for this instance is depicted in Figure 3.9b and has a total weighted completion time of* 101.9*.*

Although it is not guaranteed that the WLRL rule generates an optimal schedule, it seems to work reasonably well in practice. One can measure the performance of an algorithm by analyzing the factor by which the algorithm's solution is larger than the optimal solution. This factor is also called "approximation ratio" or "performance ratio".

Let $\phi(ALG)$ and $\phi(OPT)$ describe the objective function values of the algorithm's solution and the optimal solution, respectively. The algorithm's performance ratio is $\frac{\phi(ALG)}{\phi(OPT)}$. In a numerical experiment involving 20,000 randomly generated instances with 4 to 8 jobs, 2 to 6 machines, 1 to 20 loops per job, and weights ranging from 1 to 20 (all sampled uniformly random), the average performance ratio of the WLRL rule was 1.01, while the empirical worst-case performance ratio was 1.11.

Our goal in what follows is to derive and prove a theoretical worst-case bound on the performance ratio of the WLRL rule. For the parallel machine scheduling problem $P||\sum w_jC_j$, Kawaguchi & Kyan (1986) showed that the "Weighted Shortest Processing Time First" (WSPT) rule, which schedules the jobs following the $\frac{w_j}{p_j}$ priority rule (see, e.g., Pinedo, 2022), has a worst-case performance ratio of $\frac{1}{2}\big(1+\sqrt{2}\big)$. Schwiegelshohn (2011) presented an alternative proof of this bound by analyzing the structure of worst-case instances. We show that for reentrant flow shops with unit processing times, the WLRL rule also has a worst-case performance ratio of $\frac{1}{2}\big(1+\sqrt{2}\big)$. We achieve our result in three steps. First, we identify properties of a worst-case instance, similar to Schwiegelshohn (2011). For instances that satisfy these properties, we then create related instances of the $P||\sum w_jC_j$ problem in the second step. Finally, we use these instances to conclude our proof.

Throughout this section, we denote an instance of the reentrant flow shop problem as I. The WLRL schedule and optimal schedule are denoted as $WLRL(I)$ and $OPT(I)$, respectively, or $WLRL$ and OPT if the instance is clear from the context. Furthermore, we denote the performance ratio of the WLRL rule as:

$$\rho(I) = \frac{\phi\big(WLRL(I)\big)}{\phi\big(OPT(I)\big)} = \frac{\sum w_jC_j\big(WLRL(I)\big)}{\sum w_jC_j\big(OPT(I)\big)}.$$

The first step of the proof is to identify two properties that potential worst-case instances must satisfy. The following lemma shows that in a worst-case instance, the weight of each job is equal to its number of loops.

Lemma 3.4.12. *For every instance I, there is an instance I' with $\rho(I) \leq \rho(I')$ and $w_j = \mathcal{L}_j$ for all jobs j.*

Proof. Replace p_j by $\mathcal{L}_j$ in Corollary 1 of Schwiegelshohn (2011). □

If $w_j = \mathcal{L}_j$ for all jobs j, any list schedule can be generated according to the WLRL rule by using appropriate tie-breakers. In fact, we can create any list schedule via the WLRL rule by adding or subtracting arbitrarily small δ values to the weights. The influence of such perturbations on the performance ratio is negligible. Therefore, we can focus our analysis on a worst-case and an optimal order.

The following lemma shows that a potential worst-case instance has some "small" jobs with one loop and at most $m-1$ "big" jobs with two or more loops. Let $T_{idl}(\sigma)$ denote the first idle time on machine 1 under schedule σ. The idea is to create another instance with a higher performance ratio that splits off one loop of a job that currently completes earlier than the first idle time on machine 1 in the worst-case schedule.

Lemma 3.4.13. *For every instance I, there is an instance I' with $\rho(I) \leq \rho(I')$, $w_j = \mathcal{L}_j$ for all jobs j, and $\mathcal{L}_j = 1$ if $S_j < T_{idl}\big(WLRL(I')\big) - m$.*

Proof. By Lemma 3.4.12, we can assume that $w_j = \mathcal{L}_j$ for all jobs j in I. Let $\sigma(I)$ be an arbitrary schedule and let j_0 be a job from instance I with $\mathcal{L}_{j_0} > 1$ starting before time $T_{idl}\big(WLRL(I)\big) - m$ in $\sigma(I)$. We split this job into two smaller jobs j_1 and j_2 with $\mathcal{L}_{j_1} = w_{j_1} = 1$ and $\mathcal{L}_{j_2} = w_{j_2} = \mathcal{L}_{j_0} - 1$ to produce a new instance I'. We extend $\sigma(I)$ to $\sigma(I')$ by scheduling $S_{j_1}\big(\sigma(I')\big) = S_{j_0}\big(\sigma(I)\big)$, $C_{j_1}\big(\sigma(I')\big) = S_{j_2}\big(\sigma(I')\big) = S_{j_0} + m$, which implies $C_{j_2}\big(\sigma(I')\big) = C_{j_0}\big(\sigma(I)\big)$. This operation clearly does not affect the completion time of any other job and, by the assumption $S_{j_0} + m < T_{idl}\big(WLRL(I')\big)$, also results in a feasible WLRL schedule.

Such a split (strictly) decreases the total weighted completion time by:

$$\mathcal{L}_{j_0} C_{j_0} - (S_{j_0} + m) - (\mathcal{L}_{j_0} - 1) C_{j_0} = C_{j_0} - S_{j_0} - m = m\left(\mathcal{L}_{j_0} - 1\right) > 0.$$

It follows that:

$$1 \leq \rho(I) \leq \frac{\sum w_j C_j\big(WLRL(I)\big) - m\left(\mathcal{L}_{j_0} - 1\right)}{\sum w_j C_j\big(OPT(I)\big) - m\left(\mathcal{L}_{j_0} - 1\right)} \leq \frac{\sum w_j C_j\big(WLRL(I')\big)}{\sum w_j C_j\big(OPT(I')\big)} = \rho(I').$$

Iteratively, splitting jobs yields an instance I' with the claimed properties. □

Using this lemma, we can focus on instances that contain x small jobs with one loop and $B < m$ big jobs with $2 \leq y_0 \leq \ldots \leq y_{B-1}$ loops, all satisfying $w_j = \mathcal{L}_j$. One can show, using an interchange argument, that an optimal order for such an instance is generated by breaking ties according to the highest weight. That is, starting the job with the most loops at time 0, the job with the second most loops at time 1, and so on. Conversely, one can show that a worst-case order can be generated by breaking ties according to the smallest weight. That is, starting the x small jobs at times $0, 1, \ldots, x-1$ and the big jobs at times $x, x+1, \ldots, x+B-1$.

The second step of the proof is to transform a potential worst-case instance I into a corresponding instance $\bar{I}$ of the $P||\sum w_j C_j$ problem. We create such an instance that

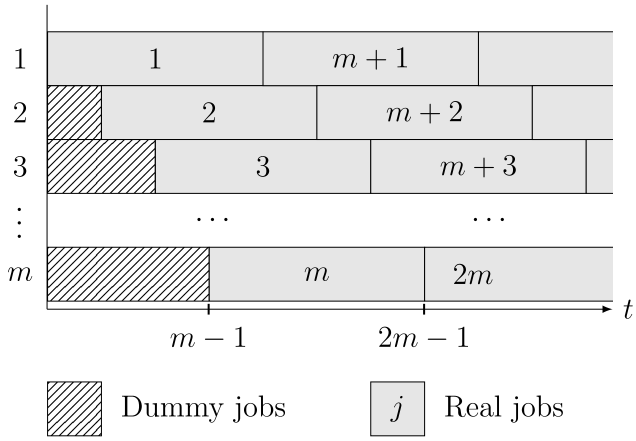


Figure 3.9: Schematic Gantt chart for the worst-case WSPT schedule in the proof of Lemma 3.4.15

contains all jobs of instance I ("real" jobs) plus some "dummy" jobs. Then, we split a subset of the real jobs of this instance to offset the total weighted completion time of the dummy jobs.

Definition 3.4.14. *For an instance I as described above, we create an instance $\bar{I}$ of the $P||\sum w_jC_j$ problem as follows: $\bar{p}_j := m\mathcal{L}_j,\ \bar{w}_j := m\mathcal{L}_j,\ j = 1, \dots, x + B$. Additionally, we define $m - 1$ dummy jobs with $\bar{p}_{x+B+i} := i,\ \bar{w}_{x+B+i} := i,\ i = 1, \dots, m - 1$.*

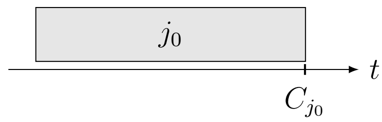
As $\bar{w}_j = \bar{p}_j$ holds for all jobs in instance $\bar{I}$, any list schedule can be seen as a WSPT schedule; we can achieve a unique WSPT order by modifying the weights by very small values δ.

Next, we show that for a convenient ordering, the "real" jobs of instance $\bar{I}$ have the same completion times as in a worst-case schedule of instance I.

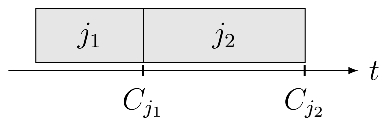
Lemma 3.4.15. *There is a WSPT ordering of the jobs of instance $\bar{I}$ such that the completion times of the real jobs are equal to the completion times of a worst-case WLRL schedule of instance I.*

Proof. We schedule the jobs as follows: schedule real job 1 at time 0 on machine 1. Schedule the dummy jobs such that dummy job $x+B+i-1$ with length $i-1$ is scheduled on machine i, $i = 2, \dots, m$. Schedule real job 2 at time 1 on machine 2. Schedule real job 3 at time 2 on machine 3 and so on, see Figure 3.9. One can easily verify that the completion times of this schedule are equal to those of the worst-case schedule of instance I. □

Corollary 3.4.16. *There is a WSPT ordering of the jobs of instance $\bar{I}$ such that the completion times of the real jobs are equal to those of an optimal schedule of instance I.*



(a) Original job



(b) Split job

Figure 3.10: Schematic display of job splitting

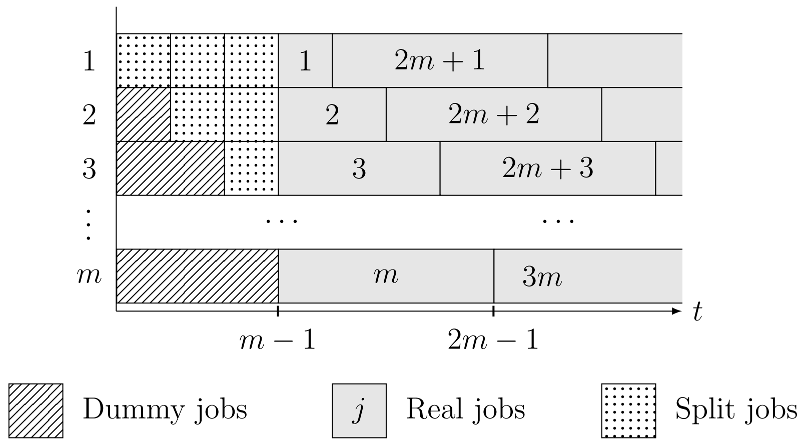


Figure 3.11: Schematic Gantt chart for the worst-case WSPT schedule after job splitting

Proof. Reverse the order of the real jobs in the proof of Lemma 3.4.15. $\square$

In what follows, we aim to offset the weighted completion time of the dummy jobs by splitting some real jobs. One can easily verify that for an ordering as described in Lemma 3.4.15, the weighted completion time of the dummy jobs is $\Delta := \sum_{j=x+B+1}^{x+B+m-1} \bar{w}_j C_j\big(WSPT(\bar{I})\big) = \sum_{i=1}^{m-1} i^2 = \frac{m(m-1)(2m-1)}{6}$. To offset this increase in the total weighted completion time, we can split up some real jobs with processing times m. Specifically, we split a job j_0 with $\bar{w}_{j_0} = \bar{p}_{j_0}$ into two jobs j_1 and j_2 with $\bar{w}_{j_1} = \bar{p}_{j_1}$, $\bar{w}_{j_2} = \bar{p}_{j_2}$ and $\bar{p}_{j_1} + \bar{p}_{j_2} = \bar{p}_{j_0}$, see Figure 3.10. Such a split reduces the total weighted completion time of a schedule by $\bar{p}_{j_1}\bar{p}_{j_2}$ (Schwiegelshohn, 2011).

Similarly to Lemma 3.4.13, this procedure can be repeated until all "split" jobs starting before time $m-1$ have a processing time of 1. Such an instance is illustrated in Figure 3.11. One can easily verify that these splits result in a reduction in the total weighted completion time of $\sum_{i=1}^{m-1} i^2 = \Delta$. We denote the instance in which the jobs have been split up as $\bar{I}'$.

The third step of the proof is to show the worst-case performance ratio of WLRL using instances $\bar{I}, \bar{I}'$, and the worst-case performance ratio $\bar{\rho}$ of the WSPT rule.

Theorem 3.4.17. *The WLRL rule has a worst-case performance ratio of* $\frac{1}{2}\big(1+\sqrt{2}\big)$.

Proof. Consider a potential worst-case instance $I = (x, y_0, \ldots, y_{B-1})$ and the corresponding instances $\bar{I}$ and $\bar{I}'$. Let $C_j\big(WSPT(\bar{I})\big)$ denote the completion times achieved when using the ordering of Lemma 3.4.15 and let $C_j\big(OPT(\bar{I})\big)$ denote the completion times achieved when using the ordering of Corollary 3.4.16. Note that the true optimal solution to $\bar{I}$ may have an even lower total weighted completion time. The following inequalities hold:

$$
\begin{aligned}
\frac{1}{2}\big(1+\sqrt{2}\big) \geq \bar{\rho}(\bar{I}') \geq & \frac{\sum_{j=1}^{x+B+m-1} \bar{w}_j C_j\big(WSPT(\bar{I})\big) - \Delta}{\sum_{j=1}^{x+B+m-1} \bar{w}_j C_j\big(OPT(\bar{I})\big) - \Delta} \\
= & \frac{\sum_{j=1}^{x+B} \bar{w}_j C_j\big(WSPT(\bar{I})\big) + \Delta - \Delta}{\sum_{j=1}^{x+B} \bar{w}_j C_j\big(OPT(\bar{I})\big) + \Delta - \Delta} \\
\geq & \frac{\sum_{j=1}^{x+B} m w_j C_j\big(WLRL(I)\big)}{\sum_{j=1}^{x+B} m w_j C_j\big(OPT(I)\big)} = \rho(I).
\end{aligned}
$$

□

Finally, we note that our worst-case performance ratio is tight. This means that there is no value $\alpha < \frac{1}{2}\big(1+\sqrt{2}\big)$ such that for every instance I, $\rho(I) \leq \alpha$. This can be seen by analyzing instances with a structure of x small jobs and $B < m$ big jobs that all have the same number of loops $y \geq 2$. This structure is similar to the structure Schwiegelshohn (2011) identified for the parallel machine problem and the analysis is done analogously.

3.5 Fixed-size flow shops

In this section, we consider the reentrant flow shop problem in which the number of machines m is not part of the problem input but is considered a fixed parameter. We show that the problem remains weakly NP-hard under this assumption and present a pseudo-polynomial time algorithm. Additionally, we show that the problem can be approximated within arbitrary precision in polynomial time.

3.5.1 Weak NP-hardness

To show that the problem remains NP-hard if the number of machines is fixed, we present a reduction from the classical PARTITION problem (Karp, 1972).

Definition 3.5.1. *The decision problem PARTITION is defined as follows. Given positive integers* $a_1, \dots, a_q$ *and* $b = \sum_{j=1}^{q} a_j$*, is there a subset* $\mathcal{J} \subset \mathcal{S} = \{1, \dots, q\}$ *such that* $\sum_{j \in \mathcal{J}} a_j = \frac{b}{2}$*?*

Our reduction uses the fact that we can assume an optimal schedule to be non-interruptive. This implies that in an optimal schedule, we have two different progressions, starting at time 0 and at time 1, that form a partition of the jobs. The analysis of this reduction is similar to the one from Lenstra et al. (1977) for a related problem.

Theorem 3.5.2. *Minimizing the total weighted completion time for a reentrant flow shop with* $m = 2$ *and unit processing times (*$F2|reentry,\ p_{ijk} = 1|\sum w_j C_j$*) is NP-hard.*

Proof. Let $(b, a_1, \dots, a_q)$ be an instance of PARTITION. Without loss of generality, we can assume b to be even. We define an instance of the reentrant flow shop problem with unit processing times as $n := q$, $\mathcal{L}_j := a_j$, and $w_j := a_j$ for all $j = 1, \dots, n$. Note that for all jobs j, $\frac{w_j}{\mathcal{L}_j} = 1$. Given an optimal schedule for the created instance, we can assume this schedule to be non-interruptive by Theorem 3.3.6. We define $\mathcal{J}$ as the set of jobs that start their loops at even times, i.e., $\mathcal{J} = \big\{j : S_{j1} \in \{0, 2, 4 \dots\}\big\}$. The remaining jobs of set $\mathcal{S} \setminus \mathcal{J}$ then start all their loops at odd times, see Figure 3.13a. As $\frac{w_j}{\mathcal{L}_j} = 1$, the total weighted completion time of this schedule does not depend on the order of jobs within sets $\mathcal{J}$ and $\mathcal{S} \setminus \mathcal{J}$, but only on the assignment of jobs to either set. We refer to this schedule as $\sigma(\mathcal{J})$ with completion times $C_j\big(\sigma(\mathcal{J})\big)$ and to its total weighted completion time as $Z(\mathcal{J}) = \sum_{j=1}^{n} w_j C_j\big(\sigma(\mathcal{J})\big)$.

Consider another schedule that starts all jobs currently in set $\mathcal{S} \setminus \mathcal{J}$ at even times after all jobs of set $\mathcal{J}$ are completed; see Figure 3.13b. We refer to this schedule as $\sigma(\mathcal{S})$ with completion times $C_j\big(\sigma(\mathcal{S})\big)$ and to its total weighted completion time as $Z(\mathcal{S}) = \sum_{j=1}^{n} w_j C_j\big(\sigma(\mathcal{S})\big)$. Note that $Z(\mathcal{S})$ does not depend on the order of jobs, but only on the given instance of PARTITION. Therefore, it can be seen as constant in the following. We can rewrite $Z(\mathcal{S})$ as follows:

$$
\begin{aligned}
Z(\mathcal{S}) =& \sum_{j\in\mathcal{J}} w_j C_j\big(\sigma(\mathcal{S})\big) + \sum_{j\in\mathcal{S}\setminus\mathcal{J}} w_j C_j\big(\sigma(\mathcal{S})\big) \\
=& \sum_{j\in\mathcal{J}} w_j C_j\big(\sigma(\mathcal{J})\big) + \sum_{j\in\mathcal{S}\setminus\mathcal{J}} w_j \big[C_j\big(\sigma(\mathcal{J})\big) + 2\sum_{i\in\mathcal{J}} \mathcal{L}_i - 1\big] \\
=& Z(\mathcal{J}) + \sum_{j\in\mathcal{S}\setminus\mathcal{J}} w_j \left(2\sum_{i\in\mathcal{J}} \mathcal{L}_i\right) - \sum_{j\in\mathcal{S}\setminus\mathcal{J}} w_j \\
=& Z(\mathcal{J}) + 2\left(\sum_{j\in\mathcal{J}} a_j\right)\left(\sum_{j\in\mathcal{S}\setminus\mathcal{J}} a_j\right) - \sum_{j\in\mathcal{S}\setminus\mathcal{J}} a_j.
\end{aligned}
$$

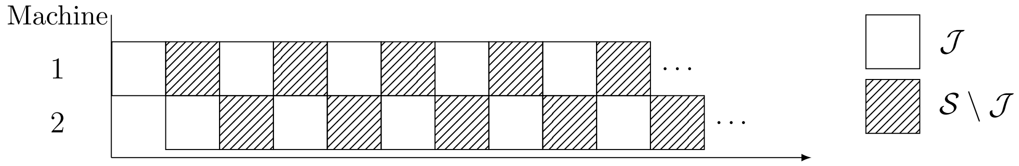
We define $c \coloneqq \sum_{j\in\mathcal{J}} a_j - \frac{1}{2}b \in \mathbb{Z}$ and the constant $y \coloneqq Z(\mathcal{S}) - \frac{1}{2}b^2 + \frac{1}{2}b$. Together with the above we get:

$$
\begin{aligned}
Z(\mathcal{J}) =& Z(\mathcal{S}) - 2\left(\frac{1}{2}b + c\right)\left(\frac{1}{2}b - c\right) + \frac{1}{2}b - c \\
=& Z(\mathcal{S}) - \frac{1}{2}b^2 + \frac{1}{2}b + 2c^2 - c \\
=& y + 2c^2 - c.
\end{aligned}
$$

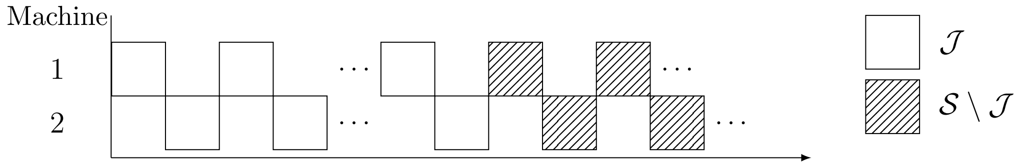
It follows that the instance of PARTITION is a yes-instance if and only if $Z(\mathcal{J}) \leq y$. $\square$

In what follows, we present a pseudo-polynomial time algorithm for $Fm|reentry,\ p_{ijk} = 1|\sum w_j C_j$ that uses dynamic programming techniques. An algorithm is said to have pseudo-polynomial runtime if its runtime is polynomial in the size of the input in unary encoding (see, e.g., Lawler, 1977; Garey & Johnson, 1978; Rothkopf, 1966; Lawler & Moore, 1969; Sahni, 1976). Due to Theorem 3.3.6, we can limit our analysis to non-interruptive schedules. This implies that once the first loop of a job is scheduled, we may assume that the remaining loops of this job are scheduled within the same progression without any interruption. Furthermore, note that jobs assigned to the same progression must be scheduled in decreasing order of $\frac{w_j}{\mathcal{L}_j}$ (this can be shown via an interchange argument). These two observations significantly reduce the class of schedules to be considered. In what follows, we assume the jobs to be indexed so that $\frac{w_1}{\mathcal{L}_1} \geq \frac{w_2}{\mathcal{L}_2} \geq \ldots \geq \frac{w_n}{\mathcal{L}_n}$ holds.

Consider now the m progressions defined by the jobs with starting times $0, 1, \ldots, m-1$.



(a) Optimal schedule $\sigma(\mathcal{J})$



(b) Schedule $\sigma(\mathcal{S})$

Figure 3.12: Schematic displays of the schedules in the proof of Theorem 3.5.2.

We denote these progressions as $\mathcal{P}_1, \ldots, \mathcal{P}_m$. It can be easily verified that $m \sum_{j=1}^{n} \mathcal{L}_j$ is an upper bound on the completion time of the last job in any optimal schedule. We define a function $V_j : \{0, 1, \ldots, m \sum_{\bar{j}=1}^{n} \mathcal{L}_{\bar{j}}\}^m \to \mathbb{N}_0 \cup \{\infty\}$ for each $j = 1, \ldots, n$. The value $V_j(t_1, \ldots, t_m)$ shall represent the minimum total weighted completion time when jobs $1, \ldots, j$ are scheduled such that the last job in progression $\mathcal{P}_i$ is completed at time t_i for $i = 1, \ldots, m$. Further, we define the function $V_0 : \{0, 1, \ldots, m \sum_{j=1}^{n} \mathcal{L}_j\}^m \to \{0, \infty\}$ with the initial condition: $V_0(0, 1, \ldots, m-1) = 0$, and $V_0(t_1, \ldots, t_m) = \infty$ for all other vectors $(t_1, \ldots, t_m) \in \{0, 1, \ldots, m \sum_{j=1}^{n} \mathcal{L}_j\}^m$.

The following dynamic programming relationship holds for all $j = 1, \ldots, n$:

$$V_j(t_1, \ldots, t_m) = \min_{i=1,\ldots,m} \big(w_j t_i + V_{j-1}(t_1, \ldots, t_i - m\mathcal{L}_j, \ldots, t_m) \big).$$

These dynamic programming equations can be solved in runtime $O\big(mn(m \sum_{j=1}^{n} \mathcal{L}_j)^m\big)$. As we consider m to be a constant in this section, this algorithm is polynomial in the number of jobs n and in the total number of loops $\sum_{j=1}^{n} \mathcal{L}_j$. In practice, tighter bounds on the runtime may be established, e.g., by finding a tighter upper bound on the last completion time of a job in an optimal schedule.

We note that the idea of our dynamic programming algorithm is similar to the dynamic programming approaches for parallel machine scheduling problems outlined in Rothkopf (1966) and Lawler & Moore (1969). The major difference of our algorithm lies in the fact that, instead of considering m parallel machines that are available at time 0, we consider progressions $\mathcal{P}_1, \ldots, \mathcal{P}_m$ starting at times $0, 1, \ldots, m-1$ and loops of jobs that are processed without interruptions.

3.5.2 A fully polynomial-time approximation scheme

In this subsection, we present a fully polynomial-time approximation scheme (FPTAS) for reentrant flow shops with unit processing times and the total weighted completion time objective. An FPTAS is defined as a family of algorithms with performance ratio $1+\varepsilon$ for all $\varepsilon > 0$ such that their runtime is polynomial in the size of the input and in $\frac{1}{\varepsilon}$. Similar to the worst-case analysis of the WLRL rule, we use dummy jobs to transform an instance of $Fm|reentry,\ p_{ijk} = 1|\sum w_jC_j$ into an instance of $Pm||\sum w_jC_j$. Then, we apply the FPTAS for $Pm||\sum w_jC_j$ proposed by Sahni (1976).

The following definition shows the transformation.

Definition 3.5.3. *For an instance I of $Fm|reentry, p_{ijk} = 1|\sum w_jC_j$, we create an instance $\bar{I}$ of $P_m||\sum_j w_jC_j$ as follows: $\bar{p}_j := m\mathcal{L}_j$, $\bar{w}_j := w_j$, $j = 1,\ldots,n$. Let $\bar{w}_{\max} := \max_{j=1,\ldots,n} \bar{w}_j$ denote the maximum weight of the jobs. We additionally define $m-1$ dummy jobs with $\bar{p}_{n+i} := i$ and $\bar{w}_j := \bar{w}_{\max}$, $j = n+1,\ldots,n+m-1$.*

The following lemma shows that we can assume that no two dummy jobs are processed on the same machine.

Lemma 3.5.4. *For a feasible schedule σ of $\bar{I}$, we can find in $O(n\log n)$ a schedule σ' such that no two dummy jobs are assigned to the same machine, the jobs are sorted according to the WSPT rule on each machine, and $\sum \bar{w}_jC_j(\sigma') \leq \sum \bar{w}_jC_j(\sigma)$.*

Proof. For jobs that are scheduled on the same machine, it is well known that sorting them according to the WSPT rule does not increase the total weighted completion time (Smith, 1956). As $\bar{w}_j = w_{\max}$ and $\bar{p}_{n+i} < \bar{p}_j$ for all $j = 1,\ldots,n$ and $i = 1,\ldots m-1$, we can assume that on each machine, all dummy jobs are completed before the first "real" job starts its processing.

Assume that in schedule σ, there are two dummy jobs, $n+j_1$ and $n+j_2$, processed as the first two jobs on the same machine i. As there are $m-1$ dummy jobs, there must be a machine i' on which no dummy job is processed. Denote as $\mathcal{J}_i$ the set of jobs scheduled on machine i apart from $n+j_1$ and $n+j_2$. Denote as j_1' the first job scheduled on machine i' and the remaining jobs on machine i' as $\mathcal{J}_{i'}$, respectively.

Consider the following two cases:

(i) $\boldsymbol{\sum_{j\in\mathcal{J}_i} \bar{w}_j \geq \sum_{j\in\mathcal{J}_{i'}} \bar{w}_j}$. We schedule dummy job $n+j_1$ as the first job on machine i'.

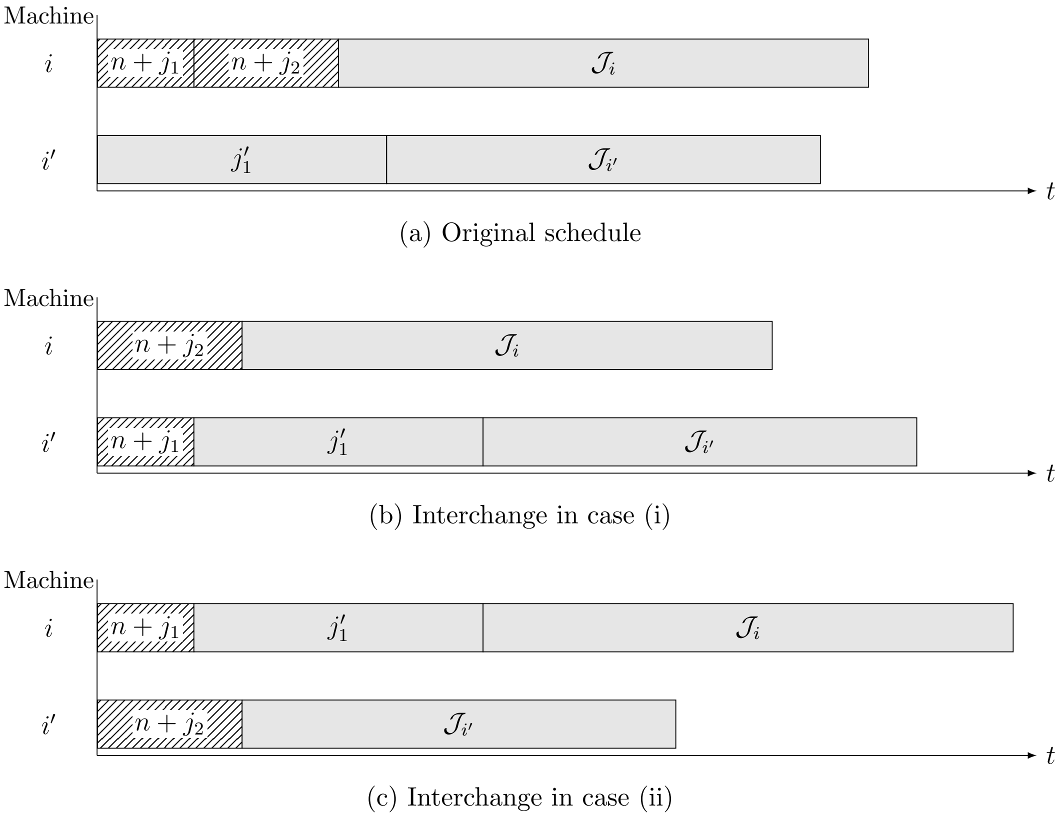


Figure 3.13: Schematic Gantt chart for the interchanges in the proof of Lemma 3.5.4

The change in the total completion time can be computed as follows:

$$\begin{aligned}&\bar{p}_{n+j_1}\left[\left(\bar{w}_{j'_1}+\sum_{j\in\mathcal{J}_{i'}}\bar{w}_j\right)-\left(\bar{w}_{n+j_2}+\sum_{j\in\mathcal{J}_i}\bar{w}_j\right)\right]\\&=\bar{p}_{i_1}\left(\bar{w}_{j'_1}-w_{\max}+\sum_{j\in\mathcal{J}_{i'}}\bar{w}_j-\sum_{j\in\mathcal{J}_i}\bar{w}_j\right)\leq 0.\end{aligned}$$

(ii) $\sum_{j\in\mathcal{J}_i}\bar{w}_j<\sum_{j\in\mathcal{J}_{i'}}\bar{w}_j$. We interchange dummy job $n+j_2$ with job j'_1. The change in the total completion time can be computed as follows:

$$\bar{p}_{n+j_1}\left(\bar{w}_{j'_1}-\bar{w}_{n+j_2}\right)+\left(\bar{p}_{j'_1}-\bar{p}_{n+j_2}\right)\left(\sum_{j\in\mathcal{J}_i}\bar{w}_j-\sum_{j\in\mathcal{J}_{i'}}\bar{w}_j\right)<0.$$

The described interchanges can be performed in constant time for fixed m while sorting each machine by non-increasing ratios $\frac{w_j}{p_j}$ may require time $O(n\log n)$.

□

Due to the previous lemma, we know that we can apply a simple interchange on schedules generated by Sahni's FPTAS to compute schedules in which all dummy jobs start processing at time 0 on separate machines. This allows us to devise an FPTAS for the reentrant flow shop problem.

Theorem 3.5.5. *There exists an FPTAS for $Fm|reentry, p_{ijk} = 1|\sum w_j C_j$.*

Proof. Let $\varepsilon > 0$ be arbitrary but fixed. For an instance I of $Fm|reentry, p_{ijk} = 1|\sum w_j C_j$, create an instance $\bar{I}$ of $Pm||\sum w_j C_j$ as described in Definition 3.5.3. Let $\phi\big(OPT(I)\big)$ and $\phi\big(OPT(\bar{I})\big)$ denote the optimal objective function values of instances I and $\bar{I}$, respectively. We apply Sahni's FPTAS and the interchanges described in Lemma 3.5.4 to instance $\bar{I}$ to obtain a schedule $\sigma(\bar{I})$ in time $O\big(n(\frac{n^2}{\varepsilon})^{m-1} + n\log n\big)$. For this schedule, $\sum_{j=1}^{n+m-1} \bar{w}_j C_j\big(\sigma(\bar{I})\big) \leq (1+\varepsilon)\phi\big(OPT(\bar{I})\big)$ holds. We define a schedule $\sigma(I)$ for instance I in which all jobs have the same completion times as the "real" jobs in schedule $\sigma(\bar{I})$. Additionally, define $\Delta := w_{\max}\sum_{i=1}^{m}(i-1) = w_{\max}\frac{m\cdot(m-1)}{2}$. We can express the total weighted completion time of σ as $\sum_{j=1}^{n} w_j C_j\big(\sigma(I)\big) = \sum_{j=1}^{n} \bar{w}_j C_j\big(\sigma(\bar{I})\big) + \Delta$. As the optimal value $\phi\big(OPT(I)\big)$ is lower bounded by $\sum_{j=1}^{n} w_j m\mathcal{L}_j \geq m w_{\max}$, we have:

$$\sum_{j=1}^{n} w_j C_j\big(\sigma(I)\big) \leq (1+\varepsilon)\phi\big(OPT(I)\big) + \varepsilon\Delta \leq (1+\varepsilon\frac{m+1}{2})\phi\big(OPT(I)\big).$$

As m is constant, this completes the proof. □

3.6 Conclusion and outlook

In this paper, we considered scheduling a set of jobs that must go through multiple loops in a flow shop with the objective of minimizing the total (weighted) completion time. We introduced non-interruptive schedules and showed that there is a non-interruptive schedule that minimizes the total weighted completion time. We showed that the general problem of minimizing the total weighted completion time is strongly NP-hard. We introduced the priority rule LRL and showed that it minimizes the total completion time and the total weighted completion time if the weights are agreeable. We introduced the priority rule WLRL and provided a tight performance guarantee of about 1.2. Additionally, we analyzed reentrant flow shops with a fixed number of machines. We presented a dynamic programming algorithm that has pseudo-polynomial runtime in the total number of loops and an FPTAS.

For future research, we propose to examine the total completion time objective for reentrant flow shops with machine-ordered and proportionate processing times. For both these processing time structures, one can easily create an instance in which optimal schedules allow for unforced idleness. For machine-ordered processing times, one could analyze a modification of the WLRL priority rule that alternates between scheduling jobs according to WLRL and waiting for a job to complete its current loop. For proportionate processing times, one can show that the problems of minimizing the makespan and minimizing the total (unweighted) completion time are strongly NP-hard by a reduction from 3-PARTITION. One could analyze a generalization of the WLRL rule that takes into account the processing times of jobs, i.e., priority values $\frac{w_j}{p_j \cdot \mathcal{L}_j}$. In a computational experiment, this rule's empirical average performance ratio was about 1.15, while the empirical worst case was about 2.09. It would be interesting to investigate whether a constant bound on the worst-case performance ratio of this priority rule can be established. Additionally, it should be analyzed how the WLRL priority rule can be embedded in more complex scheduling heuristics for reentrant shops with arbitrary processing times. For these problems, combining priority rules with constraint programming may be a promising direction.

Another direction for future research is stochastic flow shops with reentry. In addition to stochastic processing times, which have been studied for conventional flow shops (see, e.g., Pinedo, 2022; Emmons & Vairaktarakis, 2012), one could study stochastic reentries. In real-world production systems, uncertainty in reentries may arise if a job must undergo repair or rework steps after a defect has been detected. For such a setting, conditions could be established under which the "Most Expected Remaining Loops First" rule minimizes the expected makespan or the "Weighted Least Expected Remaining Loops First" rule minimizes the expected total weighted completion time. Also, one could investigate performance bounds for these priority rules that may depend on the coefficients of variation of the underlying distributions.

Bibliography

Chen, J.-S. (2006). A branch and bound procedure for the reentrant permutation flow-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *29*, 1186–1193. doi:10.1007/s00170-005-0017-x.

Choi, S.-W., & Kim, Y.-D. (2007). Minimizing makespan on a two-machine re-entrant flowshop. *Journal of the Operational Research Society*, *58*, 972–981. doi:10.1057/palgrave.jors.2602220.

Choi, S.-W., & Kim, Y.-D. (2008). Minimizing makespan on an m-machine re-entrant flowshop. *Computers & Operations Research*, *35*, 1684–1696. doi:10.1016/j.cor.2006.09.028. Part Special Issue: Algorithms and Computational Methods in Feasibility and Infeasibility.

Danping, L., & Lee, C. K. (2011). A review of the research methodology for the re-entrant scheduling problem. *International Journal of Production Research*, *49*, 2221–2242. doi:10.1080/00207541003720350.

Eastman, W. L., Even, S., & Isaacs, I. M. (1964). Bounds for the optimal scheduling of n jobs on m processors. *Management Science*, *11*, 268–279. doi:10.1287/mnsc.11.2.268.

Emmons, H., & Vairaktarakis, G. (2012). *Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications*. (1st ed.). New York, NY: Springer. doi:10.1007/978-1-4614-5152-5.

Garey, M. R., & Johnson, D. S. (1978). "Strong" NP-completeness results: Motivation, examples, and implications. *Journal of the ACM*, *25*, 499–508. doi:10.1145/322077.322090.

Graham, R., Lawler, E., Lenstra, J., & Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In P. Hammer, E. Johnson, & B. Korte (Eds.), *Discrete Optimization II* (pp. 287–326). Elsevier volume 5 of *Annals of Discrete Mathematics*. doi:10.1016/S0167-5060(08)70356-X.

Graves, S. C., Meal, H. C., Stefek, D., & Zeghmi, A. H. (1983). Scheduling of re-entrant flow shops. *Journal of Operations Management*, *3*, 197–207. doi:10.1016/0272-6963(83)90004-9.

Jing, C., Huang, W., & Tang, G. (2011). Minimizing total completion time for re-entrant flow shop scheduling problems. *Theoretical Computer Science*, *412*, 6712–6719. doi:10.1016/j.tcs.2011.08.030.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, & J. D. Bohlinger (Eds.), *Complexity of Computer Computations* (pp. 85–103). Boston, MA: Springer US. doi:10.1007/978-1-4684-2001-2_9.

Kawaguchi, T., & Kyan, S. (1986). Worst case bound of an LRF schedule for the mean weighted flow-time problem. *SIAM Journal on Computing*, *15*, 1119–1129. doi:10.1137/0215081.

Kubiak, W., Lou, S. X. C., & Wang, Y. (1996). Mean flow time minimization in reentrant job shops with a hub. *Operations Research*, *44*, 764–776. doi:10.1287/opre.44.5.764.

Lawler, E. L. (1977). A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. In P. Hammer, E. Johnson, B. Korte, & G. Nemhauser (Eds.), *Studies in Integer Programming* (pp. 331–342). Elsevier volume 1 of *Annals of Discrete Mathematics*. doi:10.1016/S0167-5060(08)70742-8.

Lawler, E. L., & Moore, J. M. (1969). A functional equation and its application to resource allocation and sequencing problems. *Management Science*, *16*, 77–84. doi:10.1287/mnsc.16.1.77.

Lenstra, J., Rinnooy Kan, A., & Brucker, P. (1977). Complexity of machine scheduling problems. In P. Hammer, E. Johnson, B. Korte, & G. Nemhauser (Eds.), *Studies in Integer Programming* (pp. 343–362). Elsevier volume 1 of *Annals of Discrete Mathematics*. doi:10.1016/S0167-5060(08)70743-X.

Lev, V., & Adiri, I. (1984). V-shop scheduling. *European Journal of Operational Research*, *18*, 51–56. doi:10.1016/0377-2217(84)90260-1.

Mönch, L., Fowler, J. W., Dauzère-Pérès, S., Mason, S. J., & Rose, O. (2011). A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling*, *14*, 583–599. doi:10.1007/s10951-010-0222-9.

Pan, J.-H., & Chen, J.-S. (2003). Minimizing makespan in re-entrant permutation flow-shops. *Journal of the Operational Research Society*, *54*, 642–653. doi:10.1057/palgrave.jors.2601556.

Pfund, M. E., Mason, S. J., & Fowler, J. W. (2006). Semiconductor manufacturing scheduling and dispatching. In J. W. Herrmann (Ed.), *Handbook of Production Scheduling* (pp. 213–241). Boston, MA: Springer US.

Pinedo, M. (2022). *Scheduling: Theory, Algorithms, and Systems*. (6th ed.). Cham: Springer International Publishing. doi:10.1007/978-3-031-05921-6.

Rothkopf, M. H. (1966). Scheduling independent tasks on parallel processors. *Management Science*, *12*, 437–447. doi:10.1287/mnsc.12.5.437.

Sahni, S. K. (1976). Algorithms for scheduling independent tasks. *Journal of the ACM*, *23*, 116–127. doi:10.1145/321921.321934.

Schwiegelshohn, U. (2011). An alternative proof of the kawaguchi–kyan bound for the largest-ratio-first rule. *Operations Research Letters*, *39*, 255–259. doi:10.1016/j.orl.2011.06.007.

Shufan, E., Grinshpoun, T., Ikar, E., & Ilani, H. (2023). Reentrant flow shop with identical jobs and makespan criterion. *International Journal of Production Research*, *61*, 183–197. doi:10.1080/00207543.2021.1957505.

Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, *3*, 59–66. doi:10.1002/nav.3800030106.

Wang, M. Y., Sethi, S. P., & van de Velde, S. L. (1997). Minimizing makespan in a class of reentrant shops. *Operations Research*, *45*, 702–712. doi:10.1287/opre.45.5.702.

Wu, K., Huang, E., Wang, M., & Zheng, M. (2022). Job scheduling of diffusion furnaces in semiconductor fabrication facilities. *European Journal of Operational Research*, *301*, 141–152. doi:10.1016/j.ejor.2021.09.044.

Yu, T.-S., & Pinedo, M. (2020). Flow shops with reentry: Reversibility properties and makespan optimal schedules. *European Journal of Operational Research*, *282*, 478–490. doi:/10.1016/j.ejor.2019.09.036.

Zheng, S., He, Z., Yang, Z., Chu, C., & Wang, N. (2023). Effective upper and lower bounds for a two-stage reentrant flexible flow shop scheduling problem. *Computers & Operations Research*, *153*, 106183. doi:10.1016/j.cor.2023.106183.