TU WIEN Informatics

Solving Multi-Mode Resource-Constrained Multi-Project Scheduling Problems by Hybrid Algorithms

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Arben Ahmeti
Registration Number 1228512

to the Faculty of Informatics

at the TU Wien

Advisor: Priv.-Doz. Dr. Nysret Musliu

The dissertation has been reviewed by:

Luca Di Gaspero	Frédéric Lardeux

Vienna, 11th February, 2022

Arben Ahmeti

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Arben Ahmeti

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 11. Februar 2022

Arben Ahmeti

I dedikohet familjes sime, veçanërisht kujtimit për nënën time të dashur, miqve të mi, veçanërisht atyre që ranë në fushën e nderit për të jetuar në përjetësi, Rifat Mëziu, Malush Ahmeti, Zenel Mëziu, Agim Mëziu...

Dedicated to my family, especially to the loving memory of my Mum, my friends, especially those who fell in the field of honor to live in eternity, Rifat Mëziu, Malush Ahmeti, Zenel Mëziu, Agim Mëziu...

Acknowledgements

First of all, I would like to thank my advisor Nysret Musliu for his continuous support in completing this study. The fruitful discussions with him and his guidance and instructions were motivating and crucial for my success. Working with him in writing papers has boosted my confidence and made me want to contribute even more to science. I will always be grateful for his support and kindness.

A very special thank you goes to my entire family who has supported me in every way. Without their support, none of this would have been possible. Even in the moments when I thought my world was upside down, I made it through with their support. I hope and believe that my achievements will be a reward and satisfaction for them. I say to Dea and Drion that I am now finished with my work. I thank you for your patience, your love and your existence. You are the shining lights in my eyes.

A big thank you goes to Tony Wauters, assistant professor at KU Leuven, for his support in running our algorithm in the MISTA 2013 benchmark environment and providing the right feedback. A big thank you also goes to Martin Josef Geiger, professor at Helmut Schmidt University, for his support with important materials on the MISTA 2013 competition. I owe a big thank you to my colleagues and friends for their feedback and support.

The financial support from the Austrian Federal Ministry of Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Society deserve my sincere thanks at this point.

Kurzfassung

Das Projektplanungsproblem befasst sich mit der Planung der Aktivitäten eines Projekts (oder mehrerer Projekte) unter verschiedenen Arten von Beschränkungen, hauptsächlich knappe Ressourcen, Zeit und Vorrangbeschränkungen. Die Aufteilung knapper Ressourcen auf die Projektaktivitäten unter Berücksichtigung aller Nebenbedingungen und der Optimierung verschiedener Ziele ist ein äußerst schwieriges Problem. Selbst die klassischste und einfachste Variante, das ressourcenbeschränkte Projektplanungsproblem (RCPSP), gehört zur Klasse der NP-schweren Probleme. Projektplanungsprobleme treten in vielen Bereichen der Industrie und in anderen Lebensbereichen auf. Daher waren sie schon immer ein interessantes und wichtiges Thema für Forschung und Industrie.

Im Laufe der Jahrzehnte haben sich verschiedene Varianten von ressourcenbeschränkten Projektplanungsproblemen herausgebildet, da unterschiedliche industrielle Anforderungen modelliert werden müssen. Dementsprechend haben Forscher verschiedene Lösungsstrategien vorgeschlagen, um diese Probleme anzugehen. Im Allgemeinen können wir einige der wichtigsten Lösungsansätze als heuristisch, meta-heuristisch, hyper-heuristisch, hybride und exakte Methoden kategorisieren. Obwohl Forscher hervorragende Lösungen für verschiedene Varianten von Projektplanungsproblemen geliefert haben, sind die optimalen Lösungen für viele Benchmark-Probleme noch unbekannt.

In dieser Arbeit stellen wir neue Lösungsansätze vor, darunter exakte, metaheuristische und hybride Methoden für eines der komplexesten Projektplanungsprobleme, das ressourcenbeschränkte Multiprojektplanungsproblem (MRCMPSP). MRCMPSP stellt reale Situationen genauer dar. Wir stellen mehrere neue Ideen zur Lösung des MRCMPSP vor, darunter neue lokale Suchtechniken auf der Grundlage von Min-Conflicts-Algorithmus und iterierter lokaler Suche, ein Constraint-Programmiermodell und hybride Methoden. Unsere Lösungsmethoden wurden erfolgreich auf andere Varianten des Projektplanungsproblems angewandt, wie das ressourcenbeschränkte Multi-Mode-Projektplanungsproblem (MRCPSP) und das ressourcenbeschränkte Multi-Projektplanungsproblem (RCMPSP). Darüber hinaus wurde unsere innovative meta-heuristische Methode, die auf Min-Conflicts-Algorithmus und Tabu-Suche basiert, erfolgreich zur Lösung des Fahrzeugrouting- und Schedulingproblems mit Lieferung und Installation von Maschinen eingesetzt, das kürzlich von der EURO Working Group in Vehicle Routing and Logistics Optimization (VeRoLog) und ORTEC eingeführt wurde.

Unsere Methoden wurden auf bestehende Benchmark-Instanzen für mehrere Projektplanungsprobleme angewandt, darunter MRCMPSP, MRCPSP und RCMPSP. Die Ergebnisse zeigen, dass unsere hybriden Methoden, die Meta-Heuristiken und Constraint-Programmierung verwenden, die Ergebnisse von State-of-the-Art-Methoden für diese Klasse von Problemen verbessern und viele neue obere Schranken für Benchmark-Instanzen liefern.

Abstract

The project scheduling problem deals with scheduling the activities of a project (or multiple projects) under various types of constraints, mainly scarce resources, time, and precedence constraints. Allocating scarce resources among project activities, taking into account all constraints and optimizing various objectives, is an extremely difficult problem. Even the most classical and simplest variant, the resource-constrained project scheduling problem (RCPSP), belongs to the class of NP-hard problems. Project scheduling problems occur in many areas of industry and other real-world situations. Therefore, they have always been an interesting and important topic for research and industry.

Over the decades, different variants of resource-constrained project scheduling problems have emerged as different industrial requirements need to be modeled. Accordingly, researchers have proposed various solution strategies to address these problems. In general, we can categorize some of the main solution approaches as heuristic, meta-heuristic, hyper-heuristic, hybrid and exact methods. Although researchers have provided outstanding solutions for different variants of project scheduling problems, the optimal solutions for many benchmark problems are still unknown.

In this thesis, we present new solution approaches, including exact, meta-heuristic, and hybrid methods for one of the most generalized forms of the project scheduling problem, the resource-constrained multi-project scheduling problem (MRCMPSP). MRCMPSP is a more accurate representation of the real-world environment. We present several new ideas for solving the MRCMPSP, including new local search techniques based on min-conflicts and iterated local search, a constraint programming model, and hybrid methods. Our solution methods have been successfully applied to other variants of the project scheduling problem, such as the resource-constrained multi-mode project scheduling problem (MRCPSP) and the resource-constrained multi-project scheduling problem (RCMPSP). In addition, our innovative meta-heuristic method based on min-conflicts and tabu search has been successfully applied to solve the vehicle routing and scheduling problem with delivery and installation of machinery (DIM) recently introduced by the EURO Working Group in Vehicle Routing and Logistics Optimization (VeRoLog) and ORTEC.

Our methods were applied on existing benchmark instances for several project scheduling problems, including MRCMPSP, MRCPSP, and RCMPSP. Computational results show that our hybrid methods, which use meta-heuristics and constraint programming, improve

the results of state-of-the-art methods for this class of problems and provide many new upper bounds for benchmark instances.

Contents

CHAPTER 1

Introduction

Scheduling problems represent a class of various combinatorial optimization problems that are of great interest to research and industry. In particular, project scheduling is a major representative of this type of problems. It is an important part of project management in many industries and business areas. The project scheduling problem occurs when resources are scarce while trying to achieve the objectives of the project. Allocation of insufficient resources to competing project activities determines their start and completion time. Most research to date has focused on scheduling activities of the resource-constrained project scheduling problem (RCPSP), usually considering the optimization of time objectives, i.e., minimizing project duration. In real situations, the need to optimize different objectives simultaneously arises very often. Meanwhile, more advanced types of problems related to project scheduling have been introduced.

Nowadays, in project management reality, the real-life environment is very complex. It includes many projects running concurrently and competing for the same insufficient resources while trying to achieve the project goals. Activities within projects can be executed in one or more different modes, each mode having its own characteristics in terms of allocated resources and time duration. The RCPSP was not sufficient to adequately describe the various real-world situations. Therefore, additional extensions to the RCPSP emerged. One of these extensions is the Multi-Mode Resource-Constrained Multiple Projects Scheduling Problem (MRCMPSP, abbreviated MMRCMPSP by some authors). It can be considered as a generalization of the RCPSP. MRCMPSP is the main topic of this thesis. MRCMPSP includes several projects defined by a set of activities performed in one or more different ways (modes), taking into account precedence constraints and an insufficient amount of different types of resources. Assigning modes to activities within a multiple project environment while considering hard and soft constraints makes the MRCMPSP a challenging problem. Any feasible solution should satisfy all hard constraints. The goal is to optimize the feasible solution(s) by an objective function related to the soft constraints. Project scheduling problems are hard to solve, and even the

basic variant (RCPSP) belongs to the class of NP-hard optimization problems [BLK83]. In MRCMPSP, even just creating a feasible schedule is an NP-complete problem [KD97b] if the activities require more than one non-renewable resource.

Despite the valuable research work on solving the MRCMPSP, it remains a very challenging problem. We could categorize some of the most important solution approaches for MRCMPSP with different optimization objectives as heuristic (e.g. [BBZ+13]), meta-heuristic (e.g. [Gei17, HHW21, KM20, KZV+16]), hyper-heuristic (e.g. [AKK+16]), and few hybrid ([TSCS16, EAM+19, KREK19, LIMMS13]) and exact methods (e.g. [SH17], [SV93]). However, the MRCMPSP still seems to be a challenging problem. Although recent algorithms have found very good solutions to existing benchmark problems, optimal solutions for many instances are still unknown. Therefore, the development of new approaches in this context is important for progress in this area. Investigating new hybrid approaches that combine meta-heuristics and exact methods is a promising direction for solving MRCMPSP and other variants of project scheduling problems.

1.1 Aims of this thesis

First, we will investigate the design of an innovative meta-heuristic and new neighborhood operators for MRCMPSP. In addition, we intend to build an exact model, more specifically a constraint programming model, for the problem as the work progresses. To date, this would be the first constraint programming model for the MRCMPSP (to the best of our knowledge). Our final aim is to design a hybrid solution approach that combines the meta-heuristics and the exact methods developed to this point. We will consult the literature to design the architecture of the hybrid model.

In addition, we intend to validate and evaluate our methods for a wide range of project scheduling problems, and not only for these. Benchmark problem instances and benchmark execution conditions of state-of-the-art algorithms from the literature would be used for proper evaluation and comparison of the new methods.

The main aims of this thesis are:

- To develop innovative meta-heuristic methods for MRCMPSP that explore the neighborhood efficiently and perform well on large problem instances. We aim to introduce new neighborhood operators and evaluate their performance. The performance of meta-heuristic methods will be evaluated on a broader range of problems and application domains.
- Development and evaluation of an exact method based on a constraint programming model. Various search strategies and configurations will be investigated in this context.
- Propose new hybrid solution approaches combining meta-heuristic techniques with constraint programming. In addition, different architectures of hybrid models from the literature will be experimented with.

- Test and evaluate hybrid solution approaches for the MRCMPSP and a broader range of the most popular variants of project scheduling problems. Compare and analyze results for benchmark problems with state-of-the-art algorithms.

1.2 Main contributions

In this thesis, we present new solution approaches for MRCMPSP, including exact, meta-heuristic, and hybrid techniques. Our solution approaches improve upon state-of-the-art results and solutions for MRCMPSP and some of the most well-known variants of the project scheduling problem. The main contributions of this thesis are:

1. We propose a new method based on a combination of the min-conflicts heuristic [MJPL92] and tabu search. We propose three new neighborhood operators and evaluate their performance. Experimental results show that the new operators are useful for this problem.

2. We implemented an iterated local search algorithm that applies the proposed min-conflicts heuristic and the new neighborhood operators. Although iterated local search has already been applied to this problem in the literature, we use a new local search technique and three new operators. We tested our approach and compared the results with state-of-the-art results for the MRCMPSP. Our method achieves competitive and comparable results to the third-ranked solver. We also applied our method to the well-known MRCPSP benchmark problems for project scheduling (MMLIB library [1]). Our method provides new upper bounds for eleven benchmark problems from this library.

3. As part of the VeRoLog Solver Challenge 2019, we implemented a solver for the vehicle routing and scheduling with delivery and installation of machines problem that employs multiple neighborhood operators based on the min-conflicts concept, similar to our approach for the MRCMPSP. Our solver yielded promising results by ranking second for the first instance, third for six instances, fourth for thirteen instances, fifth for four instances, and sixth for one instance.

4. We investigate the simulated annealing heuristic for MRCMPSP and integrate it into an iterated local search framework.

5. We provide a Constraint Programming (CP) model for MRCMPSP and apply a state-of-the-art CP solver to solve existing problem instances. Although constraint programming has been used previously for related project scheduling problems, to the best of our knowledge, this is the first time CP has been applied to MRCMPSP, which contains several extensions. Our model was tested with 30 benchmark instances from MISTA 2013 challenge. Long runs of the algorithm resulted in

[1] http://mmlib.eu/solutions.php

new upper bounds for six instances. Executing the algorithm under time limit constraints (five minutes per instance) led to the best results for three instances compared to other best solvers.

6. We developed three hybrid algorithms combining the CP model and different variants of meta-heuristic approaches. Our best hybrid algorithm improved the results and outperformed one of the previous best state-of-the-art solvers [AKK+16] for MRCMPSP for most instances. We provide new upper bounds for most MRCMPSP benchmark instances (eighteen new upper bounds and four equal upper bounds out of thirty), over one hundred new upper bounds for MMLIB MRCMPSP benchmark instances, and 81 new upper bounds out of 140 for the resource-constrained multiple-project scheduling problem (RCMPSP) instances and identical results in 28 of them.

Some of the results of this thesis have already been published in the following papers:

- Arben Ahmeti and Nysret Musliu. Min-conflicts heuristic for multi-mode resource-constrained projects scheduling. In Hernán E. Aguirre and Keiki Takadama, editors, Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, Kyoto, Japan, July 15-19, 2018, pages 237–244. ACM, 2018 [AM18].
- Arben Ahmeti and Nysret Musliu. Hybridizing constraint programming and meta-heuristics for multi-mode resource-constrained multiple projects scheduling problem. Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT, volume 1, pages 188-206, 2021 [AM21].
- Valon Kastrati, Arben Ahmeti, and Nysret Musliu. Solving vehicle routing and scheduling with delivery and installation of machines using ILS. Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT, volume 1, pages 207-223, 2021 [KAM21].

The work presented in Chapter 5 is extended and is under revision in the "Journal of Scheduling". The work presented in Chapter 8 is currently being prepared for further publication in the near future.

1.3 Organization of this thesis

This thesis is organized into eight chapters. This chapter (Chapter 1) is followed by:

- Chapter 2 describes the main components and variant characteristics of project scheduling problems, focusing on the MRCMPSP, MRCPSP, and RCMPSP problem variants we have worked on. We also present the MRCMPSP problem definition, project scheduling problem representation, classification and notation schemes, and benchmark datasets from the literature.

- Chapter 3 describes in detail our new meta-heuristic approach based on the min-conflicts heuristic and in its combination with several new neighborhood operators within an iterated local search framework. We also present the computational results of our method on benchmark datasets and their comparison with other state-of-the-art methods. Several extensions and improvements to the meta-heuristic method are presented. New neighborhood operators involved in the extensions and modifications are explained in detail and experimentally evaluated on benchmark instances. In addition, several variants of meta-heuristic approaches are analyzed and evaluated.

- Chapter 4 introduces a constraint programming model for solving MRCMPSP. It is based on an existing model, which has been applied for the related problem MRCPSP. The fine-tuning of the CP model and its experimental evaluation are also presented.

- Chapter 5 explains the hybrid solution approaches that combine meta-heuristic techniques and the exact method described in the earlier chapters. Different classifications and taxonomies for hybrid approaches from the literature are analyzed in this chapter. The developed hybrid approaches are tested and experimentally evaluated on different variants of benchmark project scheduling problems. Moreover, the performance of our approaches is compared with state-of-the-art methods for MRCMPSP, MRCPSP, and RCMPSP problems.

- Chapter 6 introduces the implementation and integration of the simulated annealing heuristic into our meta-heuristic solution. Furthermore, the role and impact of the improved meta-heuristic within the hybrid approaches are analyzed and experimentally evaluated on the benchmark instances.

- Chapter 7 describes an implementation of the min-conflicts heuristic within an ILS framework for solving vehicle routing and scheduling with delivery and installation of machines.

- Finally, Chapter 8 summarizes the research results, draws conclusions, and discusses future work.

CHAPTER 2

Project scheduling problems

Project scheduling problems are about scheduling activities of a project (or multiple projects) under different types of constraints, mainly resource, time and precedence constraints, while considering the optimization of different types of objectives. They are encountered in many areas of industry and in real-world situations. Therefore, project scheduling problems have been an interesting and important topic for research and industry.

2.1 Main components of project scheduling problems

In most of the literature, the characteristics and differences between project scheduling problems are analyzed in terms of their main components: activities, resources, precedence relationships, and objectives (performance measures) [SSW94].

2.1.1 Activities

Every project is comprised of a certain number of activities (tasks or operations). Activities can be preemptive or non-preemptive (the activity cannot be interrupted until it is finished). In multi-mode versions of the project scheduling problem, activities can be executed in one or more modes. The execution mode of an activity determines the time required to complete the activity and the specific resource requirements. In most representations, the first and last activities are dummy activities, that is, activities with a duration of zero and no resource requirement.

2.1.2 Resources

The activities of a project may use different resources to complete. Resources are classified based on types and categories [Bła86]. Classification of resources by types is based on the functions they perform, i.e., resources of the same type perform the

same functions. Classification into categories can be interpreted from the perspective of resource constraints and divisibility. [Bła86] distinguishes three categories of resources from the perspective of resource constraints. Renewable resources have a fixed capacity per time unit (period), non-renewable resources have a fixed capacity for the entire project duration and doubly constrained resources have a fixed capacity per time unit and for the entire project duration. Furthermore, in the project scheduling problem presented in [WKS+16], renewable resources are further divided into local renewable resources and global renewable resources. Local renewable resources are available to each individual project, and global renewable resources are shared among all projects.

Partially renewable resources introduced by [BDKS99] are available within a subset of the time units. Partially renewable resources are considered a generalization for renewable and non-renewable resources. Renewable resources are considered as partially renewable resources with an associated subset of exactly one time unit, whereas non-renewable resources as partially renewable resources with an associated subset for the entire planning horizon long.

In terms of resource divisibility, resource categories can be discrete and continuous. A discrete resource is a resource that can be allocated to activities in discrete quantities. Continuous resources can be allocated to activities in arbitrary (a priori unknown) quantities from a given interval.

2.1.3 Precedence relations

In many project scheduling problems, activities may be subject to precedence constraints, i.e., some activities cannot start until some others are completed. In the literature, project networks are presented as directed graphs, the Activity-on-Node (AON) or the Activity-on-Arc (AOA) representation [KW59, MRCF59]. In the AON representation, the activities are nodes of the graph, and the precedence relation between them is a directed arc. In the case of AOA, the nodes of the graph are events, and the activities are arcs of the graph that terminate at or originate from a node. We will provide more information on this topic in a later section.

In addition to the precedence component itself, in some models of project scheduling problems, i.e., problems with generalized precedence relations, time windows (minimal and maximal time lags) between the start and finish of the pair of activities are important factors to be considered.

2.1.4 Objectives

The objectives in project scheduling problems may be time-based, cost-based, or resource-based. Moreover, certain complex project scheduling problems may involve the simultaneous optimization of multiple objectives [Sł81, WKS+16]. The most common time-based objective is the minimization of the project makespan, i.e., the time span from the start to the end of the project. There are other time-based goal variants, e.g., the minimization of the weighted delays, the minimization of the total number of tardy activities, the

minimization of the mean weighted flow time of activities [SD98, RHA13], and so forth. Cost-based objectives are concerned with the situation in project scheduling when positive cash flow is an essential factor for project implementation. One of the most important representatives of this category of objectives is maximizing the net present value (NPV) [BDP81].

Resource-based objectives are concerned with minimizing the amount of change in resources used from one unit of time to another. Usually, the objective is to minimize the maximum changes, minimize the sum of all changes, or minimize the sum of all squared changes [BTR94, NZ00, BSZ06].

2.2 Project scheduling problem variants

Real-life projects coming from industry and project management have different characteristics, are very complex, and are subject to severe constraints. Therefore, many variants of project scheduling problems are subject to research and analysis. These variants differ from each other in terms of the components already mentioned. Figure 2.1 shows the diagram of these components (see also [HBS18]). In the following subsections, we will analyze some of the resource-constrained variants as special cases of project scheduling problems.

2.2.1 The resource constrained project scheduling problem (RCPSP)

RCPSP, as a generalization of the job shop scheduling problem introduced by [Gra66], is the most classical and most straightforward version of the project scheduling problem. It deals with scheduling activities of a project that require a certain amount of limited resources and are subject to precedence constraints. In the case of RCPSP, resources are renewable, and activities are executed in only one mode.

RCPSP has been studied extensively in the literature. According to [BLK83], RCPSP belongs to the group of NP-hard optimization problems. Consequently, all more generalized variants of the RCPSP are at least as complex as RCPSP.

Example 1: An example for the RCPSP expressed as a directed AON graph can be seen in Figure 2.2. Using this example, we assume that:

- R - the renewable resources set and $|R| = 1$,
- P - a single project instance comprised of eight activities, i.e., $P = \{1, 2, 3, 4, 5, 6, 7, 8\}$, where $P_j = 1$ and $P_j = 8$ are dummy activities,
- d_j - duration of activity P_j, $d_j \in D$, and $D = \{0, 1, 1, 2, 1, 2, 2, 0\}$,
- l_{jr} - units of resource r used by activity j in every period (time unit) it is executing, $l_{jr} \in L$, and $L = \{0, 2, 2, 1, 1, 1, 1, 0\}$,

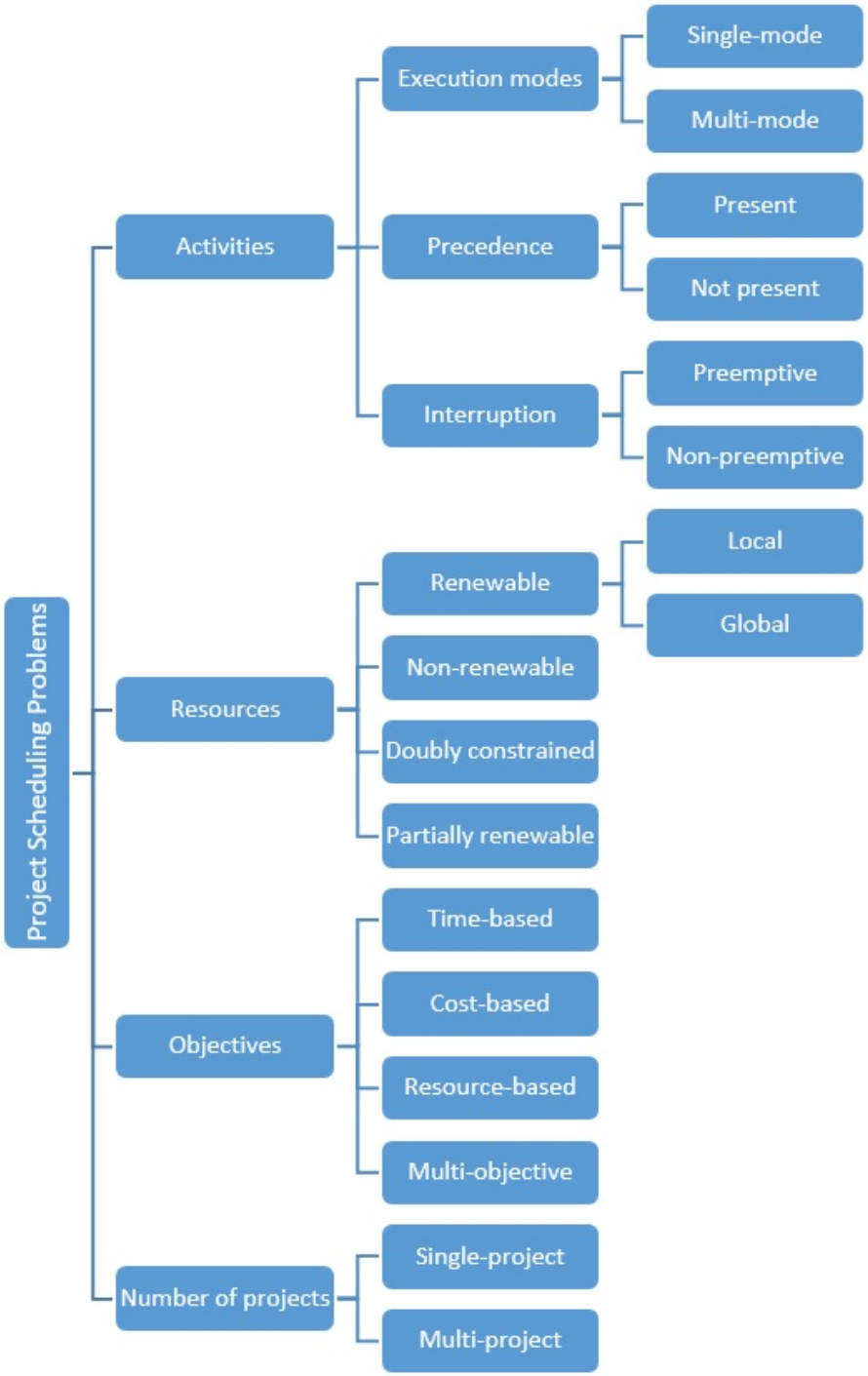

Figure 2.1: Variants of project scheduling problems

- T - upper bound of the project's makespan,
- C_{rt} - availability of resource r in period t, where $t \in \{1, .., T\}$.

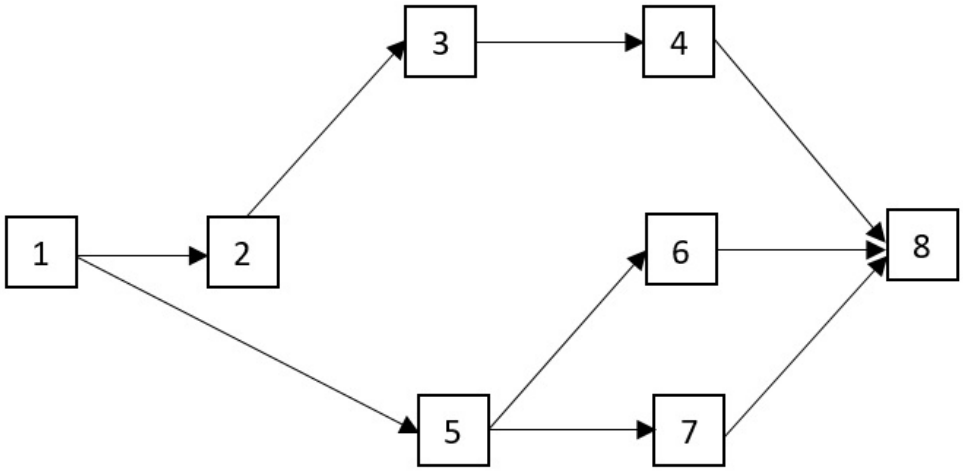

Figure 2.2: An example of the RCPSP problem

A feasible schedule for the RCPSP problem is a $|P|$-tuple $S = \{s_1, .., s_8\}$, where s_j is the start time of activity j, such that all precedence and resource constraints are respected. A feasible schedule for our example could be $S = \{0, 0, 1, 2, 2, 3, 4, 6\}$ and would look like in Figure 2.3.

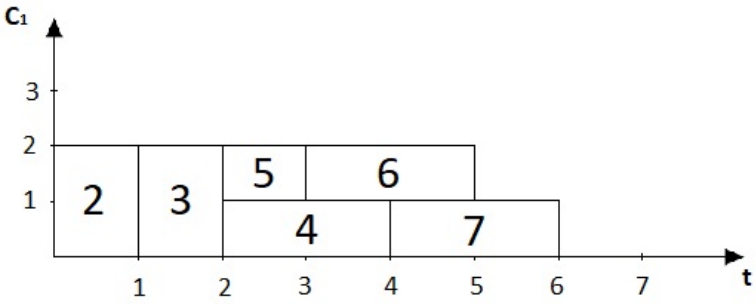

Figure 2.3: A feasible schedule for the example

2.2.2 The multi-mode resource-constrained project scheduling problem (MRCPSP)

MRCPSP extends the RCPSP problem from the perspective of execution modes of activities, i.e., activities have one or more execution modes. MRCPSP was first introduced

by [Elm77]. As mentioned earlier, the execution mode of an activity determines its duration and its specific resource requirements. In addition to just scheduling activities, as is done in the RCPSP, the MRCPSP must also consider mode assignment. Since the RCPSP belongs to the class of NP-hard optimization problems [BLK83], the MRCPSP is also at least an NP-hard problem. Moreover, [KD97b] proved that when activities require more than one non-renewable resource, creating a feasible schedule is an NP-complete problem in its own right. When each activity has been assigned one of its modes, the problem transforms into RCPSP. So far, most of the work on project scheduling problems has been devoted to solving RCPSP and MRCPSP.

2.2.3 The resource-constrained multi-project scheduling problem (RCMPSP)

The decentralized RCMPSP was introduced by [CGR07] and later extended by [Hom07]. This variant of the project scheduling problem is suitable for modeling situations where multiple projects need to be executed simultaneously. In the case of RCMPSP, the activities of projects are bound by adding two dummy activities to an artificially created project: "start" and "end". They obey resource and precedence constraints, and a common objective is a minimum duration for the multi-project schedule.

2.2.4 The multi-mode resource-constrained project scheduling problem with generalized precedence relations (MRCPSP-GPR)

As mentioned earlier, due to the need to model complex real-world scheduling cases with different objectives, there are several variants of project scheduling problems. One of these variants is the resource-constrained project scheduling problem with minimum (MRCPSP/min) and maximum (MRCPSP/max) time lags (MRCPSP/max) or otherwise known as MRCPSP with generalized precedence relations (MRCPSP-GPR). According to the definition of this problem, time windows (minimal and maximal time lags) between the start and the finish of the pair of activities can be classified into one of four categories: start-finish (SF), finish-start (FS), start-start (SS) and finish-finish (FF) (see [HDDR99] for more details). [BMR88] propose a way to represent all time lag types through one type.

2.2.5 The multi-mode resource-constrained project scheduling problem with discounted cash flows (MRCPSPDCF)

MRCPSPDCF is considered when dealing with the project scheduling problem where the cash flow is an important aspect. Positive cash flow is essential for most projects in the industry. Therefore, the main objective for this category of problems is the maximization of the net present value (NPV). The main methods dealing with this type of problem are categorized into optimization-guided, hybrid, parameter-based, and meta-heuristic approaches (see [USŞŞ01, KP97] for more details).

2.2.6 The stochastic resource-constrained project scheduling (SRCPSP)

SRCPSP is another extended variant of RCPSP inspired by project management. While in RCPSP the resource requirements and resource availability are known at the time the activities are scheduled, SRCPSP represents a class of scheduling problems where many parameters of the problem may be uncertain, e.g., the duration of some activities is not known or the amount of available resources is not known at the time the project starts.

2.2.7 The mode-identity and resource-constrained project scheduling problem (MIRCPSP)

MIRCPSP belongs to the group of less treated project scheduling problems. It was first introduced by [SSD97] as a generalization of the multi-mode variant MRCPSP. In MRCPSP, all project activities are divided into disjoint subsets, and all activities within each subset are executed in the same mode.

2.2.8 The resource-constrained project scheduling problem with multiple crashable modes (RCPSPMCM)

RCPSPMCM has also not been extensively studied. It involves the time-cost tradeoff problem, which means that the duration of a given execution mode can be reduced at a given cost. RCPSPMCM was first introduced by [AE98].

2.2.9 The test laboratory scheduling problem (TLSP)

Another extension of RCPSP that comes from industry needs in test laboratories is TLSP, which was presented in [MM18]. In TLSP, a set of jobs consisting of multiple tasks with similar characteristics must be scheduled given various time and resource constraints.

2.2.10 The multi-mode resource-constrained multi-project scheduling problem (MRCMPSP)

A more general form of the project scheduling problem with a reasonable practical significance was introduced through the MRCMPSP. It extends the RCPSP in several aspects. MRCMPSP deals with simultaneous scheduling of a set of multiple projects taking into account the availability of local and global resources under different time and resource constraints. It has practical importance, especially in the construction [KREK19] and production sectors [WKS+16]. Research on the MRCMPSP problem increased with the organization of the MISTA challenge 2013 (see [WKS+16] for more details). MRCMPSP, introduced in this challenge, extends RCPSP as follows:

- MRCPSP - the activities of a single project can be executed in multiple modes,

- RCMPSP - multiple projects share global renewable resources while they must be scheduled simultaneously.

MRCMPSP is a more realistic representation of the real-life environment in which many projects that must be executed simultaneously compete for shared limited resources. Every project is comprised of a set of non-preemptive activities that can be executed in one of several predefined modes. The execution mode of an activity has its specifics in terms of allocated resources and time duration. Moreover, the activities of each project are interrelated by precedence constraints. Each project has its release date and an associated set of different local renewable and non-renewable resources. In addition, there is a set of global renewable resources that are shared among all the projects. The goal is to find an optimal schedule of activities with respect to specific objectives that satisfies various constraints such as time, precedence, and resource constraints. As mentioned earlier, RCPSP belongs to the class of NP-hard optimization problems [BLK83]; therefore, MRCMPSP is also at least one NP-hard problem. Additionally, generating a feasible schedule concerning non-renewable resource constraints, in the case when activities may require more than one non-renewable resource, has been proved an NP-Complete problem itself [KD97b].

2.2.11 MRCMPSP - Problem definition

The MRCMPSP, as a more generalized form of the project scheduling problem, is a more accurate representation of the real-life environment. The MRCMPSP problem is comprised of a set of n projects: $P = \{1, 2, .., n\}$ and every project $i \in P$ is comprised of activities J_i that are executed in more than one of the modes taking into account different shared resources, time and precedence constraints. In addition, every project has a release date r_i, i.e., the earliest time when its activities could start. Every activity $j \in \{1, 2, .., |J_i|\}$ of every project has to be scheduled, i.e., its starting time s_{ij} has to be defined considering all constraints. The first and last activities of projects are dummy activities with only one execution mode, a duration equal to zero, and no resource requirements. There are sets of renewable and non-renewable resources:

$$L_i \in \{1, ..., |L_i^\rho|, |L_i^\rho| + 1, ..., |L_i^\rho| + |L_i^v|\} \tag{2.1}$$

where $L_i^\rho \in \{1, ..., |L_i^\rho|\}$ indicates renewable resources and $L_i^v \in \{|L_i^\rho| + 1, ..., |L_i^\rho| + |L_i^v|\}$ non-renewable resources.
All non-renewable resources have fixed capacities h_{il} for the whole project duration, $\forall \quad l \in L_i^v$ and $i \in P$. Renewable resources have a fixed capacity c_{il} per time unit, $\forall \quad l \in L_i^\rho$ and $i \in P$. There are local renewable resources dedicated to a specific project only and global renewable resources (G^ρ) that are shared among all the projects. The availability of global renewable resources is limited by c_g^ρ, $g \in G^\rho$. There are no global non-renewable resources. Every activity has more than one available execution mode. An activity's execution mode defines the time duration required to complete the activity

and its specific resource requirements. $M_{ij} \in \{1, ..., |M_{ij}|\}$ and d_{ijm} (duration of activity $j \in J_i, i \in P$ in mode $m \in M_{ij}$) define execution modes of activities. Moreover, r^{ρ}_{ijml}, r^{v}_{ijml}, and r^{ρ}_{ijmg} determine the requirements for local renewable, non-renewable , and global renewable resources, respectively, when activity $j \in J_i, i \in P$ is processed in mode $m \in M_{ij}$. Feasible projects schedules must always satisfy the following hard constraints:

- For every local non-renewable resource $l \in L^v_i$ dedicated for every project $i \in P$, its total consumption cannot exceed its capacity c_{il}:

$$\sum_{j \in J_i} \sum_{m \in_{M_{ij}}} y_{ijm} r^{\nu}_{ijml} \leq c_{il} \quad \forall i \in P, l \in L^{\nu}_i, m \in M_{ij}, \tag{2.2}$$

  where $\sum_{m \in_{M_{ij}}} y_{ijm} = 1, \qquad y_{ijm} \in \{0, 1\}$

- For every local renewable resource $l \in L^{\rho}_i$ dedicated for every project $i \in P$, its total consumption at the time unit t cannot exceed its capacity c_{il}:

$$\sum_{j \in J_i} \sum_{m \in_{M_{ij}}} x_{ijt} y_{ijm} r^{\rho}_{ijml} \leq c_{il} \quad \forall i \in P, l \in L^{\rho}_i, m \in M_{ij}, t \in [0, T], \tag{2.3}$$

  where $x_{ijt} \in \{0, 1\}$

- For every global renewable resource $g \in G^{\rho}$, its total resource consumption at the time unit t cannot exceed its capacity c^{ρ}_g:

$$\sum_{i \in P} \sum_{j \in J_i} \sum_{m \in_{M_{ij}}} x_{ijt} y_{ijm} r^{\rho}_{ijmg} \leq c_g \qquad \forall g \in G^{\rho}, t \in [0, T] \tag{2.4}$$

- Particular activities may require the completion of other activities before they start. In that case, feasible schedules must fulfill all precedence constraints between such activities, i.e. if activity $j \in J_i$, which is executed in mode m, must precede activity $j' \in J_i$:

$$\sum_{m \in_{M_{ij}}} s_{ij} + y_{ijm} d_{ijm} \leq s_{ij'} \qquad if j \prec j' \tag{2.5}$$

- Release time of every project is respected, i.e., for each activity $j \in \{1, 2, .., |J_i|\}$ of every project $i \in P$, its start time $s_{ij} \geq 0$ and $s_{ij} \geq r_i$.

The objective is to find a feasible schedule with minimum total project delays (TPD) and schedule total makespan (TMS). TPD is the primary objective, and TMS is used as a tiebreaker. Project delay of a project i is defined as the difference between Critical Path Duration (CPD), a theoretical lower bound on the earliest finish time of the project, and the actual project duration (makespan):

$$PD_i = MS_i - CPD_i \tag{2.6}$$

MS_i - makespan of project i is calculated as difference:

$$MS_i = f_i - r_i \tag{2.7}$$

f_i - finish time of project i,
r_i - the release date of project i,

Total project delay is calculated as:

$$TPD = \sum_{i=1}^{n} PD_i \tag{2.8}$$

n - number of projects.

Total makespan is the duration of the complete multi-project schedule:

$$TMS = \max_{i \in P} (f_i) - \min_{i \in P} (r_i) \tag{2.9}$$

Both soft constraints are combined into a single objective function as following:

$$F = \alpha * TPD + TMS \tag{2.10}$$

where value $\alpha = 100,000$ (similar objective function was used in the MISTA 2013 challenge).

2.3 Representation of project scheduling problems

Two main representations of project networks have been considered in the literature, AON and AOA [KW59, MRCF59], corresponding to an activity-based or event-based representation, respectively.
In the AON representation, a graph node represents a project activity with all its attributes, and a directed arc between two nodes represents a precedence relation between two activities represented by those nodes. This type of representation is more commonly used in the case of project scheduling problems with makespan objectives. The first and last activities of the project are modeled as dummy activities to indicate the start and completion of the project and to maintain the precedence relationship. An example of the AON directed graph for the RCPSP problem with makespan objective can be seen in Figure 2.2.

The AOA representation is typically used in situations where activities are associated with cash flows, e.g. the maximization of the net present value. In the case of this representation, a graph node represents an event, and an arc represents an activity. Dummy events denote the start and completion of the project, and dummy activities maintain the precedence relationship. This representation requires some preliminary

clarifications for proper interpretation when the objective includes cash flow. If some activities end at or start at a particular node, it may not be clear which activities are the cash flow accompanied with, or whether the cash flow at an event is the balance of payments and expenditures. An example for an AOA directed graph can be seen in Figure 2.4 [Rus70].

Example 1: Given this example, let us assume that:

- P_j, P_2, P_3, P_4 and P_5 are activities (arcs) with corresponding durations d_i $(i = 1, .., 5)$,
- e_1, e_2, e_3 and e_4 are four events (nodes) with corresponding occurring times $t_j \in T$, where $T = \{0, 2, 8, 11\}$, and accompanied with net cash flows $f_j \in F$, where $F = \{0, -5000, 3000, 3000\}$.

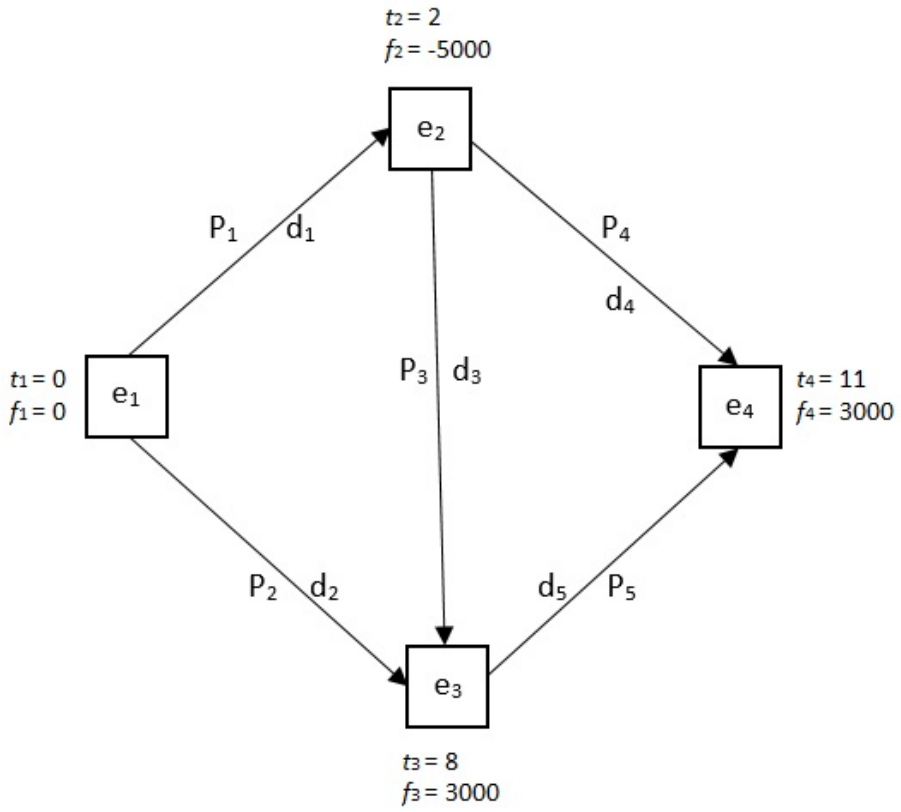

Figure 2.4: An example of the AOA representation with cash flow [Rus70]

In more constrained situations of the project scheduling problems with cash flow objectives, advance payments may be not allowed, or on the contrary, some of them must occur

much earlier to complete the activities on the schedule. If the AOA model is used for the problem in these situations, it should be adjusted to properly link expenditures to activities. A variant of the AOA model introduced by [PSDSD97] supports modeling of cash inflows (f^+) and outflows (f^-). Thus, this AOA model resolves the representation of the above-mentioned constrained problem by adding more dummy activities and increasing the complexity of the problem. Figure 2.5 shows an example of such a model [PSDSD97].

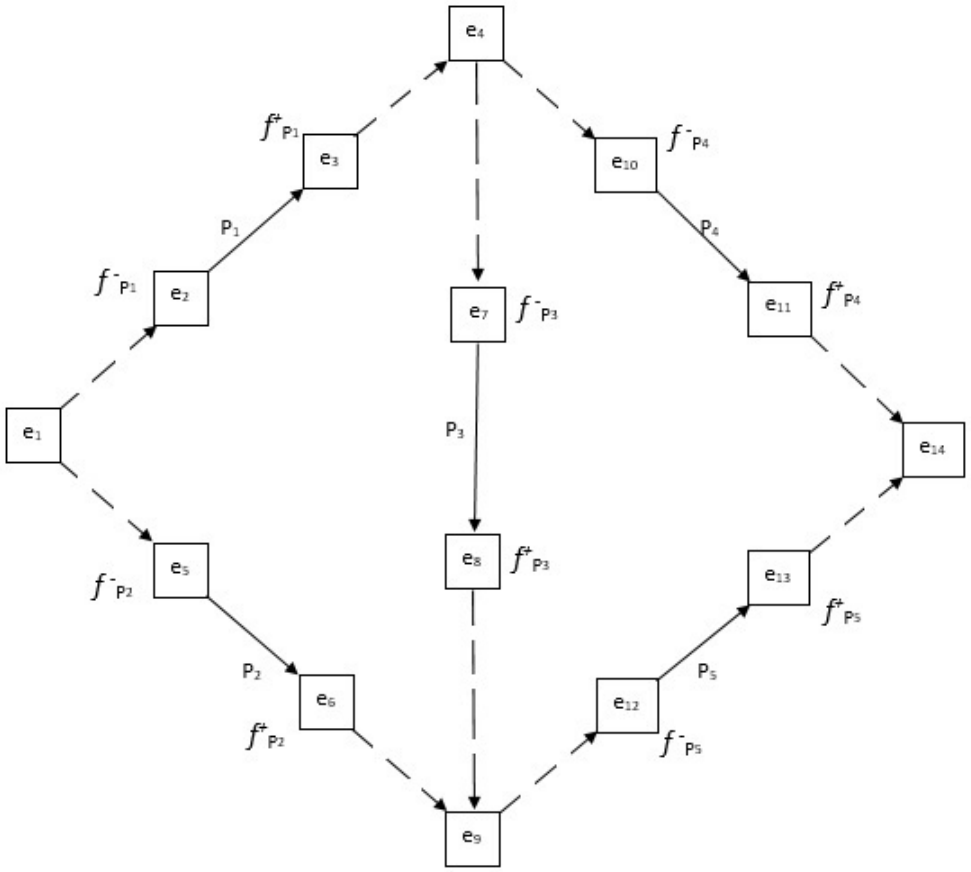

Figure 2.5: An example of the AOA representation with cash inflows (f^+) and outflows (f^-) [PSDSD97]

The AON representation can also be used for project scheduling problems with the cash flow objective, but the expenses for each activity should be known in advance, and payments are conducted for completed activities. In the case of the AOA, expenditures can be associated with many activities that start or end with one event and represented as total expenditures for that event. A hybrid model that intertwines the advantages of AON and AOA was used by [ZP99] for RCPSPCF. Some assumptions are made for this model: payments and expenditures (cash inflows f^+ and outflows f^-) for each activity

are known in advance, payments occur when the activity is completed, and expenditures occur at the start of the activity. An example of such a model is shown in Figure 2.6.

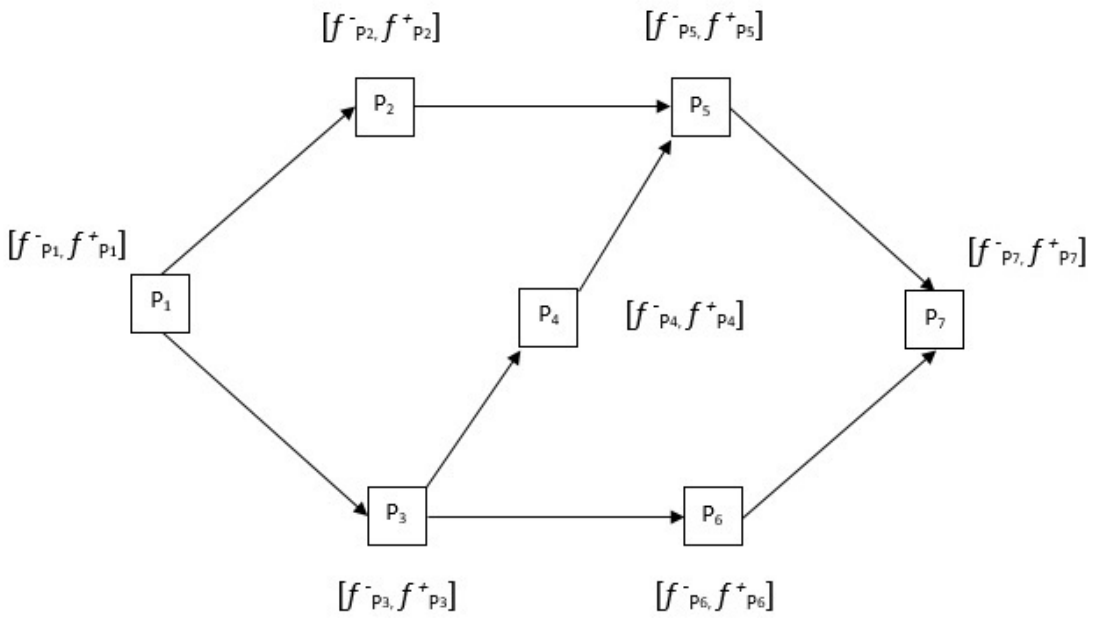

Figure 2.6: An example of the AON representation with cash inflows (f^+) and outflows (f^-) [ZP99]

2.4 Classification and notation scheme

In the last decades, various authors have published various surveys to standardize the terminology, notations, definitions, classifications, and efforts to solve project scheduling problems [KP97, HDDR99, BDM$^+$99, HB10, WJMW11, VC18, biT20]. In order to avoid misinterpretations and to use standard notations for describing project scheduling problems, [GLLK79] and [BLK83] proposed a so-called $\alpha/\beta/\gamma$ classification scheme. This scheme was extended by [HDDR99] and later also by [BDM$^+$99].

α is used to distinguish project scheduling problems and their resource requirements (see [BDM$^+$99] for more details).

PS - Project scheduling,

MPS - Multi-mode project scheduling,

PSm, σ, ρ - project scheduling problem with m resources, σ represents units of each available resource, ρ represents the maximal number of resource units required by each activity,

MPSm, σ, ρ; μ, τ, ω - multi-mode project scheduling problem with m resources, σ represents units of each available resource, ρ represents the maximal number of resource units required by each activity, μ represents non-renewable resources, τ units of each available resource, ω represents the maximal number of resource units required by each activity.

β describes activity characteristics.

$p_j = 1$ - duration for each activity is equal to one,

$p_j = sto$ - stochastic duration for each activity,

d - the deadline for the project duration,

prec - activities precedence constraints,

temp - general temporal constraints given by minimum and maximum start-start time lags between activities.

γ depicts objective function.

$\sum C_j$ - activities' total completion time,

C_{max} - maximal activity completion time,

$\sum c_j^F \beta^{C_j}$ - net present value (c_j^F is cash flow and β is discount factor),

etc.

According to this classification scheme, RCPSP would be classified as $PS|prec|C_{max}$, which means that the goal of this problem is to minimize the makespan of the project considering priority and resource constraints.

2.5 Benchmark data sets

Various solution methods for project scheduling problems from the classes of exact methods, meta-heuristics, and hyper-heuristics were proposed. It was necessary to evaluate and compare the performance of these methods using benchmark instance sets. Several benchmark instance generators have been proposed in the literature based on different parameters and problem models. [KSD95] introduced a generator named ProGen to generate multiple instance sets for RCPSP and its more generalized variant, the multi-mode resource-constrained project scheduling problem. [KS97] significantly enriched these sets with additional benchmark instances known as the project scheduling problem library project scheduling problem library (PSPLIB). [Sch98] extended ProGen for MRCPSP to a version named ProGen/max, which includes minimal and maximal

time lags, and supports three types of objectives: project duration, resource leveling, and NPV. Another instance generator named RanGen, which provides a wider range of parameters than ProGen, was introduced by [DVH03] and further enhanced by [VCD+08] to version named RanGen2 by adding new topological indicators. Moreover, [VPV14] extends the instance generator developed by [VCD+08] for projects consisting of instances with multiple modes using two repair procedures. The extended generator generated three sets of MRCPSP instances known in the literature as (multi-mode instances library (MMLIB)). [CV20] proposes a new three-stage approach that converts RCPSP instances to very hard-to-solve instances in optimality.

The approaches introduced in this thesis were tested with different variants of project scheduling problems, including MRCMPSP instances introduced in the Mista 2013 Challenge, MMLIB instances introduced by [VPV14], and RCMPSP instances from multi-project scheduling problem library (MPSPLIB), introduced by [Hom07]. MRCMPSP instances were generated through by combining multiple MRCPSP instances from PSPLIB.

2.6 Literature review

Over the decades, different solution strategies have been proposed to address different variants of resource-constrained project scheduling problems. In general, we can categorize the solution approaches for project scheduling problems as exact methods, heuristic, hybrid, meta-heuristic and hyper-heuristic approaches. Some traditional approaches, like the critical path method (CPM), and program evaluation and review technique (PERT), have several limitations in terms of project activity scheduling. They are applied to only one project at a time and assume that unlimited resources are available.

RCPSP - is one of the most analyzed variants in the literature. A branch and bound algorithm for the RCPSP was proposed by [BKST98]. They reported good results on problem instances comprised of 30, 60, and 90 activities generated by [KSD95] and [KS97]. Their approach solved instances with 60 activities within one hour. Other competitive exact approaches for solving RCPSP are proposed by [Spr96], [DH97], and [MMRB98]. [DMaBL15] proposes a hybrid greedy and genetic algorithms for RCPSP with the objective of minimizing the project makespan. Another hybrid meta-heuristic solution approach that combines genetic and local search algorithms to solve the RCPSP is proposed by [DKG13].

Some other early approaches to solving RCPSP include meta-heuristics and heuristic approaches that employ rule-based priority scheduling. Heuristic approaches are faster than exact methods and offer close to optimal solutions even for complex situations and large-size projects. As one of the earliest solution approaches for RCPSP, priority rule-based scheduling heuristic consists of two components: a schedule generation scheme and a priority rule. Two schemes are known in the literature: the serial scheduling scheme and the parallel scheduling scheme (for more details, see [Kol96]). Priority rules

determine which activity from the decision set (i.e., a subset of activities to be assigned) is assigned next to a partial schedule (i.e., a schedule comprised of a subset of assigned activities).

The priority rules are mainly categorized as activity, mode, project, or resource priority rules. Some of the well-known priority rules investigated in the literature are the earliest start time, the earliest finish time, the latest start time, the latest finish time, maximum or minimum slack first, and so forth. Some of the mode priority rules are randomly chosen mode, minimum duration mode, stochastic construction method for mode selection, maximum duration mode, minimum resource demand mode, and so forth. There are also priority rules based on penalties due to project delays: maximum duration penalty, maximum penalty, maximum total duration, and so forth. Some of the approaches from this class of heuristics are presented by [BOC93, KD97a, KDH00].

MRCPSP - is also a well-studied variant of the project scheduling problem. One of the earliest exact approaches to solving MRCPSP is linear programming proposed by [Sł81]. Furthermore, [HD98] implements a new branch and bound algorithm optimized with ten bounding criteria and compare it with other existing branch and bound approaches. Another branch and bound algorithm for RCPSP and MRCPSP was proposed by [Spr96]. [DG93] introduces a stochastic scheduling method for MRCPSP.

In addition, various meta-heuristic methods are applied to solve RCPSP and MRCPSP. The approach proposed by [SSW94] is based on a so-called multi-objective project scheduling (MPS) model, which includes three kinds of heuristics: parallel priority rules, simulated annealing and branch-and-bound. [BOC96a] introduced a simulated annealing algorithm for MRCPSP and RCPSP variants. It is capable to handle different objectives like the minimization of project makespan, the maximization of net present value and the minimization of the project cost. Another approach that consists of two simulated annealing variants is proposed by [JMR$^+$01]. Both variants, simulated annealing without penalty function and with penalty function, apply three neighborhood operators: activity shift, activity mode change and the combination of both. The variant with penalty function considers resource-related infeasible solutions. The simulated annealing approach proposed by [BL03] applies two neighborhood exploration techniques, the so-called "activity neighborhood" and "mode neighborhood" explorations, in two stages. A scatter search procedure with three improvement methods for MRCPSP was proposed by [PV11]. The improvement procedures are based on the scarcity values of renewable and non-renewable resources presented in a scarceness matrix. A hybrid scatter search method for MRCPSP was proposed by [RDK09]. The solution approach proposed by [CV11] divides the problem into two steps: the mode assignment and the single-mode project scheduling step. The former step is solved using an SAT solver and the latter using the decomposition-based genetic algorithm of [DV07]. [Zam19] proposed a four-layer heuristic, where the first layer provides an initial solution and the rest of the layers iteratively improve the solution using different strategies. A hybrid solution approach that combines mixed integer programming with an adaptive large neighborhood

search is proposed by [GSF17]. [SH17] introduced an exact solution method based on constraint programming. Other proposed solution approaches for MRCPSP include genetic algorithm, e.g., [Har98, Har01, VPV10, Ozd99, LTCB09, MT97, AMR03, EF10], tabu search, e.g., [BBK99, NI02], ant colony optimization, e.g., [CHW08], and so forth.

RCMPSP - some of the pioneering work on the multi-project scheduling problem started with the RCMPSP problem using a heuristic based on priority rules, e.g., [Fen67] presented a parallel scheduling algorithm for multi-project instances comprised of three and five projects, and the efficiency of three measurement variables is analyzed: project slippage, resource utilization, and in-process inventory. The minimum slack first (MINSLK) priority rule provides the best results considering the three measurement variables mentioned above. [PWW69] introduced zero-one (0-1) linear programming addressing multi-project and job shop scheduling problems. In this work, the optimization of three objectives is analyzed: minimization of total throughput time for all projects, minimization of the time by which all projects are completed (minimization of make-span), and minimization of total lateness or lateness penalty for all projects. Scheduling should meet the following constraints when imposed: limited resources, precedence relations between jobs, job splitting possibilities, project and job due dates, substitution of resources to perform jobs, concurrent and non-concurrent job performance requirements. A typical example of a heuristic based on priority rules for the RCMPSP problem is presented by [KD82]. They generated multi-project instances where projects contain from 34 to 63 activities, and the resource requirements for each activity are from two to six units. Moreover, they introduce six new priority rules; they also show that the priority rules maximum total work content and shortest activity from the shortest project are the best algorithms to schedule multi-projects when the objective is to minimize the mean project delay. This work was extended to a multi-project approach by [KN85]. They introduced penalties based on project delays and analyzed a multi-project problem comprised of three projects containing 24 to 33 activities for small problems and 50 to 66 activities for large ones. Their computational experience shows that the maximum penalty priority rule is the best algorithm when minimizing the sum of the project weight delays. [Hom07] proposed a restart evolution strategy (RES) for RCPSP embedded in a multi-agent system for solving MPSPLIB RCMPSP. A scheduling agent executes a RES procedure to schedule activities and negotiate resource allocations for each project. A modified (μ, λ) coordination mechanism from evolution strategies used in a multi-agent system is proposed in [Hom12]. [RPSN21] introduced a solution method for RCMPSP combining priority rules and a genetic algorithm.

MRCPSP-GPR - [DH99] introduced a local search method using a tabu search procedure for solving MRCPSP-GPR. Their method executes in two consecutive phases. First, it assigns execution modes to all activities, and consequently, the problem is solved as a resource-constrained project problem with fixed execution modes. [SSH08] presented an exact method for solving MRCPSP-GPR, and [BN96] presented two types of heuristics for solving RCPSP/max for two objectives: the resource-leveling and minimum project

duration. [SH16] propose three mathematical models in their exact solution approach for MRCPSP-GPR.

MRCPSPDCF - in their approach, [IE94] propose a tabu search procedure for scheduling project activities with cash inflows and outflows with the objective of maximizing NPV. [CZC+10], initially formulated MRCPSPDCF as a graph-based search problem and then applied the ant colony optimization (ACO) procedure to solve it. A GA algorithm for RCPSPDC and MRCPSPDC for three payment models is proposed by [LV16]. [EAM+19] proposed a hybrid method that combines mixed-integer non-linear programming with a genetic algorithm and multi-objective simulated annealing algorithm for a real-life conditions problem that considers renewable and non-renewable resources. The main objectives for the problem are to maximize the NPV and minimize the project completion time. Another hybrid method aimed at optimizing cash flow in a resource-constrained multi-project environment is proposed by [HHW21]. This method combines a non-linear integer programming optimization model with tabu search, simulated annealing and a combination of the latter two meta-heuristics under project deadline and renewable resources constraints.

SRCPSP, RCPSPMCM, and MIRPCPSP - belong to a group of less analyzed project scheduling problems in the literature. [RCL18] introduced a new class of generalized preprocessor (open-loop) policies for the SRCPSP problem. [LW15] proposed a dynamic (closed-loop) policy that insists on selecting the best set of starting activities at each decision point.

[AE98] introduce a two-stage heuristic procedure for solving RCPSPMCM. In the first stage, a feasible solution is generated, and in the second stage, the same schedule is improved through six improvement rules. Furthermore, an exact solution method for RCPSPMCM is proposed in [EAC01]. [SSD97] proposes a randomized approach combining static and dynamic priority rules for solving MIRCPSP.

TLSP - A local search framework consisting of min-conflicts and simulated annealing was applied as a solution approach for TLSP and its sub-problem TLSP-S. TLSP-S has a fixed list of jobs. In [MM21], an instance generator for TLSP is presented, and the approach is validated against two generated instances and three real-world instances taken from an industrial company's laboratory. This work is further extended in [MMS21] by introducing four new neighborhoods, which are combined with the existing ones and alter the grouping of tasks of a schedule. Constraint programming and hybrid approaches for these problems are presented in these works [GMM19, DGMM20, GMM21].

MRCMPSP - was not addressed in the literature until the early 1990s. [SV93] introduced an integer programming model for solving MRCMPSP. Their multi-criteria model is comprised of the planning and scheduling stages. [Tse08] extended their work in [MT97] for MRCMPSP. In addition to a genetic algorithm for the MRCMPSP, another

parallel scheduling algorithm is developed to find a close-to-optimal solution. The latter one serves as a baseline to validate the performance of the first algorithm, and includes a combination of an activity and mode priority rule. According to [LMT00], multi-project scheduling can be addressed from two perspectives: single-project approach and multi-project approach. In the first case, the activities of the projects are considered independently, whereas in the second case, the activities of the projects are bound into an artificially created project by adding two dummy activities. They have developed a multi-criteria heuristic algorithm that takes into consideration time and non-time aspects in order to improve resource allocation in multi-project scheduling: mean project delay or multi-project duration increase, project splitting, in-process inventory, resource leveling, or idle resources. Their multi-criteria heuristic algorithm for multi-project scheduling is comprised of two phases. In the first phase, the algorithm generates a good schedule for multi-project with one of the time criteria used: mean project delay or multi-project duration increase. In the second phase, the initial schedule from the first phase is improved with one of the non-time criteria: project splitting, in-process inventory, resource leveling, or idle resources. Moreover, they validated their computational work through data processing from a survey organized in the Valencian Region, Spain, with about 1000 surveys sent to small and medium-sized companies.

The MISTA challenge 2013 has boosted research efforts for MRCMPSP. Several researchers have been working on it over the past few years. [AKK+16] applied a combination of hyper-heuristics and Monte-Carlo Tree Search (MCTS). Their two-phase (construct and improvement phase) approach has as its primary objective the minimization of total project delay (TPD) and total makespan (TMS) of projects, with TMS serving as a tiebreaker. According to [AKK+16], reasonable solutions have a proximate order of the projects. Therefore, the construction phase in this approach has to do with a creation of an initial solution. In this phase, the initial solutions are attempted with a partial structure that resembles the good solutions. Since the main objective of this problem, minimization of the total project delay (TPD), promotes solutions comprised of activities grouped by projects, and in order to improve the performance of the improvement phase, a version of the monte-carlo tree search (MCTS) search method is used to classify projects into so-called "start", "middle", and "end" partitions in terms of total project time. In the improvement phase, a memetic algorithm is used with a local search procedure based on hyper-heuristics that control a large set of neighborhood moves. In order to increase the overall performance, the population size was kept small so that one CPU core can be used for each local search. [Gei17] proposed an iterated local search and a variable neighborhood search with four neighborhoods. This approach starts from an initial feasible solution, generates neighbors through four neighborhood operators, and tests them for acceptance. Perturbation persists on random change of execution mode of an activity followed by a mode-repair-procedure. [TSCS16] proposed a hybrid search method that combines a parallel local search with multiple neighborhoods with integer programming. [KREK19] compared two solution approaches, single-project and multi-project, for MRCMPSP with multiple objectives. They use different heuristic rules and an integer programming model to optimize time, cost and quality in a multi-

project environment. Their approach is evaluated using PSPLIB's benchmark instance library and several real-life case study projects from the construction sector. A two-stage optimization method was proposed by [KM20] using a genetic algorithm and a simulated annealing algorithm for MRCMPSP. Automatic design of a hybrid iterated local search was considered by [LIMMS13]. A stochastic local search procedure with two neighborhoods was proposed In [BBZ+13]. [BBU15] implemented a genetic algorithm for a multi-project environment with projects with assigned due dates and a resource dedication policy. In [KZV+16] a genetic algorithm was implemented for mode assignment and a priority rule heuristic for job selection in MRCMPSP. Other methods proposed for this problem include large neighborhood search (LNS), which is used in combination with mixed integer programming (MIP) method introduced in [AH13]. [APPR13] introduced a three-phased solver. In the first phase, feasible solutions are generated for each project and combined into a unique feasible solution for the entire problem. In the second phase, simulated annealing is employed to improve the solution, and in the third phase further improvement is performed using tabu search and an extra neighborhood operator that changes two modes. [BDMX13] implemented a solver based on a combination of an evolutionary approach and local search. The local search uses three neighborhood operators and serves as a post-processing procedure for the evolutionary algorithm.

CHAPTER 3

Combining min-conflicts heuristic with iterated local search (ILS) for MRCMPSP

In this chapter, we present our solution approach for MRCMPSP, which includes several innovative ideas such as a new min-conflicts heuristic and new neighborhood operators. According to the definition, the min-conflicts heuristic considers a set of variables in conflict, randomly selects a variable from the set, and assigns a value to that variable that minimizes the conflict [MJPL92]. In our solution approach, the variables in conflict are activities that compete for the same resources at a given time t of the schedule (see illustration in Figure 3.1). A randomly selected activity is assigned the mode or the position on the schedule (depending on the neighborhood operator) that takes into account the resource constraints and provides the best objective function value. In order to prevent getting stuck in a local minimum and avoid cycles, we implemented two types of tabu lists. One tabu list keeps track of the most recent activities whose modes have been changed, and the other the tracks of activities whose positions on the schedule have been changed.

Furthermore, we investigate the iterated local search (ILS) framework for MRCMPSP, where the embedded local search is comprised of the min-conflicts heuristic and new neighborhood operators. Several extensions and improvements for this solution approach are introduced and tested. The ILS framework is described in [LMS03] and consists of four main components: initial solution, embedded local search, acceptance criterion, and perturbation (see Algorithm 3.1).

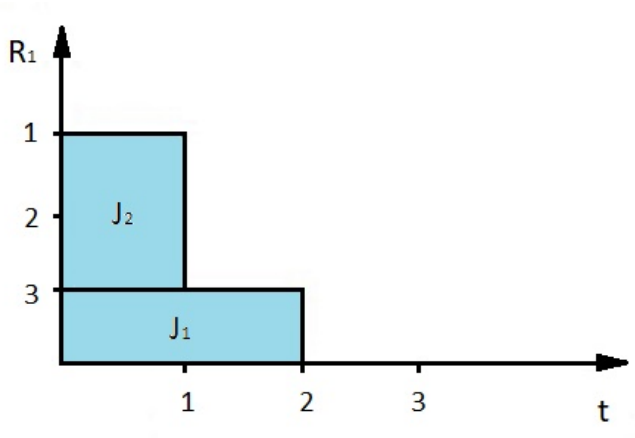

Figure 3.1: Activities in conflict for resources

Algorithm 3.1: Method ILS

```
s0 = InitialSolution();
s1 = s0;
do
    s2 = LocalSearch*(s1);
    s2 = AcceptanceCriterion(s1, s2);
    s1 = Perturbation(s2);
while termination-condition = true;
```

3.1 General description of our algorithm

In the following sections, we review the individual components of our solution approach (MinCONv1) for MRCMPSP. It is comprised of the initial solution generator, the local search (a combination of min-conflicts heuristic and several neighborhood operators), the acceptance criterion of a solution, and the perturbation strategy.

3.1.1 Schedule representation

Schedule alternatives are represented as a pair of vectors: $\overrightarrow{S} = \{\overrightarrow{\pi}, \overrightarrow{M}\}$, where $\overrightarrow{\pi}$ is the vector of activities and $\overrightarrow{M}$ is its corresponding mode vector. The activities in $\overrightarrow{\pi}$ are re-arranged as if they were activities of a larger unique project, i.e., if $j \in \{1, 2, .., |J_i|\}$ is an activity of project i, where $i \in$ P and P a set of n projects, P $= \{1, 2, .., n\}$, then:

$$\overrightarrow{\pi} = \{1, 2, .., |J_1|, |J_1| + 1, .., |J_1| + |J_2| + .. + |J_n|\} \tag{3.1}$$

Similar representations were used by [AKK+16], [TSCS16] and [Gei17]. A schedule solution alternative is constructed using the serial scheduling scheme (SSS), which

constructs a schedule by sequentially placing activities into a schedule as early as possible. This approach was also used by other researchers ([AKK+16], [TSCS16] and [Gei17]).

3.1.2 Initial solution

In the first phase, a schedule is constructed that represents the relaxed form of the problem, i.e., assigning activities to modes that result in minimal project delay for each project and disregard resource constraints. The schedule in this phase is feasible considering precedence constraints of the activities. Construction is a two-stage process:

- 1st stage - the min-conflicts heuristic is used to assign feasible modes to activities that take into account non-renewable resource constraints. In this case, the set of variables consists of the activities that are in conflict for non-renewable resources, and the domain of their values are their modes. A variable is randomly selected from this set and assigned a mode that minimizes the amount of non-renewable resources demanded by activities involved in the conflict.

- 2nd stage - the scheduling scheme is constructed respecting the renewable resources and the precedence constraints. The serial scheduling scheme is applied to build a feasible schedule by assigning activities into a schedule sequentially, one activity at a time, as early as possible.

Then, the feasible solution obtained is improved by our "two modes changes" neighborhood operator to achieve a faster convergence to local optima. The whole procedure is described in Algorithm 3.2 and Algorithm 3.3.

Neighborhood operators can be broadly grouped into those that change the modes of activities and those that change the positions of activities within a schedule. As described by [Gei17], neighborhood operators that change the modes of activities have a greater impact on the schedule than other operators. The improved schedule is passed to the local search heuristic for further improvement.

3.1.3 Neighborhoods and Local Search

We propose a local search heuristic based on a combination of the min-conflicts heuristic and multiple project-wise neighborhood operators. This technique operates in the feasible search space, i.e., assignments of modes of activities that violate non-renewable resource constraints are rejected. Several neighborhood operators used here are based on previous works ([AKK+16], [TSCS16], [Gei17], [AH13], etc).

We introduce three new neighborhood operators: Clone-project (CloneProj), Clone-project-partially (CloneProjPart) and Clone-sequence (CloneSeq). These neighborhood operators are evaluated experimentally by running the algorithm with and without them. More on this later in this chapter.

Algorithm 3.2: First stage - Initial solution construction

1 Remove unfeasible modes;

Input: $\overrightarrow{S'} \in \{\overrightarrow{S}\}$, *feasible schedule concerning activity precedence constraints*; *Modes(j)*, the set of execution modes of j; d_{jm}, the duration of activity j in execution mode m;

Output: Feasible initial solution regarding non-renewable resources and precedence constraints;

2 **foreach** *activity* $j \in \overrightarrow{\pi'}$ **do**

3 Find execution mode of j that provides minimal duration, $d_{jm} = \min_{k \in Modes(j)} d_{jk}$;

4 **end**

5 **repeat**

6 **if** $\overrightarrow{S'}$ *is feasible* **then**

7 **return** $\overrightarrow{S'}$

8 **end**

9 Chose a variable (activity) randomly from the set of conflicting variables;

10 Chose the value for the variable that minimizes conflicts;

11 **until** $\overrightarrow{S'}$ *is feasible regarding non-renewable resources*;

Algorithm 3.3: Second stage - Initial solution construction

Input: $\overrightarrow{S'} \in \{\overrightarrow{S}\}$, *a feasible schedule regarding non − renewable resources and precedence constraints*; r_i, the release date of project i; *Prec(j)* $\subset \overrightarrow{\pi'}$, the set of predecessor activities of j; d_{jm}, the duration of activity j in execution mode m;

Output: Feasible initial solution;

1 **foreach** *activity* $j \in \overrightarrow{\pi'}$ **do**

2 Find the earliest start time of j, $t_j = \max\{r_j, \max_{k \in Prec(j)} t_k + d_{km}\}$;

3 **end**

4 Perform some preliminary improvement using *MinConTMC()* neighborhood operator;

New neighborhood operators

Clone project (CloneProj) - is a project-wise operator that "clones" modes of activities of a given project within the current schedule with modes of activities from the best schedule constructed so far. The activities of a randomly selected project within the current schedule are assigned the same modes of the same activities from the best schedule.

CloneProjPart and CloneSeq - likewise ClonProj, "clones" modes from a sequence of activities of a particular project within the active schedule (CloneProjPart) and a sequence of activities within active schedule (CloneSeq), respectively. In the first case, the size of the sequence is chosen randomly in a range between 40%-50% of the size of a given project, and the start is chosen randomly, taking as a reference point the beginning activity of that project. In the second case, the start of the sequence is chosen randomly at a random point in the schedule, and the size in the range between 15-20% of the size of the whole schedule.

Example: Let $n = 2$ be number of projects, where P1 $= \{1, 2, 3, 4, 5\}$ and P2 $= \{6, 7, 8, 9, 10\}$, $\overrightarrow{S}_{best} = \{P_{best}, M_{best}\}$ the best found schedule, where $P_{best} = \{1, 6, 2, 7, 3, 4, 8, 5, 9, 10\}$ and $M_{best} = \{1, 3, 2, 1, 3, 1, 2, 1, 2, 3\}$, and $\overrightarrow{S} = \{P, M\}$ an active schedule, where $P = \{1, 6, 2, 3, 7, 4, 8, 5, 9, 10\}$ and $M = \{1, 1, 3, 2, 1, 3, 1, 1, 3, 2\}$ as in Figure 3.2.

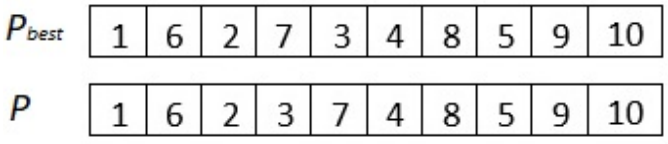

Figure 3.2: Sample of a schedule with two projects, P1 and P2

When *CloneProj*, with *projID = 1* is applied to $\overrightarrow{S}$, it affects activities 1, 2, 3, 4, 5 of P1 (see Figure 3.3) and consequently $M = \{1, 1, 2, 3, 1, 1, 1, 1, 3, 2\}$.

Figure 3.3: Sample after applying *CloneProj* with *projID = 1*

When *CloneProjPart*, with *projID = 1*, *seqSize = 40%* and *startSeq = 2* is applied to $\overrightarrow{S}$, it affects activities 2 and 3 of P1 (see Figure 3.4) and consequently $M = \{1, 1, 2, 3, 1, 3, 1, 1, 3, 2\}$.
When *CloneSeq*, with *seqSize = 20%* and *startSeq = 4* is applied to $\overrightarrow{S}$, it affects activity 3 of P1 and activity 7 of P2 (see Figure 3.5) and consequently $M = \{1, 1, 3, 3, 1, 3, 1, 1, 3, 2\}$.

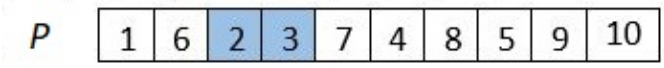

Figure 3.4: Sample after applying *ClonePrejPart* with *seqSize = 40%* and *startSeq = 2*

Figure 3.5: Sample after applying *CloneSeq* with *seqSize = 20%* and *startSeq = 4*

Other neighborhood operators

MinConOMC - changes the mode of a randomly selected activity from a set of variables in conflict for resources. Similar operators have been implemented by [AKK+16], [TSCS16], [Gei17], [LIMMS13] and [BBZ+13]. We apply the min-conflicts heuristic to generate the neighborhood by constructing a set of variables according to activities that conflict at a given time t of the schedule in terms of renewable and non-renewable resources. In our experiments, we tested two strategies for constructing the set of variables in conflict: the set of running activities at the same time t (t is chosen randomly) and a short sequence of activities within the schedule. Although the former option produces slightly better results, we chose the second option due to the performance issue. The size of the tabu list is parameterized and depends on the size of the project. The domain of the values of a conflicting variable is the set of its execution modes. A variable is randomly selected from the set of conflicting variables, and if it is not on the tabu list, it is assigned a mode that minimizes the conflict and it enters the tabu list. MinConOMC is executed until no further improvement is made to the current schedule.

MinConTMC - changes the modes of two activities. A similar operator was implemented by [AKK+16], [TSCS16] and [Gei17]. As in the previous case, we apply the ideas of min-conflicts heuristic to generate the neighborhood by constructing a set of variables consisting of activities that compete for the same type of resources at a given time t of the schedule. In this case, the set of variables includes every pair of activities that are in conflict, and the domain of their values consists of every dual combination of their modes. Obviously, the size of the neighborhood is larger. A couple of variables is randomly selected from the set of conflicting variables and is assigned modes that minimize the conflict.

MinConFMC - this operator changes the modes of four activities. Like in the case of MinConOMC and MinConTMC, we apply min-conflicts heuristic ideas to generate the neighborhood, with the difference that this operator is applied in a very limited neighborhood and only in cases when MinConOMC and MinConTMC do not achieve any further improvement. A similar operator was used in [TSCS16].

MinConSJL - this operator is based on implementations of [AKK+16], [TSCS16], [Gei17], [LIMMS13] and [BBZ+13] and the ideas of min-conflicts heuristic. The min-conflicts heuristic is used to generate the neighborhood by constructing a variable set consisting of all activities found in the sequence from the location of a randomly selected activity to the location of its last predecessor. The domain of values for every variable consists of locations in the schedule from the variable's current position (excluding the variable's current position) to its last predecessor. MinConSJL selects the best swap. A tabu list is used when selecting an activity from the schedule.

MinConSJR - is identical to MinConSJL, except that the variable set is created from all activities found in the sequence from the location of a randomly selected activity in the schedule to the position of its first successor.

Invert subsequence (INVS) - A similar operator is implemented by [TSCS16] and [Gei17]. INVS inverts activities from the location of a randomly chosen activity in the schedule to its last predecessor in our implementation.

Compress project and move to the end (ComE) - is a project-wise operator based on the implementation of [AKK+16] that compresses and moves the activities of a randomly selected project to the end of the schedule.

Compress project and move to the front (ComF) - is a project-wise operator based on the implementation of [AKK+16] and it compresses the activities of a randomly selected project and moves them to the beginning of the schedule.

Partial compress (PCom) - is based on the ideas of [AKK+16] and [TSCS16] and compresses up to 50% of the activities starting from the end of a randomly selected project.

The overall algorithm (MinCONv1) The overall algorithm is presented in Algorithm 3.4. The sequence of the execution of neighborhood operators is adjusted through thorough experimentation. Initially, the project-wise neighborhood operators are applied, executing as long as improvements occur (lines 3 - 10). Then, the neighborhood operators MinConOMC and MinConTMC or MinConFMC, MinConSJL, MinConSJR, and INVS are executed. MinConOMC runs until no further improvement is achieved, and MinConTMC runs for a specific time that depends on the size of the project. In our experiments, MinConOMC and MinConTMC appeared to run most of the time (see lines 11 - 20). The neighborhood operators MinConFMC, MinConSJL, MinConSJR, and INVS, are used only when no improvementcan be achieved with the previous operators (see lines 21 - 23). The application of the tabu list appeared to have a positive impact on the quality of the solution. In our approach, two tabu lists are used, one tabu list that records the last activities whose modes have been changed and the other tabu list that records the activities whose positions on the schedule have been changed. A solution

generated by operators is accepted if it is better than the actual best solution, except for the case of MinConSJL and MinConTMC operators, where a solution of the same or better quality is accepted. In our experiments, we found that accepting equally good solutions contributed to generating promising diverse structures of project schedules. A similar observation was reported by [Gei17] in the case of one of his neighborhood operators.

3.1.4 Acceptance criterion

We have experimented with two variants. In the first variant, only the better solutions are accepted. In the second case, the solution is accepted if it is up to 10% worse than the best solution. In our experiments, better results were obtained with the first variant (lines 27-31, algorithm 3.4).

3.1.5 Perturbation

Due to the significant impact on the schedule, the perturbation strategy is based on the modes' change. We performed a feasible change of modes of randomly selected activities, while the subsequent configuration of modes remains feasible. The perturbation size, which is the number of activities (randomly selected) whose modes are changed, was important to our results. For some instances, e.g., A7, B3, B7, X1, X8, we obtained better results with perturbation size of 1 (1% of the size of the instance), but for some other instances, e.g., A10, B10, X7, X10, we got better results with perturbation size of up to 10% of the size of the instance. Therefore, we applied an adaptive perturbation strategy where the perturbation size is gradually increased in each iteration (up to 10% of the size of the actual instance) if the solution is not improved. When the solution is improved, the perturbation size is reset to 1 (lines 33-38, Algorithm 3.4).

3.1.6 Parameter tuning

As mentioned earlier, we have applied two tabu lists: one tabu list for operators that manipulate modes and one for operators that manipulate positions of activities in the schedule. The sizes of these tabu lists depend on the total number of activities in the given instance. More specifically, the sizes of the tabu lists are parametrized and experimentally determined as a percentage of the total number of activities. Other parameters include the size of the variable set for MinConOMC and MinConTMC, the execution time of MinConTMC, and the perturbation size threshold. The execution time is determined by multiplying the average number of modes per activity by the time-based parameter with these values $\in \{1, 2, 3, .., 10\}$.

We used the SMAC tool [HHL11, LEF+17] to optimize the parameters of our algorithm. The CPUTIME_LIMIT was set to 172800 seconds (2 days), and the following input was used:

Algorithm 3.4: The overall algorithm (MinCONv1)

Input: $\overrightarrow{S}_i \in \{\overrightarrow{S}\}$, initial schedules generated from earlier stages, i number of threads and $i = \{1, .., 4\}$, $\overrightarrow{S}_{local}$ and $\overrightarrow{S}_{best} = \overrightarrow{S}_i$;
Output: An optimal local solution;

```
1  NoImprovement = true; LocalImprovement = true;
2  repeat
3      repeat
4          Apply combinations of ClonePROJ(), ClonePROJPart(), ComE(), ComF()
           to each S_i;
5          if S_local > min_{i∈{1,..,4}} S_i then
6              S_local = min_{i∈{1,..,4}} S_i; NoImprovement = false;
7          else
8              NoImprovement = true;
9          end
10     until NoImprovement;
11     if LocalImprovement then
12         do
13             Apply MinConOMC() to each S_i;
14             if S_local > min_{i∈{1,..,4}} S_i then
15                 S_local = min_{i∈{1,..,4}} S_i; LocalImprovement = false;
16             else
17                 LocalImprovement = true;
18             end
19         while LocalImprovement;
20         Apply PCom(), MinConTMC() to each S_i;
21     else
22         Apply MinConFMC(), MinConSJL(), MinConSJR(), INVS () to each S_i
23     end
24     if S_local > min_{i∈{1,..,4}} S_i then
25         S_local = min_{i∈{1,..,4}} S_i;
26     end
27     if S_local < S_best then
28         S_best= S_local; NoImprovement = false;
29     else
30         NoImprovement = true;
31     end
32     LocalImprovement = NoImprovement;
33     if LocalImprovement = false then
34         LocalImprovement = true; perturbationSize += 1;
35     else
36         perturbationSize = 1;
37     end
38     Reset S_local; Perturbate(S_i, perturbationSize);
39 until timeExpired;
```

- Training instances consist of sets A and B of instances from MISTA challenge (20 instances),
- Testing instances consist of set X of instances from MISTA challenge (10 instances),
- List of parameters and their domains:
  - *ModeOp-Tabu-list-length* $\in \{3, .., 40\}$,
  - *SeqOp-Tabu-list-length* $\in \{3, .., 40\}$,
  - *VarSetSize* $\in \{3, .., 20\}$,
  - *OpExecTime* $\in \{1, .., 10\}$.
  - *PertSizeThres* $\in \{1, .., 10\}$.

This setup produced parameter configuration as given in Table 3.1:

Table 3.1: SMAC output for parameter values

Parameter name	**Value**
ModeOp-Tabu-list-length	32%
SeqOp-Tabu-list-length	16%
VarSetSize	13
OpExecTime	5
PertSizeThres	10%

In addition to the above experiment, we also experimented with the Cartesian combination of parameters with a limited domain of values and a limited number of instances. Regarding the threshold value for the perturbation size, we found during the tests that after increasing the threshold value for the perturbation quantity above 10%, we could obtain almost no improvement in the instances involved in the tests. We obtained the following configuration of the parameters (see Table 3.2) that were further used in our experiments These parameter values are similar to the values obtained by SMAC, but provided slightly better results for our benchmark problems.

Table 3.2: Parameters used for tests

Parameter name	**Value**
ModeOp-Tabu-list-length	30%
SeqOp-Tabu-list-length	20%
VarSetSize	11
OpExecTime	6
PertSizeThres	10%

3.2 Computational results

Our experiments were performed on a computer with a 64-bit Intel Core i5 processor (3.3 GHz) CPU of four cores, 8GB RAM and the Microsoft Windows operating system. The algorithm was implemented in C# (using Visual Studio 2012), and it runs in parallel with four threads. The executables and the source code of our algorithm can be downloaded from this website: http://dbai.tuwien.ac.at/user/aahmeti/. The solution validator can be found on the MISTA 2013 Challenge website. All the solvers presented in the MISTA 2013 Challenge with which we compared our results were run in parallel in a multithreaded environment.

3.2.1 Benchmark instances

We tested our meta-heuristic approach on the MRCMPSP benchmark instances presented in the MISTA 2013 challenge and the well-known MMLIB instances.

MRCMPSP MISTA 2013 challenge instances

The benchmark instances used in the MISTA 2013 challenge are comprised of three sets A, B and X. Each set of benchmark instances is comprised of ten instances. Table 3.3 lists th characteristics of these instances. A global renewable resource replaces a local renewable resource when present. A project with two global renewable resources has no local renewable resources left. In the preprocessing phase, we removed infeasible modes of activities from the data structure. A mode is considered infeasible if any part of its renewable resource requirements exceeds its capacity. Similar preprocessing was performed by [Gei17] and [BBZ+13].

The MMLIB instances

Although our main objective is to develop solution approaches for MRCMPSP, we also applied our algorithm to solve known MMLIB instances. The MMLIB problem instances are single project multi-mode resource-constrained instances introduced by [VPV14]. This library is comprised of three different sets of well-known and heavily studied problem instances: MMLIB50, MMLIB100 and MMLIB+. The objective function for these problems is the minimization of the project makespan.

The first set contains 540 instances consisting of fifty activities with three execution modes, two renewable and two non-renewable resources. The second set contains 540 instances comprised of one hundred activities with three execution modes, two renewable and two non-renewable resources. The third set is the largest and most complex; it contains 3240 instances comprised of 100 activities with up to nine execution modes, with up to four renewable and four non-renewable resources. The main characteristics of these instances are shown in Table 3.4.

Table 3.3: MRCMPSP (MISTA 2013 challenge) benchmark instances characteristics

Ins.	Proj.	Activities	Modes	Local renewable resources	Non-renewable resources	Global renewable resources
A-1	2	20	3	1	2	1
A-2	2	40	3	1	2	1
A-3	2	60	3	1	2	1
A-4	5	50	3	1	2	1
A-5	5	100	3	1	2	1
A-6	5	150	3	1	2	1
A-7	10	100	3	0	2	2
A-8	10	200	3	0	2	2
A-9	10	300	3	1	2	1
A-10	10	300	3	1	2	1
B-1	10	100	3	1	2	1
B-2	10	200	3	0	2	2
B-3	10	300	3	1	2	1
B-4	15	150	3	1	2	1
B-5	15	300	3	1	2	1
B-6	15	450	3	1	2	1
B-7	20	200	3	1	2	1
B-8	20	400	3	0	2	2
B-9	20	600	3	1	2	1
B-10	20	420	3	0	2	2
X-1	10	100	3	0	2	2
X-2	10	200	3	1	2	1
X-3	10	300	3	1	2	1
X-4	15	150	3	0	2	2
X-5	15	300	3	1	2	1
X-6	15	450	3	1	2	1
X-7	20	200	3	1	2	1
X-8	20	400	3	1	2	1
X-9	20	600	3	1	2	1
X-10	20	410	3	1	2	1

The MMLIB instances have been the subject of intensive research by many researchers for a long time. Some of the best known results for these instances generated by different solvers, under different experimental settings, have been reported by [VPV14]. Table 3.5 lists these results for the MMLIB50, MMLIB100, and MMLIB+ datasets.

Table 3.4: MMLIB benchmark instances characteristics [VPV14]

MMLIB instances			
	MMLIB50	MMLIB100	MMLIB+
Number of instances	540	540	3240
Renewable resources	2	2	2,4
Non-renewable resources	2	2	2,4
Number of modes per activity	3	3	3, 6, 9
Number of activities per project	50	100	50, 100

Table 3.5: Statistics of results generated by different solvers for the MMLIB50, MMLIB100, and MMLIB+ datasets according to 1000, 5000 and 50000 number of schedules reported in [VPV14] and [Gei17]

MMLIB instances results									
	MMLIB50			MMLIB100			MMLIB+		
Author	Number of schedules			Number of schedules			Number of schedules		
	1000	5000	50000	1000	5000	50000	1000	5000	50000
[SSW94]	(21.85)	(23.52)	(25.00)	(18.89)	(19.81)	(21.48)	-	(4.78)	(4.78)
[Boc96b]	(67.59)	(69.63)	(78.52)	(66.67)	(66.67)	(66.67)	-	(54.48)	(65.77)
[BL03]	(72.96)	(77.78)	(82.78)	(66.67)	(67.04)	(67.41)	-	(67.96)	(70.59)
[CV11]	(83.33)	(83.33)	(83.15)	(83.33)	(83.33)	(81.85)	-	(50.00)	(53.90)
[Ozd99]	(83.15)	(83.33)	(83.33)	(66.67)	(67.59)	(67.59)	-	(68.64)	(68.77)
[CHW08]	(83.33)	(83.33)	(83.33)	(83.33)	(83.33)	(83.33)	-	(66.67)	(66.67)
[MT97]	(72.22)	(83.52)	46.91	(67.04)	(69.63)	(83.33)	-	(72.59)	(97.22)
[AMR03]	56.06	43.05	40.69	61.80	52.67	46.68	-	177.55	164.87
[WVBC11]	(89.07)	(94.63)	33.49	(83.15)	(83.89)	(96.85)	-	145.78	132.88
[JDSR08]	49.98	38.86	32.01	(91.85)	49.41	40.23	-	(94.29)	137.99
[ZTL06]	49.25	35.94	30.23	57.42	44.05	35.35	-	(99.81)	(99.85)
[TC09]	65.17	37.92	29.44	72.95	66.04	37.04	-	183.02	142.14
[WF12]	43.17	31.95	28.76	52.94	38.55	30.45	-	144.84	110.43
[RDK09]	38.49	32.16	28.55	45.16	37.00	34.17	-	131.45	131.07
[JMR+01]	49.06	33.81	27.81	53.97	39.05	30.27	-	121.09	103.19
[EF10]	43.84	32.47	26.92	56.21	40.22	30.02	-	130.06	106.35
[Har01]	35.40	30.61	26.81	39.96	33.98	29.04	-	132.01	111.45
[LTCB09]	34.16	28.59	26.69	36.29	31.01	27.89	-	114.07	102.73
[DJSL09]	46.19	32.46	26.27	52.31	36.87	28.92	-	126.69	103.75
[PV10]	34.07	27.12	24.93	37.58	29.55	25.63	-	(97.59)	(97.59)
[PV11]	28.17	25.45	23.79	29.77	26.51	24.02	-	101.45	92.76
[Gei17]	42.96	33.02	28.35	53.26	44.11	32.91	170.67	149.50	114.22

(Number): percentage of feasible solutions found.
Number: average percentage deviation from the lower bounds.

3.2.2 Experiments with MRCMPSP benchmark instances - MISTA 2013 challenge instances

We ran our algorithm under similar conditions as the MISTA 2013 challenge. We performed ten runs per instance, and the time limit for each run was set to 300 seconds. Table 3.6 shows the results for three sets of instances proposed in this challenge. We report the best result, the average, and the standard deviation for each instance. The results are given in TPD/TMS format, where TPD is the primary objective, and TMS is used as a tiebreaker. Evaluation of the new neighborhood operators was the first objective. We tested our algorithm with and without these operators for every instance of benchmark data sets of the MISTA 2013 challenge. Our algorithm achieved better results with the neighborhood operators CloneProj, CloneProjPart, and CloneSeq for almost every instance. The improvements in results for each instance can be seen in Figure 3.6. Our new neighborhood operators play the role of an intensification mechanism for our solver.

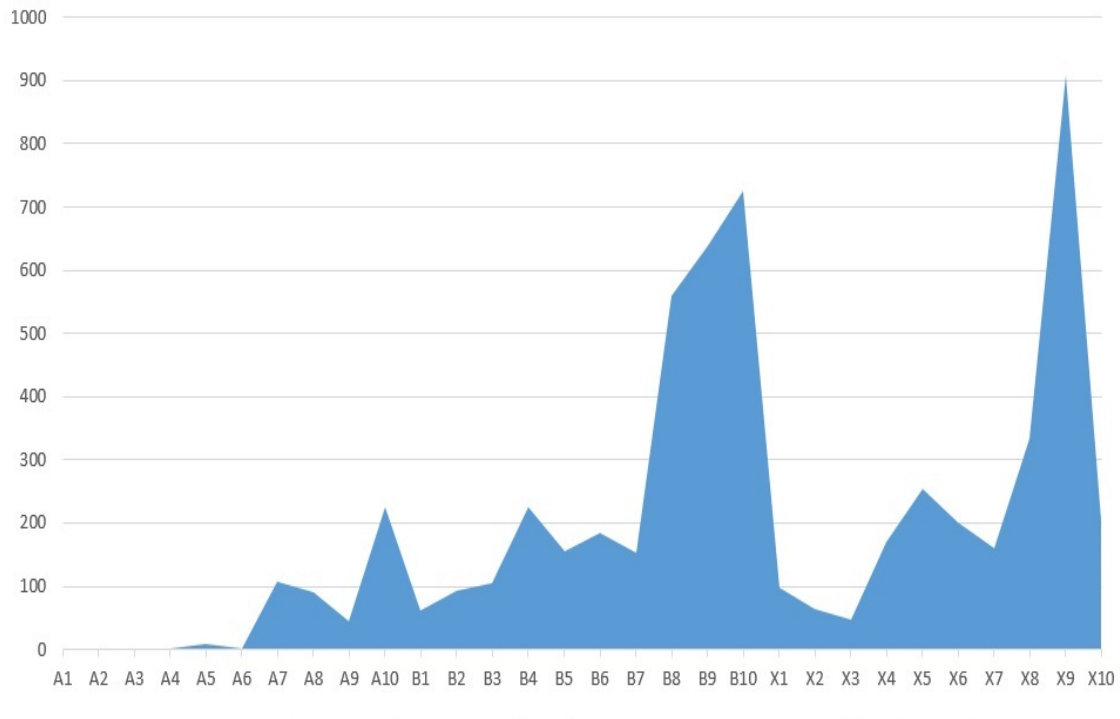

Figure 3.6: Improvement of results achieved with neighborhood operators *CloneProj*, *CloneProjPart* and *CloneSeq*

The horizontal axis shows the instances involved in the tests, while the vertical axis shows the level of improvements (differences between TPDs for each instance obtained

by running the algorithm with and without *xClone* operators).

Table 3.6: Results of our algorithm (MinCONv1) on MISTA 2013 challenge instances under time limit restrictions

Ins.	Best TPD/TMS	Average TPD/TMS	St. Dev.
A1	1/23	1/23	0/0
A2	2/41	2/41	0/0
A3	0/50	0/50	0/0
A4	65/45	66/46	0/0
A5	163/108	168/110	4/2
A6	152/96	166/102	8/3
A7	652/208	690/212	22/3
A8	335/163	376/164	19/3
A9	253/137	278/145	12/5
A10	980/331	1037/342	33/7
B1	361/129	377/132	10/2
B2	502/180	545/185	21/5
B3	637/224	672/228	20/3
B4	1415/290	1476/300	35/6
B5	927/268	1005/274	44/5
B6	1146/253	1209/263	36/5
B7	864/249	939/247	69/3
B8	3836/626	4059/635	139/12
B9	5757/926	6174/966	241/20
B10	3654/514	3931/522	95/8
X1	427/150	456/150	15/4
X2	408/174	431/176	11/3
X3	407/206	424/206	17/3
X4	1081/221	1123/221	42/5
X5	2089/428	2201/420	65/6
X6	953/284	1021/283	33/10
X7	968/248	1043/250	41/8
X8	1515/324	1662/338	96/10
X9	4167/776	4495/795	147/13
X10	1934/427	2127/451	124/11

In the next experiment, we compared our algorithm with the best performing approaches of the MISTA challenge. As in the MISTA challenge, a different computer was used (a machine with 64-bit Intel Core i7 processor (3.4 GHz) CPU of eight cores, 8 GB RAM); we set the runtime in our machine based on the benchmark program (the 64-bit version)

of the 2011 International Timetabling Competition [1]. This program was executed in the MISTA machine for 645 seconds and 690 seconds in our machine. Therefore, we set the running time of our algorithm to 321 seconds (a time limit of 300 seconds was used in the MISTA challenge). The results for instance sets B and X used in the second phase of the competition are shown in Table 3.7. Due to space limitations, only the results for TPD/TMS and no results for the average and standard deviation are given in this table.

Table 3.7: Comparison of MinCONv1 with the best performing algorithms to date for MRCMPSP, under time limit restrictions

Ins.	[AKK+16]	[Gei17]	[TSCS16]	[AH13]	[APPR13]	MinCONv1
B1	**349/127**	353/125	363/132	432/128	371/127	361/129
B2	443/167	490/176	**434/160**	526/153	790/174	502/180
B3	**545/210**	598/215	660/207	638/205	693/214	637/224
B4	1292/287	**1274/289**	1548/295	1469/297	1671/287	1415/290
B5	**820/254**	866/254	919/254	1075/249	1061/255	927/268
B6	**912/227**	1044/242	1128/232	1083/217	1286/259	1146/253
B7	**792/228**	834/234	908/246	905/231	916/228	864/249
B8	**3176/533**	3585/568	3276/529	3662/528	4959/584	3836/626
B9	**4192/746**	4674/796	5373/769	5465/746	6996/812	5757/926
B10	**3249/456**	3518/469	3325/447	4033/427	4550/483	3654/514
X1	398/142	394/142	**392/142**	478/144	486/150	427/150
X2	**349/163**	368/165	418/165	423/159	431/162	408/174
X3	**324/192**	372/195	326/188	391/186	418/182	407/206
X4	**955/213**	970/215	986/207	1054/198	1133/210	1081/221
X5	**1768/374**	1938/386	2043/375	2076/367	2236/376	2089/428
X6	**719/232**	844/253	880/240	872/214	1008/245	953/284
X7	**861/237**	879/231	944/234	993/229	1029/235	968/248
X8	**1233/283**	1380/296	1478/289	1656/270	1578/298	1515/324
X9	**3268/643**	3645/688	4169/662	5130/635	4708/696	4167/776
X10	**1600/381**	1669/402	1851/385	1974/376	2248/393	1934/427

The best results were obtained by solvers that used a larger number of neighborhood operators. The best performing solver [AKK+16] used 13 neighborhood operators, whereas the second-ranked solver introduced by [Gei17] used four operators. Compared to other approaches, our solver is competitive with the third-ranked solver [TSCS16] (our solver gives better results for instances B1, B3, B4, B7, X2, X9). Moreover, our algorithm outperforms the fourth and fifth-ranked solvers ([AH13], [APPR13] in 13 and 20 instances, respectively.

The authors of the first three solvers ([AKK+16], [Gei17], [TSCS16]) also reported results obtained by using other experimental settings. [AKK+16] presented new results of their solver obtained with over 2500 runs per instance. Geiger [Gei17] used 20 runs per instance

[1] https://www.utwente.nl/ctit/hstt/itc2011/benchmarking/

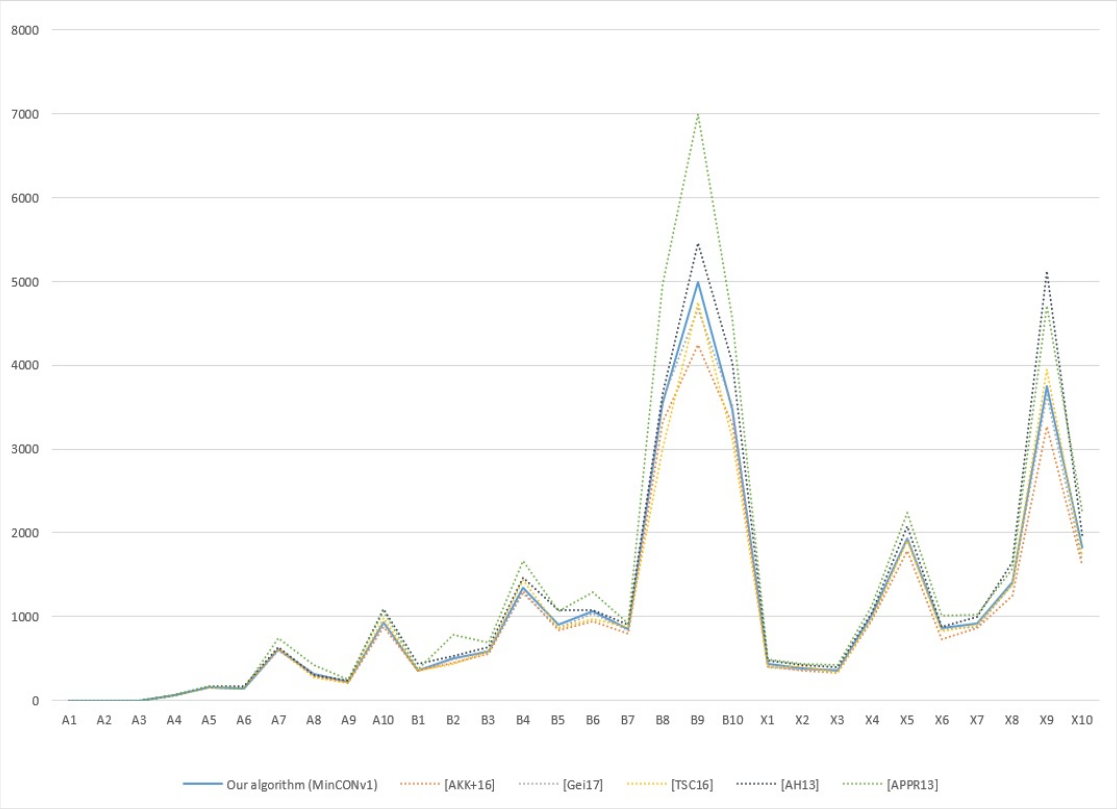

Figure 3.7: Comparison between MinCONv1 results and some of the best results achieved so far for the MRCMPSP

and [TSCS16] used 50 runs. We also experimented with a time limit of 1 hour and five runs per instance. A comparison of the best results can be found in Table 3.8. Although the results cannot be compared directly due to different experimental settings, we can see that our solver gives better results for five instances (A8, B3, B6, B8, B10) compared to [Gei17] and for nine instances (A4, A6, A7, A10, B1, B4, B7, X2, X9) compared with [TSCS16], and matching results in three instances (A1, A2, A3) compared with [AKK$^+$16].

Figure 3.7 shows the comparison between our results and the best results obtained to date (to the best of our knowledge) for the MRCMPSP problem.

3.2.3 Experiments with MRCPSP benchmark instances - MMLIB instances

In order to evaluate the performance of our meta-heuristic approach, we also tested our algorithm on single-project MRCPSP benchmark instances introduced by [VPV14], i.e.,

Table 3.8: Comparison of MinCONv1 results with the best approaches using different experimental settings.

Ins.	[AKK+16] 2500 runs	[Gei17] 20 runs	[TSCS16] 50 runs	MinCONv1 5 runs, 1h	Average	St. Dev.
A1	**1/23**	**1/23**	**1/23**	**1/23**	1/23	0/0
A2	**2/41**	**2/41**	**2/41**	**2/41**	2/41	0/0
A3	**0/50**	**0/50**	**0/50**	**0/50**	0/50	0/0
A4	**65/42**	**65/42**	68/50	65/45	66/42	0/2
A5	**150/103**	153/104	154/104	157/106	160/107	3/1
A6	**133/99**	144/94	151/94	149/96	154/99	4/2
A7	**590/190**	601/206	626/194	614/209	636/209	11/2
A8	**272/148**	319/162	281/147	318/160	326/162	9/2
A9	**197/122**	225/128	212/127	225/135	237/136	6/1
A10	**836/303**	920/313	983/309	936/324	973/330	27/4
B1	**345/124**	349/130	358/131	353/130	369/132	4/1
B2	**431/158**	481/171	431/159	502/181	512/183	8/2
B3	**526/200**	604/214	585/196	585/216	599/219	10/2
B4	**1252/275**	1283/287	1435/294	1340/297	1356/293	11/4
B5	**807/245**	866/252	867/254	904/268	919/270	10/4
B6	**905/225**	1067/246	970/224	1064/249	1090/252	22/4
B7	**782/225**	827/232	876/234	850/240	862/241	10/3
B8	3048/523	3618/565	**3001/520**	3550/600	3664/611	128/7
B9	**4062/738**	4606/783	4753/741	4992/845	5277/868	171/15
B10	3140/436	3541/473	**3123/430**	3471/493	3536/498	56/3
X1	**386/137**	-	392/142	433/146	441/148	5/4
X2	**345/158**	-	416/167	381/170	394/171	8/2
X3	**310/187**	-	332/177	352/205	365/203	8/3
X4	**907/201**	-	980/209	1027/217	1035/215	7/2
X5	**1727/362**	-	1904/369	1927/398	1971/403	41/7
X6	**690/226**	-	821/237	861/263	886/267	17/3
X7	**831/220**	-	909/232	913/ 236	931/244	12/6
X8	**1201/279**	-	1389/281	1417/311	1453/315	32/6
X9	**3155/632**	-	3945/639	3756/715	3923/733	100/12
X10	**1573/373**	-	1718/377	1827/420	1862/422	31/4

MMLIB50, MMLIB100, and MMLIB+ instances having the project makespan as the performance measure. MinCONv1 was able to provide new upper bounds to the literature for six benchmark instances from the MMLIB+ library (Table 3.9). For many other instances, we obtained matching results with the previous best solvers. These instances have been the subject of continuous research work, as mentioned earlier. The results reported in this chapter cover the period up to the beginning of 2018. The work presented

so far in this chapter is published in [AM18]. Adapting our approach for solving MMLIB instances was achieved simply by initializing constant α in 2.10 to zero. The problem representation is compatible with MRCMPSP instances.

Table 3.9: New upper bounds achieved by meta-heuristic for MRCPSP problem

	Project total makespans	
Instance	**MinCONv1**	**Different Authors**
Jall64_4.mm	169	174
Jall64_5.mm	149	153
Jall65_2.mm	130	136
Jall65_4.mm	130	138
Jall154_3.mm	148	158
Jall154_4.mm	134	141

3.3 Extensions to MinCONv1

Our efforts to improve MinCONv1 resulted in two extensions of it. The first extension, denoted as MinCONv2, is achieved by adding a new neighborhood operator called *SwapAndModeCh*. The work presented in the following sections of this chapter is published in [AM21].

3.3.1 Extension MinCONv2

SwapAndModeCh is a combination of two existing neighborhood operators, *SwapActivity* and *OneModeChange*, that were used in the implementation of MinCONv1. We encounter the implementation of these operators in the work of [AKK$^+$16], [TSCS16] and [Gei17]. Moreover, in the implementation of local search, we excluded the four-mode-change (*MinConFMC*) neighborhood operator in MinCONv1 in order to speed up the algorithm convergence within a short period of time.

SwapAndModeCh operator performs a swap between an activity and its successor along the schedule while the precedence constraint is not violated. After each swap, the activity is assigned to each of its modes in turn, and if there is an improvement, the solution is accepted.

Figure 3.8 illustrates the neighborhood generated by this operator assuming $act_i = 3$, $SuccessorOf(act_i) = 5$ in a given input solution s. Activity 3 swaps with its descendants in the schedule all the way up to its successor, activity 5, with which it has a precedence constraint. Implementation of *SwapAndModeCh* neighborhood operator is depicted in Algorithm 3.5.

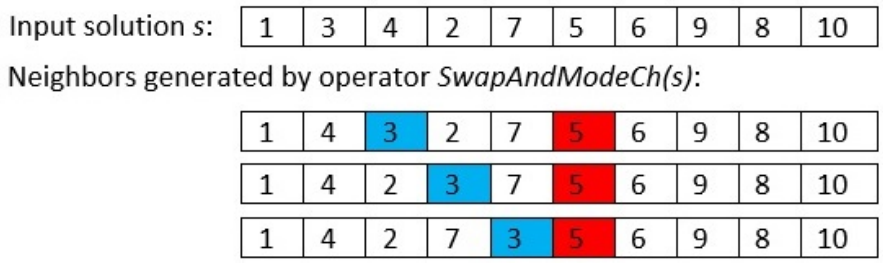

Figure 3.8: SwapAndModeCh(s) neighborhood operator

Algorithm 3.5: SwapAndModeCh(s) neighborhood operator

Input: $s' \in \{\overrightarrow{S}\}$, *a feasible schedule*;
Output: A feasible initial solution;

```
repeat
    Improved = false;
    act_i = random.select.from(J_i);
    while (not IsSuccessor(s, act_{i+1})) do
        s' ← Swap(s, act_i, act_{i+1});
        foreach modes_of_act_{i+1} do
            s'' = OneModeChange(s', act_{i+1});
            if (eval(s'') < eval(s)) then
                s ← s'';
                Improved = true;
            end
        end
    end
until not Improved;
```

3.3.2 Extension MinCONv3

Additionally, we investigated a further extension of MinCONV2 by modifying the existing neighborhood operator one-mode-change (MinConOMC). Due to the still stochastic nature of MinCONv2, we replaced the neighborhood operator MinConOMC, which is based on the idea of min-conflicts heuristic with a neighborhood operator hill-climbing-one-mode-change (HCOMC), which is based on the hill-climbing concept (see Algorithm 3.8). While MinConOMC focuses only on the variables (activities) that are in conflict for the same resources, HCOMC generates all solutions from a certain neighborhood. The function *BroadcastBestLocal()* is invoked periodically to find the best actual solution ($\overrightarrow{S}_{bestlocal}$) among the threads (see Algorithm 3.7). The implementation of MinCONv3 is shown in Algorithm 3.6.

Since our approach is executed in parallel with four threads, we divided the activities

Algorithm 3.6: The algorithm MinCONv3

Input: $\overrightarrow{S}_i \in \{\overrightarrow{S}\}$, initial schedules generated from earlier stages, i number of threads and $i = \{1, .., 4\}$, $\overrightarrow{S}_{local}$ and $\overrightarrow{S}_{best} = \overrightarrow{S}_i$;

Output: An optimal local solution;

```
1  NoImprovement = true; LocalImprovement = true;
2  repeat
3      repeat
4          Apply combinations of ClonePROJ(), ClonePROJPart(), ComE(), ComF()
           to each S_i;
5          if (BroadcastBestLocal()) then
6              NoImprovement = false;
7          else
8              NoImprovement = true;
9          end
10     until NoImprovement;
11     if LocalImprovement then
12         do
13             Apply HCOMC() to each S_i;
14             if (BroadcastBestLocal()) then
15                 LocalImprovement = false;
16             else
17                 LocalImprovement = true;
18             end
19         while LocalImprovement;
20         Apply PCom(), MinConTMC() to each S_i;
21     else
22         Apply SwapAndModeCh(), MinConSJL(), MinConSJR(), INVS () to each
           S_i
23     end
24     BroadcastBestLocal();
25     if S_local > min_{i∈{1,..,4}} S_i then
26         S_local = min_{i∈{1,..,4}} S_i;
27     end
28     if S_local < S_best then
29         S_best= S_local; NoImprovement = false;
30     else
31         NoImprovement = true;
32     end
33      LocalImprovement = NoImprovement;
34     if LocalImprovement = false then
35         LocalImprovement = true; perturbationSize += 1;
36     else
37         perturbationSize = 1;
38     end
39     Reset S_local; Perturbate(S_i, perturbationSize);
40 until timeExpired;
```

from the schedule into four equal groups and executed four instances of hill-climbing based neighborhood operators for each group of activities in parallel. This approach appeared to achieve slightly better results than MinCONv2, better average results, and lower standard deviation when executed under time limit constraints. The definition of the *BroadcastBestLocal()* method is shown in Algorithm 3.7.

Algorithm 3.7: BroadcastBestLocal()

$s_{minLocal} \leftarrow \min_{i \in \{1,..,4\}} eval(\overrightarrow{S}_i)$;
$s_{temp} \leftarrow \emptyset$;
$s' \leftarrow Swap(s, act_i, act_{i+1})$;
if $(eval(\overrightarrow{S}_{bestlocal}) > eval(s_{minLocal}))$ **then**
  $s_{minLocal} = \min_{i \in \{1,..,4\}} eval(\overrightarrow{S}_i)$;
  $s_{temp} \leftarrow \overrightarrow{S}_{arg \min_{i \in \{1,..,4\}} \overrightarrow{S}_i}$;
  foreach $\overrightarrow{S}_i$ **do**
    $\overrightarrow{S}_i \leftarrow s_{temp}$;
  end
  $\overrightarrow{S}_{bestlocal} \leftarrow s_{temp}$;
  $return \quad true$;
else
  $return \quad false$;
end

Acceptance criteria, perturbation, and parameter configuration for MinCONv3 are the same as for MinCONv1.

Algorithm 3.8: Hill climbing heuristic, HCOMC()

Input: $s \in \{\overrightarrow{S}\}, \quad a \quad feasible \quad schedule$;
Output: An improved solution;
foreach $\in Sequence(startSeq, endSeq)$ **do**
  foreach $m \in ModesOf(a)$ **do**
    $s' \leftarrow ChangeModeOfa(s, a)$;
    if $eval(s') > eval(s')$ **then**
      $s = s'$;
    end
  end
end

3.3.3 Evaluation of the extended meta-heuristics

We have conducted extensive experiments with all three variants of the meta-heuristics on different types of scheduling problems. We discuss these experiments in more detail in the following subsections. For clarity, Table 3.10 shows the list of neighborhood operators employed in each variant of the meta-heuristic.

Table 3.10: List of neighborhood operators used in each meta-heuristic variant

Neighborhood operator	MinConv3	MinConv2	MinConv1
CloneProj	✓	✓	✓
CloneProjPart	✓	✓	✓
CloneSeq	✓	✓	✓
MinConOMC	x	✓	✓
MinConTMC	✓	✓	✓
HCOMC	✓	x	x
MinConFMC	x	x	✓
MinConSJL	✓	✓	✓
MinConSJR	✓	✓	✓
INVS	✓	✓	✓
ComE	✓	✓	✓
ComF	✓	✓	✓
PCom	✓	✓	✓
SwapAndModeCh	✓	✓	x

Experiments with MRCMPSP benchmark instances - MISTA 2013 challenge instances

Under time limit constraints, we performed separate experiments with extended versions of MinCONv1, MinCONv2 and MinCONv3 (ten runs per instance and each run five minutes). As can be seen in Table 3.11, the two extended variants, MinCONv2 and MinCONv3, slightly improved most of the results for MRCMPSP benchmark instances compared with MinCONv1.

The MinCONv3 variant of meta-heuristic achieved sixteen better or equal results out of a total thirty, and significantly better average results and lower standard deviation than the MinCONv1 and MinCONv2 variants. In individual comparisons with the other approaches, the MinCONv2 variant also achieved better results in fifteen out of thirty cases, and was equal to MinCONv1 in three cases. According to the analysis performed in section 3.2.2, the MinCONv1 solver was competitive to the third-ranked solver in the MISTA competition.

Therefore, we can conclude that the MinCONv3 is at least competitive with the best third-ranked solver for the MRCMPSP.

Table 3.11: Comparison of the results (TPD/TMS) of the extended meta-heuristic with the results of the MinCONv1 solver for MRCMPSP. The algorithms are executed under time limit constraints

	MinConv1			**MinConv2**			**MinConv3**		
Ins.	Best (TPD/TMS)	Avg (TPD)	Std (TPD)	Best (TPD/TMS)	Avg (TPD)	Std (TPD)	Best (TPD/TMS)	Avg (TPD)	Std (TPD)
A1	**1/23**	**1**	**0**	**1/23**	**1**	**0**	**1/23**	**1**	**0**
A2	**2/41**	**2**	**0**	**2/41**	**2**	**0**	**2/41**	**2**	**0**
A3	**0/50**	**0**	**0**	**0/50**	**0**	**0**	**0/50**	**0**	**0**
A4	65/45	66	0	**65/42**	66	1	**65/42**	**66**	**0**
A5	163/108	168	8	162/107	171	5	**159/108**	**168**	**5**
A6	152/96	166	8	156/94	165	8	**152/96**	**162**	**6**
A7	652/208	690	22	644/203	677	17	**642/210**	**665**	**15**
A8	**335/163**	376	19	337/166	371	19	348/159	**369**	**17**
A9	253/137	278	12	**245/139**	278	12	265/145	285	**11**
A10	980/331	**1037**	**33**	**969/338**	1055	41	999/340	1044	37
B1	361/129	377	10	**355/129**	375	5	360/130	**370**	**5**
B2	502/180	545	21	**498/177**	544	16	518/187	**538**	**14**
B3	637/224	672	20	639/230	671	14	**634/224**	**662**	23
B4	1415/290	1476	35	1386/302	1466	40	**1362/289**	**1444**	38
B5	**927/268**	1005	44	955/275	1015	33	953/277	**1003**	**30**
B6	1146/253	1209	36	1139/261	1211	37	**1116/255**	**1202**	41
B7	**864/249**	939	69	890/251	967	33	924/258	973	**26**
B8	3836/626	**4059**	**139**	**3687/628**	4065	179	3785/619	4169	190
B9	**5757/926**	**6174**	**241**	5858/948	6397	286	6291/981	6803	340
B10	3654/514	3931	95	**3636/456**	**3881**	**178**	3836/507	3976	121
X1	427/150	456	15	435/148	460	15	**419/148**	**452**	**13**
X2	408/174	**431**	**11**	419/175	448	19	**398/170**	436	22
X3	407/206	**424**	17	**382/202**	429	**12**	411/209	433	**12**
X4	1081/221	1123	42	1035/221	1103	23	**1032/223**	**1102**	29
X5	2089/428	**2201**	65	2083/393	2238	**48**	**2065/407**	2223	84
X6	**953/284**	**1021**	**33**	967/281	1062	85	1016/285	1062	43
X7	968/248	1043	41	951/245	1032	40	**946/246**	**999**	**23**
X8	**1515/324**	**1662**	96	1584/329	1700	**62**	1562/329	1767	108
X9	**4167/776**	**4495**	**147**	4374/790	4791	243	4351/779	4812	237
X10	1934/427	2127	124	1938/437	2137	116	**1903/433**	**2124**	**110**

Experiments with MRCPSP benchmark instances - MMLIB instances

Additionally, we tested the extended meta-heuristic MinCONv3 with MMLIB benchmark instances, having the project makespan as the performance measure. the tests resulted in the achievement of new upper bounds for five instances (Table 3.12).

Table 3.12: New upper bounds achieved by meta-heuristic for MRCPSP

	Project total makespans	
Instance	**MinCONv3**	**Different Authors**
Jall127_3.mm	142	143
Jall128_5.mm	94	95
Jall184_1.mm	177	179
Jall263_4.mm	146	147
Jall289_5.mm	196	197

3.4 Discussion and analysis

In this chapter, we introduced a new solution approach for the MRCMPSP based on the min-conflicts heuristic. This technique efficiently explores the neighborhood by focusing on variables (activities) in conflict for the same resources. We further used the min-conflicts heuristic within the framework of iterated local search and combined it with the tabu search components. In our experiments, we found that the neighborhood operators that manipulate the modes of activities significantly impact the quality of the active schedule. Therefore, we mainly focused on neighborhood operators that manipulate the modes of activities and proposed three new project-wise neighborhood operators for the MRCMPSP. Experimental results have shown that the new operators are useful for this problem.

Our method was evaluated using MRCMPSP instances from the MISTA 2013 challenge and compared with the best approaches. The proposed method outperforms the solver ranked fourth in this competition for most instances and obtains competitive results to the third-ranked solver. Furthermore, our solver was able to improve the best existing literature results for eleven well-known instances of the MRCPSP MMLIB benchmark instances, i.e. MMLIB50, MMLIB100, and MMLIB+. The solution approach presented in this chapter will serve as the core for the meta-heuristic component of a hybrid model that we will introduce in next chapters.

Our efforts to improve our meta-heuristic solver (MinCONv1) led to two consecutive extensions. First, we fine-tuned MinCONv1 by adding a new neighborhood operator (SwapAndModeCh), which results from the combination of two existing neighborhood operators, SwapActivity and OneModeChange. Furthermore, we removed the four-mode-change (MinConFMC) neighborhood operator. This extension of MinCONv1(named MinCONv2) achieved better results in fifteen out of thirty instances and was equal to MinCONv1 in three cases.

Our second extension (named MinCONv3) is based on modifying the existing neighborhood operator one-mode-change (MinConOMC). Due to the stochastic nature of MinCONv2, we replaced the MinConOMC neighborhood operator based on the idea of min-conflicts with a neighborhood operator, hill-climbing-one-mode-change (HCOMC), based on the hill-climbing concept. This approach achieved slightly better results than

MinCONv2, better average results, and lower standard deviation. We will use the last two extensions in our future steps to develop variants of hybrid solvers in combination with a CP model for the MRCMPSP.

CHAPTER 4

A Constraint Programming Model for MRCMPSP

In this chapter, we introduce an exact method for MRCMPSP based on constraint programming. Although constraint programming has been used very successfully in the literature to solve different variants of the project scheduling problem, to the best of our knowledge, this solution paradigm has not yet been studied for MRCMPSP as a generalized variant of the project scheduling problem.

4.1 The CP model description

As stated earlier, MRCMPSP is a more general variant of project scheduling problems and is more in line with real-life industry problems. The multi-objective nature of MRCMPSP makes it an even more difficult problem to solve. Most of the exact models for MRCMPSP designed so far are based on integer programming and are presented in combination with meta-heuristics within a hybrid approach, e.g. the hybrid model of [TSCS16], which consists of a hybrid search method that combines a parallel local search with multiple neighborhoods with integer programming, or the combination of mixed integer programming with the large neighborhood search of [AH13]. In this chapter, we provide a constraint programming model for this problem that is based on the existing model ([LRSV18]), which has been applied to the related problem MRCPSP. However, since our problem is a more general variant, we propose several extensions to solve MRCMPSP through constraint programming in this chapter.

The main extensions of our constraint programming model for MRCMPSP include:

- Constraints related to a set of global renewable resources (G^ρ) shared among all the projects, whose availability is limited by c^ρ_g, $g \in G^\rho$. There are no global non-renewable resources.

- Release dates constraints and dummy activities for each project.
- Implementation of an objective function that consists of finding a feasible schedule that fulfils constraints while minimizing the TPD and the TMS. Project delay is defined as the difference between CPD and the actual project makespan. TPD is the primary objective, and TMS, as the duration of the entire multi-project schedule, is used as a tie-breaker.
- Multi-dimensional data structures to model the execution modes of the multi-project instances, the local renewable and non-renewable resources and to add global resources.

We further describe the main components and constraints of the CP model. We have implemented it using the IBM CP Optimizer. According to the problem definition, each project $i \in P = \{1, 2, .., n\}$ is comprised of a set of non-preemptive activities or jobs J_i and has a release date r_i, i.e. the earliest time when the activities of the project i can start.
In our model, activities are modeled as interval variables:

$$\text{interval}\, J_i \qquad \forall i \in P \tag{4.1}$$

An interval variable models a time interval and is the basic building block of a CP model. It has several characteristics, the most important of which are: start, end, and size. The value of an interval variable is an integer interval defined by its start and its end (for more details about CP model components, see [LRSV18]). In (6) it is denoted that all activities of all projects are declared as interval variables in the CP model.
Projects' release dates constraints and dummy activities:

$$\text{startOf}(b_i) = r_i \qquad \forall i \in P, b_i \in P_i \tag{4.2}$$

$$\text{startOf}(b_i) \leq \text{startOf}(j) \qquad \forall i \in P, j \in P_i \tag{4.3}$$

$$\text{endOf}(j) \leq \text{startOf}(e_i) \qquad \forall i \in P, e_i, j \in P_i \tag{4.4}$$

In a CP model, the *startOf* and *endOf* methods access the start and end time of an interval variable, respectively. To introduce project release dates and dummy activities as constraints in our model, we defined our *startOf* and *endOf* methods. These methods initialize the *StartMin* property of the interval variables with the corresponding project's release date (r_i) and impose the precedence constraints of the dummy activities. The *StartMin* property sets and-or returns the minimal start value of an interval variable.
Each activity $j \in J_i$, of every project $i \in P$, has one or more available execution modes $m \in M_{ij}$. The execution mode of an activity determines the duration d_{ijm} required to complete the activity and its specific resource requirements. The execution modes and the processing time in each mode for every activity are also modeled as decision variables in our model:

$$\text{interval}\, m_{ij}\, \text{optional} \qquad \forall i \in P, j \in J_i, m_{ij} \in M_{ij} \tag{4.5}$$

$$\text{interval}\, d_{ijm}\, \text{optional} \qquad \forall i \in P, j \in J_i, m \in M_{ij} \tag{4.6}$$

An interval variable in a CP model can be defined as optional, meaning it can be present or absent, i.e. it additionally has an absence value in its domain. An activity can be executed in only one certain mode at any time unit t, so its execution modes must be defined as optional.
The constraint that every activity $j \in J_i$, can be executed in only one of its modes at any time unit t is imposed through the *alternative* constraint of the CP Optimizer:

$$\text{alternative}(j, [m_{ij}]_{\in M_{ij}}) \qquad \forall i \in P, j \in J_i \tag{4.7}$$

The *alternative* constraint imposes that if the activity j is present, then exactly one of its modes (m_{ij}) is present, with the same start and end.
Resources are expressed as cumul-function expressions:

$$\text{cumulFunction}\, c_{il} \qquad \forall i \in P, l \in L_i^{\rho} \tag{4.8}$$

$$\text{cumulFunction}\, c_g^{\rho} \qquad \forall g \in G^{\rho} \tag{4.9}$$

Local and global renewable resources are modeled as *cumul function expressions*. The *cumul function expressions* model the time evolution of some quantity that can be increased or decreased by interval variables. These are the proper structures to model the evolution of the quantity of renewable resources impacted by present activities at every time unit of the schedule.

$$\text{intExpr}\, h_{il} \qquad \forall i \in P, l \in L_i^{\nu} \tag{4.10}$$

Since non-renewable resources are depleted incrementally, they are modeled as simple integer expressions. Feasible schedules of projects must always satisfy the following hard constraints:

- For each project $i \in P$ and each local non-renewable resource $l \in L_i^{\nu}$ associated with it, the total resource consumption does not exceed its capacity h_{il}. Since non-renewable resources have fixed capacities throughout the life of the project, we modeled them as scalar expressions in our model:

$$\sum_{j \in J_i} \sum_{m_{ij} \in M_{ij}} \text{presenceOf}(m_{ij})(r_{ijml}^{\nu}) \leq h_{il} \quad \forall i \in P, l \in L_i^{\nu} \tag{4.11}$$

  In the MRCMPSP model, each project has its own list of non-renewable resources. The CP Optimizer constraint *presenceOf* indicates that a certain interval variable is present. It is usually used in combination with other constraints. In (4.11), the presence of the execution mode m_{ij} of the activity j implies the demand for non-renewable resources r_{ijml}^{ν} and the sum of all demands of all activities of project i cannot exceed the total capacity h_{il}.

- For each project $i \in P$ and each local renewable resource $l \in L_i^\rho$ associated with it, the total resource consumption does not exceed its capacity c_{il}. As the name implies, local renewable resources are dedicated to a specific project and have a fixed capacity per time unit, meaning that their capacity constraints have a temporal dimension. Therefore, they are modeled as cumulative functions:

$$\sum_{j \in J_i} \sum_{m_{ij} \in M_{ij}} \text{pulse}(m_{ij}, r^\rho_{ijml}) \leq c_{il} \quad \forall i \in P, l \in L_i^\rho \tag{4.12}$$

  In the MRCMPSP model, each project has its own list of renewable resources. The *pulse* is an *elementary cumul function expression* in the CP Optimizer. It impacts (increases in this case) the value of a *cumul function expression* by the value r^ρ_{ijml} between the start and the end of the interval variable m_{ij} and is equal to zero everywhere else. The value of a *cumul function expression* cannot exceed the capacity c_{il}.

- For any time unit t and any global renewable resource $g \in G^\rho$, the total resource consumption at t does not exceed its capacity c_g. Global renewable resources are modeled as a cumulative function, similar to local renewable resources. There is only one global resource list that applies to all projects:

$$\sum_{i \in P} \sum_{j \in J_i} \sum_{m_{ij} \in M_{ij}} \text{pulse}(m_{ij}, r^\rho_{ijmg}) \leq c_g \qquad \forall g \in G^\rho \tag{4.13}$$

  There is no global list of non-renewable resources. The *pulse* increases the value of a *cumul function expression* by value r^ρ_{ijmg} between the start and the end of the interval variable m_{ij} and is equal to zero everywhere else. The value of a *cumul function expression* cannot exceed the capacity c_g.

- Feasible schedules must satisfy all precedence constraints between activities:

$$\text{endBeforeStart}(a, j) \qquad \forall i \in P, a, j \in P_i \tag{4.14}$$

The CP Optimizer method *endBeforeStart* imposes a constraint that indicates that whenever interval variables a and j are present, j cannot start before a has ended. According to the definition, the main objective function of MRCMPSP is to minimize the TPD of the projects and the TMS of the projects as tie-breakers. It is defined as:

$$f = \min(\max(\text{endOf}(j)) + \sum_{i \in P} (\alpha * \max(\text{endOf}(P_i)))), \forall j \in P_i \tag{4.15}$$

α – is a constant.

From the above formula, it can be seen that the TPD for each project is calculated using the combination of the methods *endOf* and *max*. The first method calculates the end

time of each interval variable (activity) that comprises P_i, and the second method finds the latest one. The TMS is calculated by locating the interval variable with the latest end time $\forall j \in P_i$. Finally, the entire expression is passed to the minimize objective for minimization. The constant α that determines the weight for the TPD is assigned the value of 100000, similar to the MISTA 2013 challenge conditions.

4.2 Evaluation of the CP Model

We performed experiments with our CP model on the same benchmark instances as MinCONv3. According to [LRSV18], the search in a CP model can be controlled by configuring several parameters, e.g. search type, random seed, time limit, etc. The results of the CP model for MRCMPSP instances shown in Table 4.1 are performed with the following parameter configuration:

- Search Type = Restart
- Random Seed = 1292619981
- Time Limit = 300 s

In restart search mode, the algorithm restarts and performs the depth-first search after a parameterized number of failed attempts. Another type of search we have experimented with is automatic search, which employs a large neighborhood search and failure directed search ([LRSV18]). The former tries to converge quickly to a good quality solution, and the latter tries to prove that no better solution than the existing one exists when the search space is too small or LNS cannot improve the solution further. The CP model uses a random seed parameter for tie-breaking situations only. In Table 4.1, it can be seen that the CP model provided improvements for three instances and three matching results under time constraints (compared to [AKK+16]).

Table 4.2 shows the comparison between our CP Model and the MinCONv3 for MRCMPSP. The same results are shown graphically in Figure 4.1. The algorithms are executed under time limit constraints. As can be seen, the CP model achieves better results than the MinCONv3 method for seventeen out of thirty instances. The work presented in this chapter is published in [AM21].

4.3 Discussion and analysis

Building and testing an exact model was another step towards our goal of developing a hybrid approach. We chose a CP model designed through IBM CP Optimizer for two reasons. First, CP Models have proven to be successful in solving different scheduling problems of industrial and research interest, and second, to our knowledge, this is the first CP Model developed for the MRCMPSP.

Table 4.1: Comparison of results (TPD/TMS) between our CP Model and the best solvers results for MRCMPSP. The algorithms are executed under time limit constraints

Ins.	**[AKK+16]**	**[Gei17]**	**[TSCS16]**	**[AH13]**	**CP model**
A1	**1/23**	-	-	-	**1/23**
A2	**2/41**	-	-	-	**2/41**
A3	**0/50**	-	-	-	**0/50**
A4	**65/42**	-	-	-	65/45
A5	**153/105**	-	-	-	173/110
A6	**147/96**	-	-	-	162/104
A7	**596/196**	-	-	-	655/197
A8	302/155	-	-	-	**279/148**
A9	223/119	-	-	-	**212/124**
A10	**969/314**	-	-	-	1017/317
B1	**349/127**	353/125	363/132	432/128	375/130
B2	443/167	490/176	**434/160**	526/153	438/167
B3	**545/210**	598/215	660/207	638/205	586/206
B4	1292/287	**1274/289**	1548/295	1469/297	1493/286
B5	**820/254**	866/254	919/254	1075/249	922/267
B6	**912/227**	1044/242	1128/232	1083/217	936/227
B7	**792/228**	834/234	908/246	905/231	1003/252
B8	3176/533	3585/568	3276/529	3662/528	**3113/544**
B9	**4192/746**	4674/796	5373/769	5465/746	5253/833
B10	**3249/456**	3518/469	3325/447	4033/427	3295/455
X1	398/142	394/142	**392/142**	478/144	443/144
X2	**349/163**	368/165	418/165	423/159	405/167
X3	**324/192**	372/195	326/188	391/186	346/194
X4	**955/213**	970/215	986/207	1054/198	996/208
X5	**1768/374**	1938/386	2043/375	2076/367	1940/376
X6	**719/232**	844/253	880/240	872/214	799/243
X7	**861/237**	879/231	944/234	993/229	902/233
X8	**1233/283**	1380/296	1478/289	1656/270	1366/288
X9	**3268/643**	3645/688	4169/662	5130/635	4320/760
X10	**1600/381**	1669/402	1851/385	1974/376	1739/396

The search in a CP model can be determined through the configuration of several parameters. Therefore, part of the preliminary work was invested in configuring the CP solver properly. Initial experiments appeared to be promising; the CP model achieved promising results for small and medium-size benchmark instances. Compared with the state-of-the-art algorithms for MRCMPSP, our CP model obtained three new upper bounds for well-known benchmark problem instances. It also achieved better results than

Table 4.2: Comparison of results (TPD/TMS) between our CP Model and the MinCONv3 for MRCMPSP. The algorithms are executed under time limit constraints

Ins.	CP Model	MinCONv3
A1	**1/23**	**1/23**
A2	**2/41**	**2/41**
A3	**0/50**	**0/50**
A4	65/45	**65/42**
A5	173/110	**159/108**
A6	162/104	**152/96**
A7	655/197	**642/210**
A8	**279/148**	348/159
A9	**212/124**	265/145
A10	1017/317	**999/340**
B1	375/130	**360/130**
B2	**438/167**	518/187
B3	**586/206**	634/224
B4	1493/286	**1362/289**
B5	**922/267**	953/277
B6	**936/227**	1116/255
B7	1003/252	**924/258**
B8	**3113/544**	3785/619
B9	**5253/833**	6291/981
B10	**3295/455**	3836/507
X1	443/144	**419/148**
X2	405/167	**398/170**
X3	**346/194**	411/209
X4	**996/208**	1032/223
X5	**1940/376**	2065/407
X6	**799/243**	1016/285
X7	**902/233**	946/246
X8	**1366/288**	1562/329
X9	**4320/760**	4351/779
X10	**1739/396**	1903/433

the MinCONv3 meta-heuristic solver for seventeen out of thirty instances. MinCONv3 provided better or equal results for thirteen out of thirty MRCMPSP benchmark instances under time limit constraints. In the next chapters, we will introduce the experimental results of combining this CP model with a meta-heuristic method in a hybrid approach. The starting point for this hybrid model design architecture is based on several models analyzed in the literature. As we will show later, this hybrid approach has achieved new upper bounds for many instances of different categories of project scheduling problems

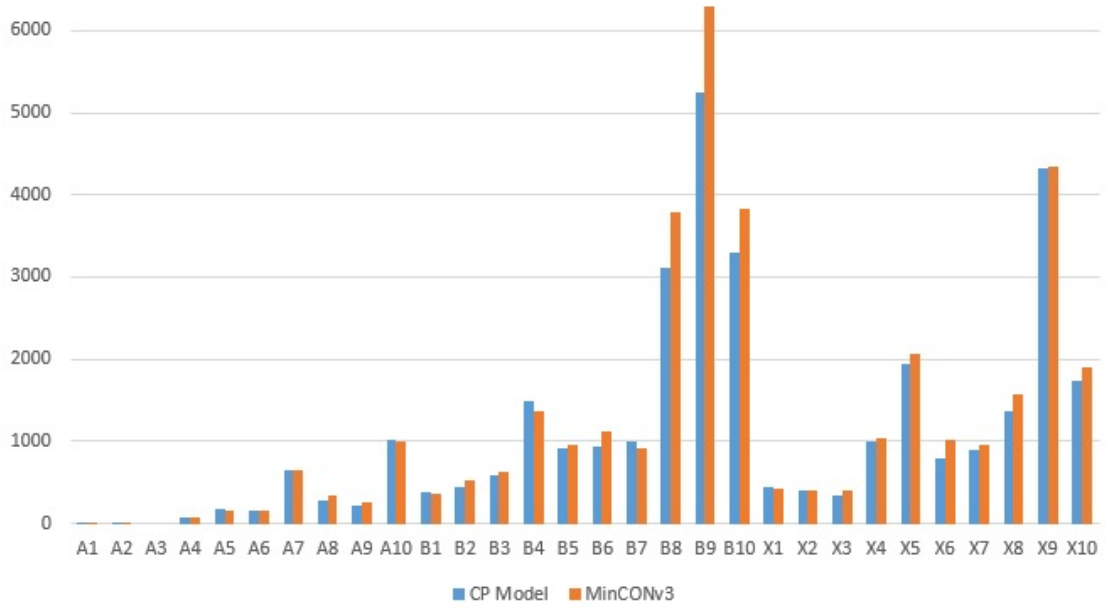

Figure 4.1: Comparison of results (TPD/TMS) between our CP Model and the MinCONv3 for MRCMPSP. The algorithms are executed under time limit constraints

and outperformed all state-of-the-art solvers for most instances under relaxed time constraints. Another important aspect of our efforts was testing our hybrid model and its constituent components, i.e., the meta-heuristic and the CP model, on other variants of the project scheduling problems.

CHAPTER 5

Hybrid method

In this chapter, we introduce our hybrid method that combines meta-heuristic algorithms and the exact model for MRCMPSP described in the previous chapters. Our main goal regarding hybrid approaches for MRCMPSP is to combine two complementary search strategies.

5.1 Hybrid method description

Various classifications and taxonomies for hybrid approaches can be found in the literature ([Tal09]), such as combining meta-heuristics with constraint programming, combining meta-heuristics with exact methods from mathematical programming, combining meta-heuristics with other meta-heuristics or machine learning techniques. We chose to combine an exact approach, a CP model, with a meta-heuristic approach based on the local search described in the previous section. The pseudo-code of our hybrid algorithms is presented in Algorithm 5.1.

According to [Tal09], our hybrid model could be classified as a high-level relay hybrid (HRH) approach. A solution is improved sequentially by executing a constraint programming model and a meta-heuristic. First, an initial solution is generated as input for the CP model, the required inputs are added to the CP model (lines 1 - 3), and the execution time and other parameters are specified (lines 4 - 5). Sequentially, the CP model and the meta-heuristic improve the initial solution (lines 7 - 15). The improved solution (s) generated by the CP model (line 7) is used as the initial solution for the meta-heuristic $MinCONvx(s, Time)$ (line 8). The meta-heuristic continues in the same way; it tries to further improve the solution s for a configurable time. It returns a solution (s'') that is evaluated if it is the best solution found so far (lines 9 - 11). If it is not, a perturbation of the best solution returned by the meta-heuristic (s'') is performed (lines 12 - 14). The perturbation strategy is based on a mode change of an activity and the inversion of a short sequence of activities randomly selected from the schedule. It always

Algorithm 5.1: The hybrid model for MRCMPSP combining a CP model and an extended meta-heuristic

Input: $s \leftarrow GenerateInitialSolution()$;
$s_{best} \leftarrow s$;
Output: An optimized feasible solution;
1 $AddTo\_CP\_Model(Projects, Activities, Modes, Resources)$;
2 $AddTo\_CP\_Model(AllConstraintsTypes)$;
`/* See equations (4.2), (4.3), (4.4), (4.7), (4.11), (4.12), (4.13) and (4.14) */`
3 $AddTo\_CP\_Model(ObjectiveFunction)$;
`/* See equation (4.15) */`
4 $Time \leftarrow 15 \quad sec$;
5 $CP.SetParameters(Time * 3, SearchType, RandomSeed)$;
6 **while** $(not \quad Termination\_Condition)$ **do**
7 $\quad s \leftarrow Execute\_CP\_Model(s)$;
8 $\quad s'' \leftarrow MinCONvx(s, Time)$;
9 $\quad$ **if** $(eval(s'') < eval(s_{best}))$ **then**
10 $\quad\quad s_{best} \leftarrow s''$;
11 $\quad\quad s \leftarrow s''$;
12 $\quad$ **else**
13 $\quad\quad act_i \leftarrow random.select.from(J_i)$;
14 $\quad\quad s \leftarrow RandomOneModeChange(s_{best}, act_i)$;
15 $\quad\quad s \leftarrow RandomInvertSubSequence(s)$;
16 $\quad$ **end**
17 **end**

produces feasible solutions. The perturbed solution serves as a starting point for the CP model. We experimented with the assigned execution time for each algorithm in a ratio of two to one and three to one in favour of the CP model within an execution sequence. Slightly better results were obtained with a ratio of three to one. We conducted separate experiments with two hybrid models containing each meta-heuristic variant: the hybrid model (HBv1) consisting of the MinCONv2 meta-heuristic and the CP model, and the hybrid model (HBv2) consisting of the MinCONv3 meta-heuristic and the CP model. A flowchart illustrates Algorithm 5.1 as shown in Figure 5.1.

5.2 Evaluation of the hybrid method

The hybrid method was evaluated through experimental tests with the MISTA 2013 Challenge MRCMPSP and MMLIB MRCPSP instances. We present detailed results in the following sections.

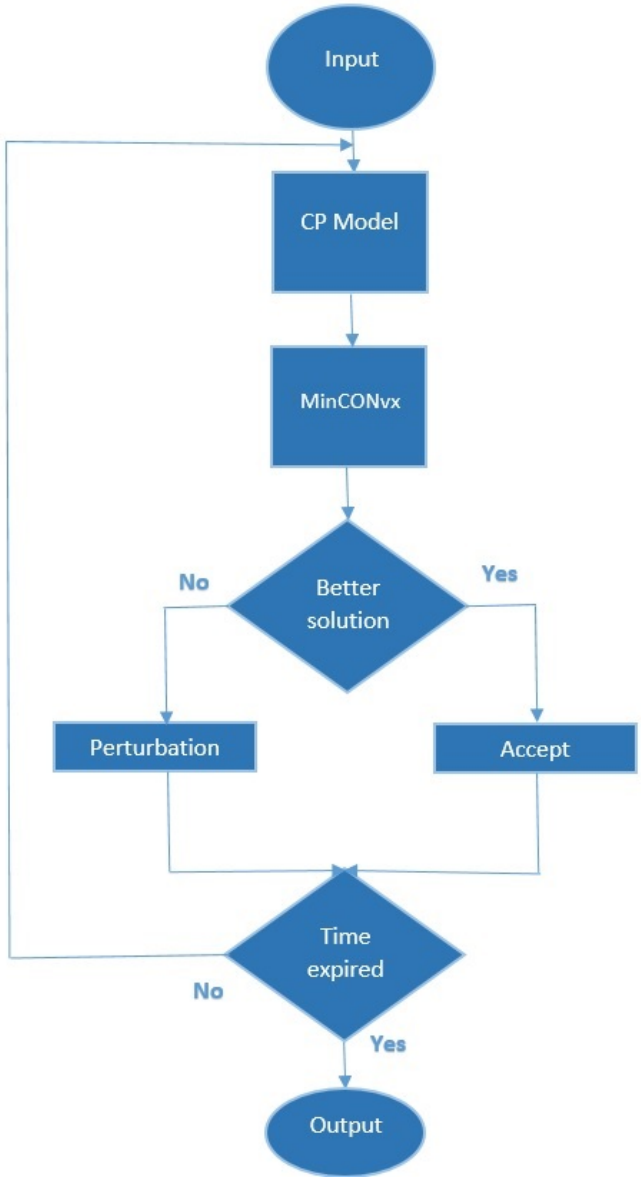

Figure 5.1: The hybrid model

5.2.1 Experiments with MRCMPSP benchmark instances - MISTA 2013 challenge instances

In experiments with a time limit, i.e. ten runs per instance and five minutes per run, our hybrid solver obtained a new upper bound for the MRCMPSP instance B2, better results for nine and equal results for four benchmark instances (see Table 5.1) compared with the results of the best solvers ([AKK+16]), [Gei17], and [TSCS16]. The solver of [AKK+16] performed better. It performed better for fifteen instances, and [Gei17] and [TSCS16]

each performed better by one instance. Average results and standard deviations are also

Table 5.1: Comparison of the HBv1 results (TPD/TMS) with the best solvers' results, under time limit constraints for MRCMPSP

Ins.	[AKK+16]	[Gei17]	[TSCS16]	HBv1
A1	**1/23**	-	-	**1/23**
A2	**2/41**	-	-	**2/41**
A3	**0/50**	-	-	**0/50**
A4	**65/42**	-	-	**65/42**
A5	153/105	-	-	**152/104**
A6	147/96	-	-	**144/92**
A7	**596/196**	-	-	615/205
A8	302/155	-	-	**276/150**
A9	223/119	-	-	**200/121**
A10	969/314	-	-	**921/313**
B1	**349/127**	353/125	363/132	364/126
B2	443/167	490/176	434/160	**419/162**
B3	**545/210**	598/215	660/207	566/212
B4	1292/287	**1274/289**	1548/295	1368/291
B5	**820/254**	866/254	919/254	887/262
B6	**912/227**	1044/242	1128/232	925/233
B7	**792/228**	834/234	908/246	859/239
B8	3176/533	3585/568	3276/529	**3145/561**
B9	**4192/746**	4674/796	5373/769	5262/884
B10	**3249/456**	3518/469	3325/447	3415/473
X1	398/142	394/142	**392/142**	408/142
X2	**349/163**	368/165	418/165	375/167
X3	324/192	372/195	326/188	**318/187**
X4	955/213	970/215	986/207	**939/210**
X5	**1768/374**	1938/386	2043/375	1878/386
X6	**719/232**	844/253	880/240	768/240
X7	**861/237**	879/231	944/234	890/235
X8	**1233/283**	1380/296	1478/289	1310/287
X9	**3268/643**	3645/688	4169/662	3840/718
X10	**1600/381**	1669/402	1851/385	1727/403

given (see Table 5.2).

Differences between results of our HBv1 solver and [AKK+16] under short time limit are presented graphically in Figure 5.2. It can be noticed that for very large instances, e.g. B9 and X9, the hyper-heuristic approach performs better compared with our hybrid method. This is due to the nature of the CP model that is a constituent part of the hybrid method; it performs very well for small and medium instances, but less well for

Table 5.2: Comparison of the results (TPD/TMS) of our hybrid approaches with the results of the best solver ([AKK+16]) for MRCMPSP

	[AKK+16]			**HBv1**			**HBv2**		
Ins.	**Best (TPD /TMS)**	Avg (TPD)	Std (TPD)	**Best (TPD /TMS)**	Avg (TPD)	Std (TPD)	**Best (TPD /TMS)**	Avg (TPD)	Std (TPD)
A1	**1/23**	-	-	**1/23**	1	0	**1/23**	1	0
A2	**2/41**	-	-	**2/41**	2	0	**2/41**	2	0
A3	**0/50**	-	-	**0/50**	0	0	**0/50**	0	0
A4	**65/42**	-	-	**65/42**	65	0	**65/42**	66	0
A5	153/105	-	-	**152/104**	165	8	**152/104**	162	7
A6	147/96	-	-	144/92	160	5	**141/92**	158	9
A7	**596/196**	-	-	615/205	639	2	621/201	641	14
A8	302/155	-	-	**276/150**	291	11	282/150	305	22
A9	223/119	-	-	200/121	215	11	**199/125**	211	10
A10	969/314	-	-	**921/313**	982	41	947/328	989	29
B1	**349/127**	352	-	364/126	379	11	361/128	375	9
B2	443/167	454	-	**419/162**	461	21	430/160	444	9
B3	**545/210**	554	-	566/212	601	19	555/208	599	23
B4	**1292/287**	1305	-	1368/291	1464	65	1368/291	1431	42
B5	**820/254**	833	-	887/262	923	18	862/259	933	47
B6	**912/227**	953	-	925/233	992	38	945/229	993	40
B7	**792/228**	801	-	859/239	921	49	863/241	926	32
B8	3176/533	3314	-	**3145/561**	3511	179	3312/556	3605	186
B9	**4192/746**	4264	-	5262/884	5740	320	5071/840	5470	244
B10	**3249/456**	3338	-	3415/473	3583	137	3324/463	3548	112
X1	**398/142**	405	-	408/142	432	21	405/141	432	15
X2	**349/163**	357	-	375/167	402	20	392/166	404	13
X3	324/192	330	-	**318/187**	346	19	336/188	357	17
X4	955/213	971	-	**939/210**	1033	53	1001/217	1051	48
X5	**1768/374**	1785	-	1878/386	1956	43	1933/383	2024	81
X6	**719/232**	738	-	768/240	840	72	782/242	844	53
X7	**861/237**	868	-	890/235	925	24	898/232	926	18
X8	**1233/283**	1257	-	1310/287	1452	92	1325/292	1379	45
X9	**3268/643**	3303	-	3840/718	4134	163	3835/708	4107	243
X10	**1600/381**	1614	-	1727/403	1777	36	1709/385	1755	33

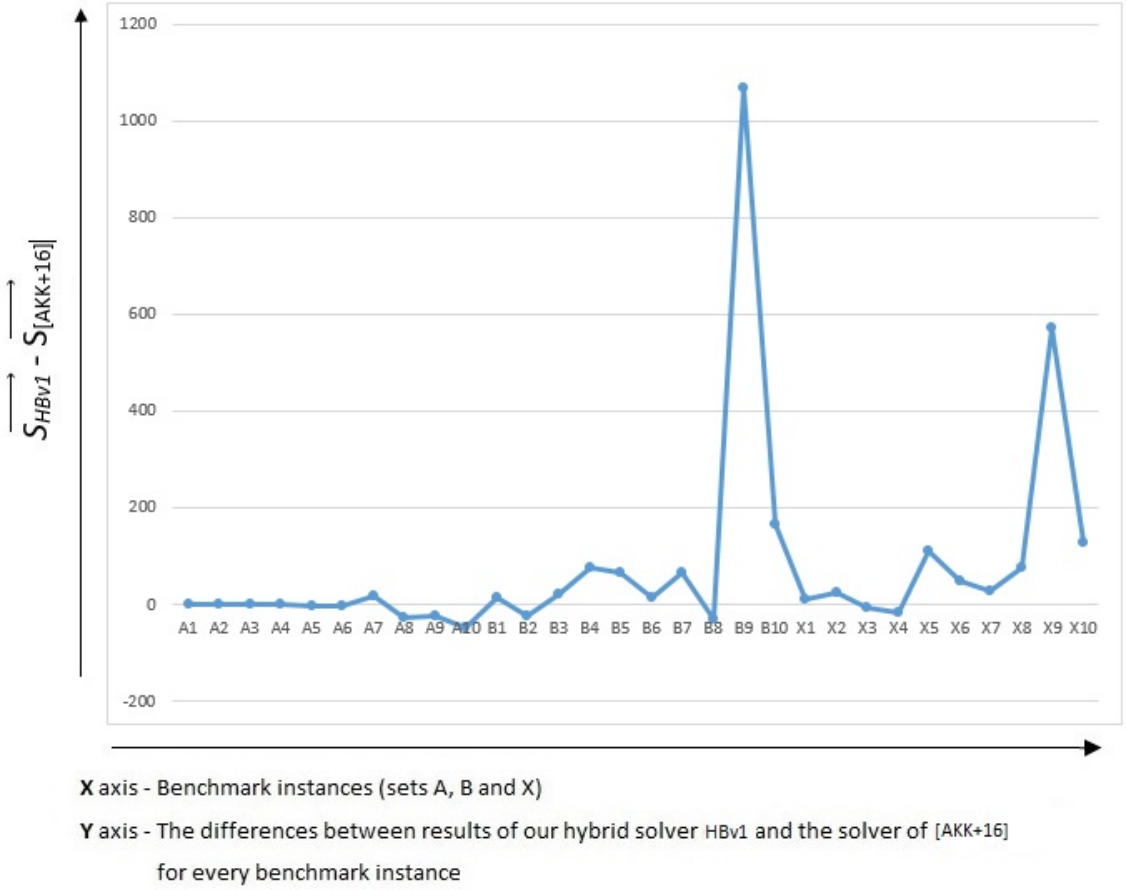

Figure 5.2: Comparison of our results (TPD/TMS) with the results of the best solver of the MISTA competition ([AKK$^+$16]) for MRCMPSP. The algorithms are executed under time limit constraints

very large instances.
The hybrid approach appears to be far more successful than executing the algorithms that comprise it separately. In the case of very large instances and time limit constraints, e.g. MISTA 2013 Challenge conditions, meta-heuristic approaches appear to be somewhat better. Under time relaxed conditions, our hybrid approach outperforms all solvers for most instances.

Our hybrid approach (HBv1) outperformed the best solver implemented by [AKK$^+$16] for the MRCMPSP for most instances, when algorithms were run under relaxed time limits. It generated new upper bounds for the majority in instances of this group (see Table 5.3). The solver presented in [AKK$^+$16] ran 2500 times for each instance and five minutes for each run. We executed our solver for twenty-four hours only once for each instance. Our algorithm converged much earlier for the majority of the instances within

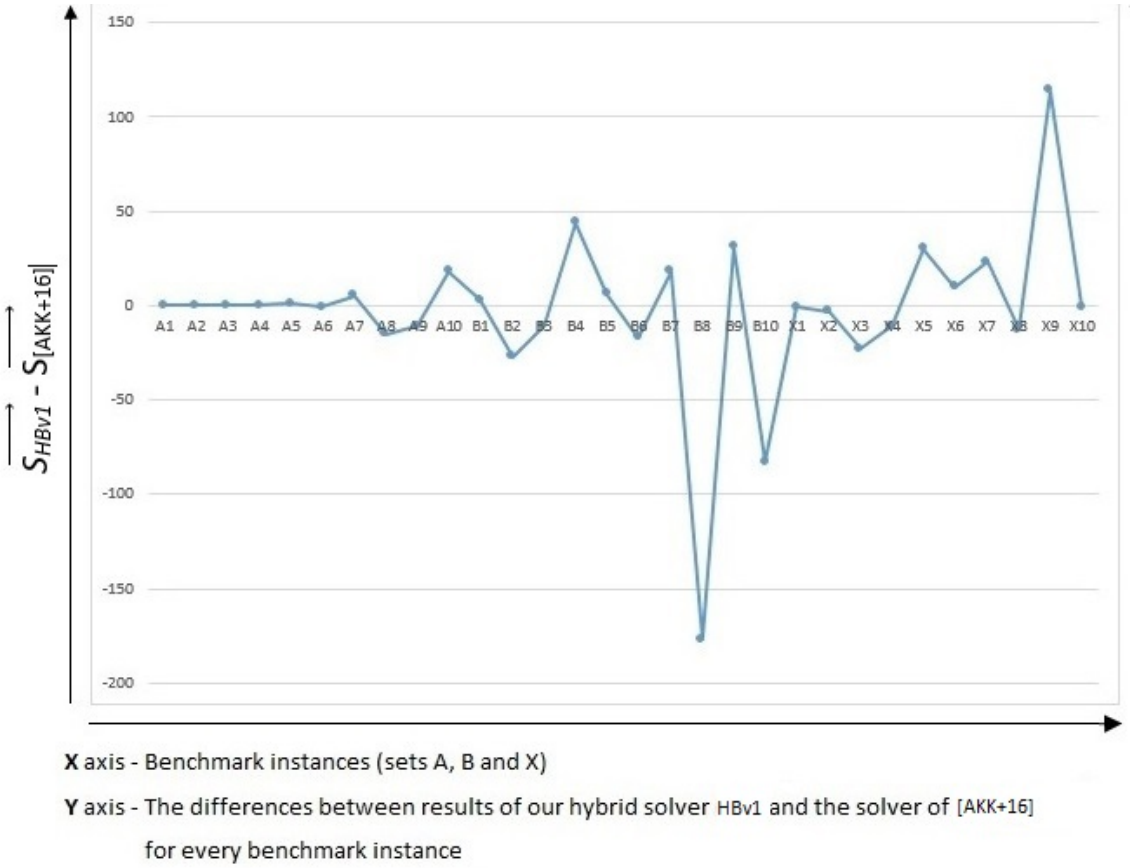

Figure 5.3: Comparison of our results (TPD/TMS) with the best solver's results ([AKK$^+$16]) for MRCMPSP. The algorithms are executed under time relaxed conditions

a few minutes. The comparison of the results of our hybrid method with those of the solver implemented by [AKK$^+$16], obtained under time relaxed constraints is also shown in Figure 5.3, where the differences between the results for each benchmark instance of both solvers are visually presented.

The comparison of HBv1 with HBv2 resulted that HBv1 achieved better results for the largest number of instances but worse average results and higher standard deviation (see Table 5.2). It is also noted that HBv1 dominates its constituent components, the CP model and MinConv2, giving better results for twenty-three out of thirty instances and equal results for three instances (see Table 5.4). The CP model provides better results for three instances, and MinConv2 has one better result. The MinConv2 gives better or equal results for twelve instances compared with the CP model. The HBv2 model dominates the CP model and MinConv3 by giving better results for twenty out of thirty instances and matching results for three instances. The CP model gives better results for three instances, and MinConv3 has one better result. MinConv3 gives better or equal

Table 5.3: Comparison of the results (TPD/TMS) of our hybrid approach with the previous best solution results for MRCMPSP under relaxed time limit

Ins.	**[AKK+16]**	**[Gei17]**	**[TSCS16]**	**HBv1**
A1	**1/23**	**1/23**	**1/23**	**1/23**
A2	**2/41**	**2/41**	**2/41**	**2/41**
A3	**0/50**	**0/50**	**0/50**	**0/50**
A4	**65/42**	**65/42**	68/50	**65/42**
A5	**150/103**	153/104	154/104	151/104
A6	133/99	144/94	151/94	**132/90**
A7	**590/190**	601/206	626/194	595/189
A8	272/148	319/162	281/147	**257/147**
A9	197/122	225/128	212/127	**186/122**
A10	**836/303**	920/313	983/309	854/307
B1	**345/124**	349/130	358/131	348/127
B2	431/158	481/171	431/159	**404/160**
B3	526/200	604/214	585/196	**515/204**
B4	**1252/275**	1283/287	1435/294	1296/283
B5	**807/245**	866/252	867/254	813/250
B6	905/225	1067/246	970/224	**888/219**
B7	**782/225**	827/232	876/234	800/233
B8	3048/523	3618/565	3001/520	**2871/525**
B9	**4062/738**	4606/783	4753/741	4093/736
B10	3140/436	3541/473	3123/430	**3057/437**
X1	386/137	-	392/142	**385/139**
X2	345/158	-	416/167	**342/163**
X3	310/187	-	332/177	**287/183**
X4	907/201	-	980/209	**896/204**
X5	**1727/362**	-	1904/369	1757/370
X6	**690/226**	-	821/237	700/232
X7	**831/220**	-	909/232	854/224
X8	1201/279	-	1389/281	**1188/279**
X9	**3155/632**	-	3945/639	3269/641
X10	1573/373	-	1718/377	**1572/374**

Table 5.4: Comparison of results (TPD/TMS) between the hybrid model, the CP Model, and the meta-heuristics for MRCMPSP. Algorithms are run under time limit

Ins.	HBv1 (TPD/TMS)	HBv2 (TPD/TMS)	CP Model (TPD/TMS)	MinConv2 (TPD/TMS)	MinConv3 (TPD/TMS)
A1	**1/23**	**1/23**	**1/23**	**1/23**	**1/23**
A2	**2/41**	**2/41**	**2/41**	**2/41**	**2/41**
A3	**0/50**	**0/50**	**0/50**	**0/50**	**0/50**
A4	**65/42**	**65/42**	65/45	**65/42**	**65/42**
A5	**152/104**	**152/104**	173/110	162/107	159/108
A6	144/92	**141/92**	162/104	156/94	152/96
A7	**615/205**	621/201	655/197	644/203	642/210
A8	**276/150**	282/150	279/148	337/166	348/159
A9	200/121	**199/125**	212/124	245/139	265/145
A10	**921/313**	947/328	1017/317	969/338	999/340
B1	364/126	361/128	375/130	**355/129**	360/130
B2	**419/162**	430/160	438/167	498/177	518/187
B3	566/212	**555/208**	586/206	639/230	634/224
B4	1368/291	1368/291	1493/286	1386/302	**1362/289**
B5	887/262	**862/259**	922/267	955/275	953/277
B6	**925/233**	945/229	936/227	1139/261	1116/255
B7	**859/239**	863/241	1003/252	890/251	924/258
B8	3145/561	3312/556	**3113/544**	3687/628	3785/619
B9	5262/884	**5071/840**	5253/833	5858/948	6291/981
B10	3415/473	3324/463	**3295/455**	3636/456	3836/507
X1	**408/142**	405/141	443/144	435/148	419/148
X2	**375/167**	392/166	405/167	419/175	398/170
X3	**318/187**	336/188	346/194	382/202	411/209
X4	**939/210**	1001/217	996/208	1035/221	1032/223
X5	**1878/386**	1933/383	1940/376	2083/393	2065/407
X6	**768/240**	782/242	799/243	967/281	1016/285
X7	**890/235**	898/232	902/233	951/245	946/246
X8	**1310/287**	1325/292	1366/288	1584/329	1562/329
X9	3840/718	**3835/708**	4320/760	4374/790	4351/779
X10	1727/403	**1709/385**	1739/396	1938/437	1903/433

results for thirteen instances compared with the CP model. From the overall comparison, HBv1 gives better results for fourteen instances, HBv2 gives better results for seven instances, the CP model gives better results for two instances, MinConv2 and MinConv3 give better results for one instance each. HBv1 and HBv2 give identical results for five instances. These results are shown graphically in Figure 5.4.

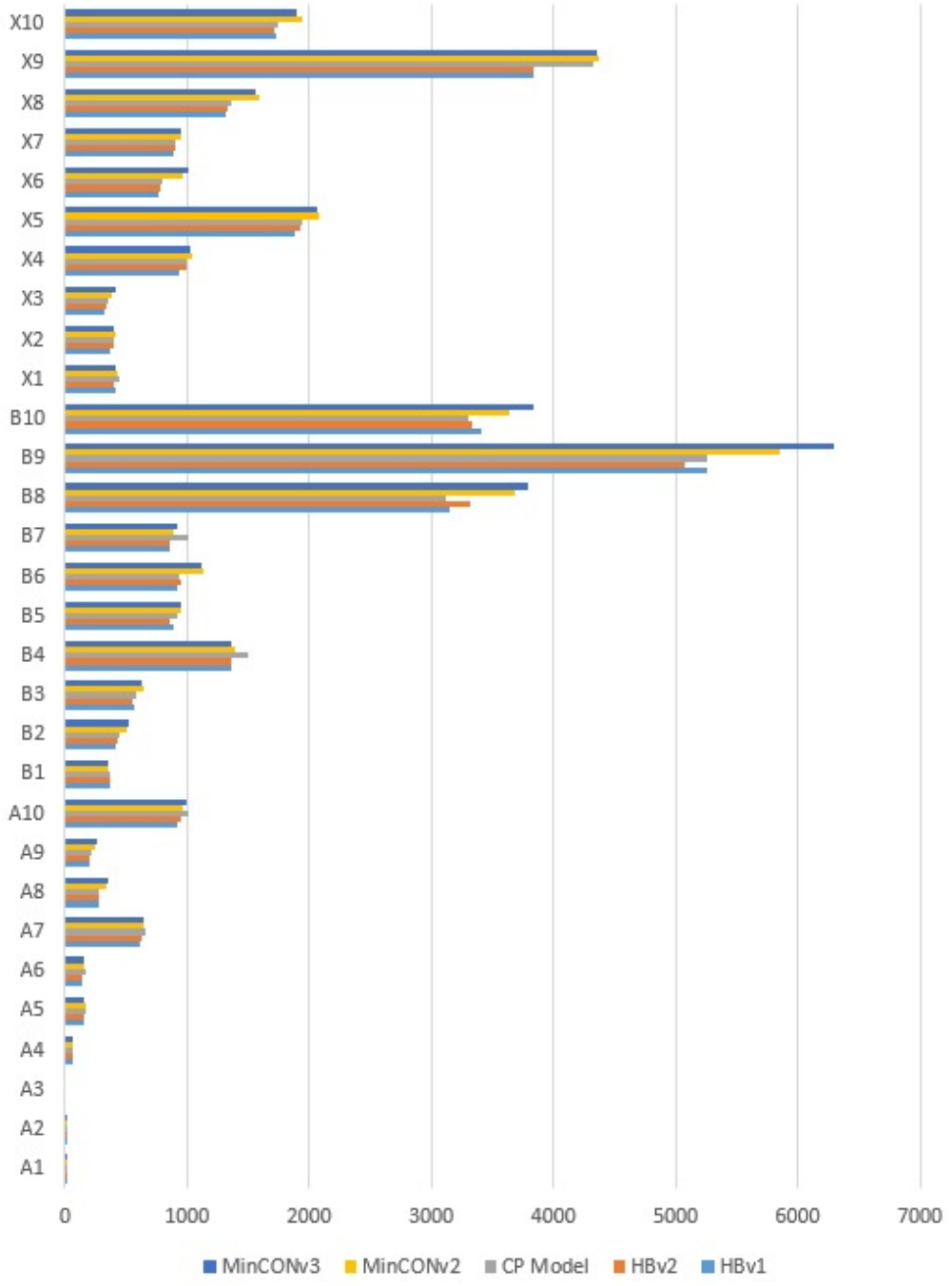

Figure 5.4: Comparison of results (TPD/TMS) between the hybrid model, the CP Model, and the meta-heuristics for MRCMPSP. Algorithms are run under time limit

5.2.2 Experiments with MRCPSP benchmark instances - MMLIB instances

We also evaluated the HBv1 solver on MMLIB benchmark instances (mmlib50, mmlib100 and mmlibPlus), and the experiments resulted in new upper bounds for fifty instances and equal results for many more compared with the results of state-of-the-art solvers (see Table 5.5). We executed the algorithm for thirty minutes for each instance. The results reported in this chapter were achieved during the tests conducted in December 2019. The work presented in this chapter is published in [AM21].

Table 5.5: New upper bounds for MRCPSP achieved by the hybrid method where the makespan (MS) is the objective (December 2019)

MMLIB Instances					
mmlib50		**mmlib100**		**mmlibPlus**	
Ins.	Makespan	Ins.	Makespan	Ins.	Makespan
J507_1.mm	43	J1008_5.mm	49	Jall146_1.mm	64
J507_5.mm	40	J10045_3.mm	53	Jall185_3.mm	107
J5043_5.mm	63	J10074_4.mm	75	Jall193_3.mm	75
J5045_4.mm	35	J10076_3.mm	58	Jall254_3.mm	81
J5046_5.mm	38	J10079_1.mm	128	Jall256_3.mm	113
J5047_5.mm	38	J10080_5.mm	83	Jall302_1.mm	47
J5048_2.mm	39	J10081_1.mm	85	Jall344_4.mm	83
J5080_5.mm	69	J10081_2.mm	74	Jall346_2.mm	78
-	-	J10081_3.mm	70	Jall347_3.mm	57
-	-	J10082_3.mm	78	Jall363_1.mm	57
-	-	J10082_4.mm	94	Jall371_1.mm	84
-	-	J10083_2.mm	79	Jall372_3.mm	72
-	-	J10083_3.mm	71	Jall374_4.mm	66
-	-	J10084_3.mm	78	Jall394_2.mm	102
-	-	J10084_5.mm	73	Jall399_1.mm	237
-	-	J10092_2.mm	80	Jall401_2.mm	193
-	-	J10092_5.mm	71	Jall410_5.mm	62
-	-	J10094_1.mm	54	Jall458_3.mm	79
-	-	-	-	Jall509_1.mm	138
-	-	-	-	Jall512_1.mm	132
-	-	-	-	Jall537_3.mm	133
-	-	-	-	Jall537_4.mm	104
-	-	-	-	Jall556_2.mm	99
-	-	-	-	Jall566_5.mm	124

5.3 Discussion and analysis

In this chapter, we investigated two hybrid approaches for MRCMPSP: the hybrid model comprised of the CP model and the MinCONv2 variant of the meta-heuristic called HBv1, and the combination with the MinCONv3 variant of the meta-heuristic called HBv2. We performed separate experiments with both solvers on existing MRCMPSP benchmark instances and compared the results with the state-of-the-art solver for this problem.

Extensive testings were performed in various experimental settings. The hybrid solver HBv1 produced new upper bounds for nine and equal results for four out of thirty benchmark instances compared with the best solver for MRCMPSP [AKK+16]. The hybrid solver HBv1 also achieved fourteen better results and six matching results compared with HBv2. Under time constraints, MinCONv3, a constituent part of HBv2, achieved slightly better results than MniCONv2, which constitutes HBv1. Consequently, it seems that the meta-heuristic solver with a more stochastic nature forms a more efficient combination with an exact solver. Nevertheless, the HBv2 achieved better average results and lower standard deviation for more instances. Additionally, we tested our approach with MMLIB benchmark instances, and experiments resulted in new upper bounds for fifty instances.

Our main objective was to evaluate a hybrid approach that combines two complementary search strategies. This study introduces a successful combination of an exact model (CP model) with a meta-heuristic approach and a particular perturbation mechanism. Considering that both hybrid models obtained better results than the algorithms that compose them, we believe that it is of high interest to investigate different hybrid approaches that combine different search strategies to create a robust and efficient method. In particular, the role of meta-heuristics in our hybrid method should be further studied. In our case, we noticed that the hyper-heuristic approach still performed better than our hybrid approach for very large instances, e.g., B9 and X9, under short time constraints. Since the exact model relies on the large neighborhood search and performs very well on small and medium instances, it is up to the meta-heuristic component to perform an efficient search within very short time constraints in order to provide excellent results even on very large instances.

Furthermore, in the above experiments, we found that improving the complementary search nature of algorithms forming a potential hybrid method would be of great interest. In this context, if we analyze the results and behaviour of the HBv1 again, it turns out that it achieves better results than HBv2, even though constituting components of HBv2 have achieved better results compared with their counterparts in HBv1. Recall that the main difference between HBv1 and HBv2 is their meta-heuristic component, MinCONv2 and MinCONv3. The former representing a more stochastic version than the latter. Therefore, at this stage, we also considered implementing a simulated annealing based approach as a new meta-heuristic. On the one hand, this approach helps to avoid getting stuck in a local optimum and, on the other hand, it increases the stochastic nature of the algorithm. The next chapter will give a complete insight into this implementation.

CHAPTER 6

Integration of the Simulated Annealing in the Hybrid Approach for MRCMPSP

This chapter introduces the implementation of the simulated annealing heuristic for MRCMPSP and its integration into our hybrid approach HBv1 described in the previous chapter. Simulated annealing is added to MinCONv2 meta-heuristic as a new neighborhood operator. It is invoked only when the search gets stuck in a local optimum. This extended meta-heuristic is referred to as MinCONv4. Additionally, this chapter investigates a new hybrid model resulting from the combination of the CP model described in Chapter 4 and MinCONv4. The new hybrid model is referred to as HBv3. Initially, MinCONv4 is evaluated on MISTA 2013 challenge benchmark instances, and then HBv3 is evaluated on MISTA 2013 challenge, MMLIB, and RCMPSP instances and compared with the state-of-the-art solvers.

6.1 Simulated annealing for MRCMPSP

We have implemented SA in three variants, depending on the types of neighborhoods they generate:

- The first variant is based on changing the execution mode of an activity.
- The second variant shifts the position of a randomly selected activity in the schedule in the direction of its first predecessor.
- The third variant shifts the position of a randomly selected activity in the schedule in the direction of its first successor.

The implementation of the first variant is shown in Algorithm 6.1. As can be seen, first the parameters for the maximum temperature (*Tmax*) and the temperature drop step (t_{step}) are initialized (see lines 2-3). Then, an activity is randomly selected from a set of activities (see line 5), and then all its execution modes are changed in turn (see lines 6-7). Only better solutions or solutions that satisfy the Metropolis criterion are accepted (see lines 8-11). The temperature is gradually decreased until a certain threshold is reached (see lines 13-14), otherwise it is heated again. The whole procedure is executed until a certain time threshold is reached. The other two variants are implemented similarly, except that instead of the procedure that changes the execution mode of an activity (see line 5), the actual procedures are called that change the positions of randomly selected activities in the schedule.

Algorithm 6.1: Simulated annealing algorithm, SA_1(s, startSeq, endSeq)

Input: $s \in \{\overrightarrow{S}\}$, *a feasible schedule*;
Output: An improved solution;

```
1  repeat
2      T ← T_max;
3      t ← t_step;
4      repeat
5          a ← Random(startSeq, endSeq);
6          foreach mode_of_a do
7              s' ← OneModeChange(s, a);
8              prob ← Random.Double[0, 1);
9              if (eval(s') < eval(s)) or (prob < e^(−|eval(s')−eval(s)|/T)) then
10                 s ← s';
11             end
12         end
13         T ← T * t;
14     until T < 1;
15     T ← T_max;
16 until TimeElapsed;
```

The simulated annealing algorithm has been added to MinCONv2 meta-heuristic as a new neighborhood operator. The resulting meta-heuristic, referred to as MinCONv4, is shown in Algorithm 6.2. Simulated annealing is invoked only when the search gets stuck in a local optimum, i.e. when the parameter *SA-activationFreq*, which represents the number of the search iterations without improvements, reaches its threshold (see lines 11-13). Since our algorithm operates in a multi-threaded environment, two threads are used to execute the first simulated annealing variant, while only one thread is used to execute each of the other variants.

Other neighborhood operators used in the Algorithm 6.2 are the same as those described in Chapter 3: one-mode change (MinConOMC), two-mode change (MinConTMC), shifting

Algorithm 6.2: The algorithm MinCONv4

Input: $\overrightarrow{S}_i \in \{\overrightarrow{S}\}$, initial schedules generated from earlier stages;
Output: An optimal local solution;

NoImprovement = true; LocalImprovement = true;
repeat
  repeat
    Apply combinations of *CloneProj(), CloneProjPart(), ComE(), ComF()*;
    if *(BroadcastBestLocal())* **then**
      NoImprovement = false;
    else
      NoImprovement = true;
    end
  until *NoImprovement*;
  if $SAactivationFreq > threshold$ **then**
    $SA_x(\overrightarrow{S}_i, sSeq_i, eSeq_i)$;
  end
  if *LocalImprovement* **then**
    do
      Apply *MinConOMC()*;
      if *(BroadcastBestLocal())* **then**
        LocalImprovement = false;
      else
        LocalImprovement = true;
      end
    while *LocalImprovement*;
    Apply *PCom(), MinConTMC()*;
  else
    Apply *SwapAndModeCh(), MinConSJL(), MinConSJR(), INVS ()*;
  end
  BroadcastBestLocal();
  if $\overrightarrow{S}_{local} > \min_{i\in\{1,..,4\}} \overrightarrow{S}_i$ **then**
    $\overrightarrow{S}_{local} = \min_{i\in\{1,..,4\}} \overrightarrow{S}_i$;
  end
  if $\overrightarrow{S}_{local} < \overrightarrow{S}_{best}$ **then**
    $\overrightarrow{S}_{best} = \overrightarrow{S}_{local}$; *NoImprovement = false*;
  else
    NoImprovement = true;
  end
  LocalImprovement = NoImprovement;
  if *LocalImprovement = false* **then**
    LocalImprovement = true; perturbationSize += 1; *SAactivationFreq* += 1;
  else
    perturbationSize = 1; *SAactivationFreq* = 1;
  end
  Reset $\overrightarrow{S}_{local}$; Perturbate($\overrightarrow{S}_i$, *perturbationSize*);
until *timeExpired*;

an activity to its last predecessor (MinConSJL), shifting an activity to its first successor (MinConSJR), inverting a subsequence of activities (INVS), swapping an activity and a one-mode change (SwapAndModeCh), compressing a project and moving it to the end (ComE), compressing a project and moving it to the beginning (ComF), cloning a project (CloneProj), partially cloning a project (CloneProjPart), and cloning a sequence from a project (CloneSeq). A flowchart for Algorithm 6.2 is shown in Figure 6.1.

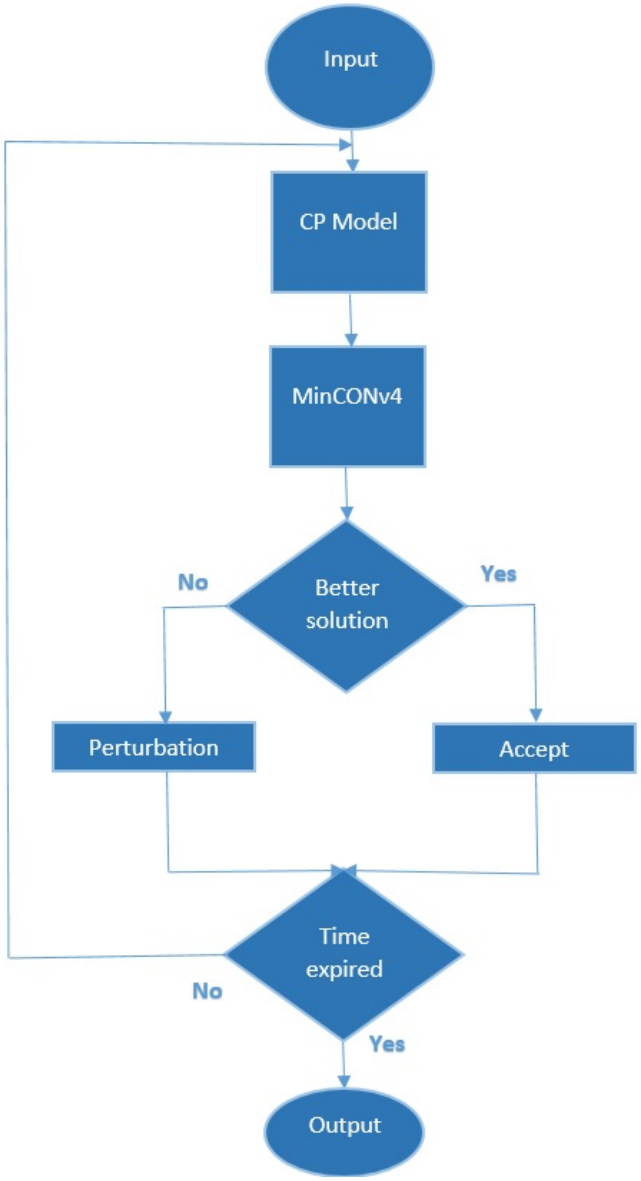

Figure 6.1: The new hybrid solver

6.1.1 Acceptance criteria and perturbation

We accept only better solutions and apply an adaptive perturbation strategy depending on the instance size. The perturbation consists in changing the modes of up to 10 % of the randomly selected activities of the current schedule. When simulated annealing is executed, it is possible to accept slightly worse solutions to escape the local minimum.

6.1.2 Parameter tuning

The parameters used in the algorithm 6.2 are shown in the table 6.1, where: *ModeTBLength* represents a dimension of the tabu list for neighborhood operators manipulating modes of activities. It is determined as a percentage of the total number of activities in a given instance. *SeqTBLength* represents a dimension of the tabu list for neighborhood operators that manipulate positions of activities in the schedule and it is determined as a percentage of the total number of activities in a given instance, *VarSetSize* represents the size of variable set for the *MinConTMC* operator, *PertSizeThreshold* is the threshold for the size of the perturbation, *Tmax* is the maximum temperature in the simulated annealing algorithm, t_{step} is the temperature decrease step, and *TimeElapsed* is the time in which the Simulated Annealing is executed. The parameter *SA-activationFreq* represents the number of the search iterations without improvements before the execution of the simulated annealing. Similarly, *seqPerturbationFreq* represents the frequency of running the entire hybrid solver without any improvements before performing perturbation based on the change in the randomly selected activity position. The values of the parameters were fine-tuned using the SMAC tool [LEF+17, HHL11].

Table 6.1: Parameters used for tests

Parameter	Value	Domain of values
ModeOp-Tabu-List-Length	30%	{10%, 15%, 20%, 25%, 30%, 35%, 40%}
SeqOp-Tabu-List-Length	20%	{10%, 15%, 20%, 25%, 30%, 35%, 40%}
VarSetSize	11	{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
PertSizeThreshold	10%	{3%, 5%, 7%, 10%, 13%, 15%, 20%}
Tmax	300	{100, 300, 500, 800, 1000, 3000}
t_{step}	0.9	{0.8, 0.85, 0.9, 0.95}
TimeElapsed	2	{1, 2, 3, 5, 10}
SA-activationFreq	6	{3, 6, 8, 10, 12, 15}
seqPerturbationFreq	11	{3, 5, 7, 9, 11, 13, 15}

6.2 Computational results

We tested and validated our approach with benchmark instances of three types of scheduling problems: MRCMPSP benchmark instances used in the MISTA 2013 challenge,

well-known MMLIB MRCPSP instances, and MPSPLIB RCMPSP [1] instances.

6.2.1 Experiments with MRCMPSP benchmark instances - MISTA 2013 challenge instances

The MinCONv4 slightly improves the results for MRCMPSP instances compared with MinCONv2 (see Table 6.2). Moreover, the new hybrid solver (HBv3), which is composed of the CP model and the new meta-heuristic component (MinCONv4), provides new upper bounds for fourteen MRCMPSP instances out of thirty. It outperforms the best solvers for MRCMPSP for most instances under relaxed time limits (see Table 6.3). The same results (TPD only) are shown in a friendlier way in Figure 6.2. Our HBv3 solver outperforms the hyper-heuristic solver implemented by [AKK+16] by giving better results for sixteen and matching results for four instances, and the hybrid solver HBv1 by giving better results for fifteen and matching results for five out of thirty instances. The [AKK+16] results shown in this table were obtained after 2500 executions of the algorithm for each instance, with each execution taking 5 minutes. We ran our solver for a much shorter time, two hours, with only one run for each instance. Our algorithm converged much earlier, within a few minutes, for most instances. The comparison of the results of our HBv3 solver with those of the [AKK+16] solver is shown in Figure 6.3.

In terms of very short execution time (five minutes per run), the algorithm of [AKK+16] still converges faster than our hybrid solver for most instances. Our algorithm gave better results for eight instances and matching results for four out of thirty instances (see Table 6.4).

These results are also shown graphically in Figure 6.4.

6.2.2 Experiments with MRCPSP benchmark instances - MMLIB instances

In addition, we tested our new hybrid solver with known MMLIB instances, obtaining 49 new upper bounds (see table 6.5) and matching results for 2429 instances compared to the previous best solvers. The results reported in this chapter were obtained during tests conducted in June 2021.

6.2.3 Experiments with RCMPSP benchmark instances - MPSPLIB instances

The MPSPLIB RCMPSP problem instances are multi-project single-mode resource-constrained instances. This library is comprised of instances consisting of 2 to 20 projects, 20 instances with projects consisting of 30 activities, 60 instances with projects consisting of 90 activities, and 60 instances with projects consisting of 120 activities. The objective functions for these problems are:

[1] http://www.mpsplib.com/ranking.php

Table 6.2: Comparison of MinCONv4 results (TPD/TMS) with MinCONv3 results for MRCMPSP under time limit constraints

	MinCONv3			MinCONv4		
Ins.	**Best (TPD/TMS)**	**Avg (TPD)**	**Std (TPD)**	**Best (TPD/TMS)**	**Avg (TPD)**	**Std (TPD)**
A1	**1/23**	1	0	**1/23**	1	0
A2	**2/41**	2	0	**2/41**	2	0
A3	**0/50**	0	0	**0/50**	0	0
A4	**65/42**	66	0	**65/42**	66	0
A5	159/108	168	5	**159/104**	168	4
A6	**152/96**	162	6	154/100	167	7
A7	**642/210**	665	15	646/217	667	15
A8	348/159	369	17	**339/166**	373	18
A9	265/145	285	11	**252/136**	280	17
A10	999/340	1044	37	**977/332**	1051	37
B1	360/130	370	5	**359/128**	371	9
B2	**518/187**	538	14	529/185	559	20
B3	**634/224**	662	23	649/227	667	25
B4	**1362/289**	1444	38	1405/298	1461	37
B5	**953/277**	1003	30	976/277	1029	35
B6	**1116/255**	1202	41	1176/259	1263	47
B7	924/258	973	26	**917/241**	951	20
B8	3785/619	4169	190	**3746/609**	4101	264
B9	6291/981	6803	340	**5730/947**	6263	311
B10	3836/507	3976	121	**3788/520**	3940	115
X1	**419/148**	452	13	430/152	464	17
X2	**398/170**	436	22	422/178	439	13
X3	411/209	433	12	**408/206**	429	9
X4	1032/223	1102	29	**1029/218**	1095	43
X5	2065/407	2223	84	**2012/407**	2191	96
X6	1016/285	1062	43	**924/273**	1089	72
X7	**946/246**	999	23	952/245	1018	36
X8	1562/329	1767	108	**1536/328**	1691	67
X9	**4351/779**	4812	237	4430/783	4797	364
X10	**1903/433**	2124	110	2001/435	2158	104

- Minimization of Average Project Delay (APD)
- Minimization of Total Makespan (TMS)
- Minimization of Standard Deviation of the Project Delay (DPD)

Table 6.3: Comparison of the new hybrid solver's results (TPD/TMS) with results of the best solvers for MRCMPSP under relaxed time limit

Ins.	[AKK+16]	[Gei17]	[TSCS16]	HBv1	HBv3
A1	**1/23**	**1/23**	**1/23**	**1/23**	**1/23**
A2	**2/41**	**2/41**	**2/41**	**2/41**	**2/41**
A3	**0/50**	**0/50**	**0/50**	**0/50**	**0/50**
A4	**65/42**	**65/42**	68/50	**65/42**	**65/42**
A5	**150/103**	153/104	154/104	151/104	152/104
A6	133/99	144/94	151/94	**132/90**	133/90
A7	**590/190**	601/206	626/194	595/189	595/203
A8	272/148	319/162	281/147	257/147	**251/146**
A9	197/122	225/128	212/127	186/122	**184/121**
A10	**836/303**	920/313	983/309	854/307	889/314
B1	**345/124**	349/130	358/131	348/127	348/127
B2	431/158	481/171	431/159	404/160	**399/159**
B3	526/200	604/214	585/196	515/204	**509/203**
B4	1252/275	1283/287	1435/294	1296/283	**1247/280**
B5	807/245	866/252	867/254	813/250	**804/251**
B6	905/225	1067/246	970/224	888/219	**830/221**
B7	**782/225**	827/232	876/234	800/233	808/238
B8	3048/523	3618/565	3001/520	2871/525	**2724/513**
B9	**4062/738**	4606/783	4753/741	4093/736	4111/736
B10	3140/436	3541/473	3123/430	3057/437	**2901/426**
X1	386/137	-	392/142	**385/139**	388/141
X2	345/158	-	416/167	342/163	**341/163**
X3	310/187	-	332/177	**287/183**	292/186
X4	907/201	-	980/209	896/204	**895/206**
X5	1727/362	-	1904/369	1757/370	**1722/365**
X6	690/226	-	821/237	700/232	**670/224**
X7	**831/220**	-	909/232	854/224	858/228
X8	1201/279	-	1389/281	1188/279	**1180/277**
X9	**3155/632**	-	3945/639	3269/641	3211/643
X10	1573/373	-	1718/377	**1572/374**	1593/378

Our new hybrid solver outperformed all solvers for most instances of this problem. It obtained 81 new upper bounds out of 140 for RCMPSP instances and identical results in 28 of them compared to the previous best solvers (see Table 6.6). The target for these problems was to minimize three objective functions: the minimization of APD, the minimization of TMS, and the minimization of the DPD. The listing in Table 6.6 was designed with APD as the primary objective, but for the instances for which we have given new upper bounds, our algorithm has the best results for both TMS and DPD for

Table 6.4: Comparison of the HBv3 results (TPD/TMS) with [AKK+16] solver results, under time limit constraints for MRCMPSP

	Asta			HBv3		
Ins.	**Best (TPD/TMS)**	**Avg (TPD)**	**Std (TPD)**	**Best (TPD/TMS)**	**Avg (TPD)**	**Std (TPD)**
A1	**1/23**	-	-	**1/23**	1	0
A2	**2/41**	**-**	**-**	**2/41**	2	0
A3	**0/50**	-	-	**0/50**	0	0
A4	**65/42**	-	-	**65/42**	65	0
A5	**153/105**	-	-	154/103	163	6
A6	147/96	-	-	**145/94**	159	5
A7	**596/196**	-	-	612/198	633	16
A8	302/155	-	-	**276/153**	294	10
A9	223/119	-	-	**195/121**	209	8
A10	969/314	-	-	**918/315**	968	29
B1	**349/127**	352	-	364/129	375	7
B2	443/167	454	-	**436/163**	459	13
B3	**545/210**	554	-	587/210	600	10
B4	**1292/287**	1305	-	1356/285	1442	49
B5	**820/254**	833	-	861/259	902	30
B6	**912/227**	953	-	951/230	1021	55
B7	**792/228**	801	-	868/236	922	40
B8	3176/533	3314	-	**3175/531**	3547	202
B9	**4192/746**	4264	-	5057/816	5248	159
B10	**3249/456**	3338	-	3287/457	3565	119
X1	**398/142**	405	-	412/144	430	13
X2	**349/163**	357	-	386/167	410	14
X3	324/192	330	-	**323/189**	352	27
X4	955/213	971	-	**939/206**	1047	36
X5	**1768/374**	1785	-	1909/371	1999	51
X6	**719/232**	738	-	756/238	806	36
X7	**861/237**	868	-	908/228	937	21
X8	**1233/283**	1257	-	1351/287	1409	41
X9	**3268/643**	3303	-	3765/694	4076	213
X10	**1600/381**	1614	-	1702/394	1768	53

the majority of the instances.

Table 6.5: New upper bounds for MRCPSP achieved by the hybrid method where the makespan (MS) is the objective (June 2021)

MMLIB Instances					
mmlib50		**mmlib100**		**mmlibPlus**	
Ins.	Makespan	Ins.	Makespan	Ins.	Makespan
J507_5.mm	40	J1009_5.mm	34	Jall37_4.mm	52
J508_1.mm	36	J10046_4.mm	63	Jall38_1.mm	48
J508_5.mm	41	J10056_1.mm	42	Jall59_5.mm	59
-	-	J10056_4.mm	45	Jall66_3.mm	118
-	-	J10073_1.mm	80	Jall94_5.mm	96
-	-	J10074_2.mm	82	Jall145_1.mm	94
-	-	J10074_3.mm	69	Jall154_3.mm	147
-	-	J10080_2.mm	92	Jall159_1.mm	57
-	-	J10081_5.mm	83	Jall184_3.mm	155
-	-	J10082_3.mm	78	Jall185_3.mm	107
-	-	J10082_1.mm	79	Jall185_5.mm	116
-	-	J10084_2.mm	81	Jall190_2.mm	95
-	-	J10092_3.mm	65	Jall192_3.mm	62
-	-	-	-	Jall203_5.mm	70
-	-	-	-	Jall204_2.mm	65
-	-	-	-	Jall208_1.mm	237
-	-	-	-	Jall208_3.mm	259
-	-	-	-	Jall212_5.mm	111
-	-	-	-	Jall229_1.mm	99
-	-	-	-	Jall237_5.mm	82
-	-	-	-	Jall263_4.mm	145
-	-	-	-	Jall265_2.mm	180
-	-	-	-	Jall290_2.mm	129
-	-	-	-	Jall298_1.mm	98
				Jall320_1.mm	110
				Jall372_3.mm	72
				Jall399_2.mm	212
				Jall399_5.mm	213
				Jall400_3.mm	213
				Jall457_3.mm	121
				Jall564_4.mm	86
				Jall566_3.mm	133
				Jall619_2.mm	166

6.3 Discussion and analysis

In this chapter, we presented the implementation of the simulated annealing heuristic for the MRCMPSP in three variants, depending on the types of neighborhoods they generate. Moreover, we integrated it with an existing hybrid solver consisting of a CP model and a meta-heuristic. Our approach has been evaluated on several types of project scheduling benchmark instances with different objectives: MISTA 2013 Challenge MRCMPSP, MMLIB MRCPSP, and MPSPLIB RCMPSP instances. As mentioned earlier, the objective for the MRCMPSP was to minimize TPD and TMS, having TPD as the main objective. The objective for MMLIB MRCPSP was to minimize the project makespan, and the target for MPSPLIB RCMPSP was to minimize three objective functions: the minimization of APD, the minimization of the minimization of TMS, and the minimization of the DPD. Our algorithm achieved new upper bounds for many benchmark instances and outperformed state-of-the-art solvers for most instances of these problems. It achieved new upper bounds for fourteen out of thirty MRCMPSP instances and outperformed the best hyper-heuristic solver implemented by [AKK+16] as well as the hybrid solver HBv1 for most instances. For known MMLIB instances, our approach achieved 49 new upper bounds and matching results for 2429 instances compared to the previous best solvers. In the case of MPSPLIB RCMPSP, our new hybrid solver obtained 81 new upper bounds out of 140 RCMPSP instances and identical results in 28 of them compared to the previous best solvers.

Our approach proved to be very promising for several variants of the project scheduling problem. There is still room for improvement, especially in the direction of the meta-heuristic component of our approach. For example, in our experiment, we found that in the case of very large instances, the hyper-heuristic approach of [AKK+16] still converges faster than our hybrid solver.

Table 6.6: New upper bounds for MPSPLIB RCMPSP instances obtained by HBv3 (June 2021)

Id	Instance	No. jobs	No. proj.	Gl. Res.	APD	TMS	DPD
1	mp_j30_a10_nr1	30	10	2	78.8	191	52.721
2	mp_j30_a10_nr2	30	10	1	7.6	108	10.384
3	mp_j30_a10_nr4	30	10	3	15.4	156	25.838
4	mp_j30_a10_nr5	30	10	1	49.1	190	44.817
5	mp_j30_a20_nr1	30	20	2	184.75	426	118.893
6	mp_j30_a20_nr2	30	20	1	69.3	282	55.876
7	mp_j30_a20_nr3	30	20	2	92.5	317	66.731
8	mp_j30_a20_nr4	30	20	3	26.3	192	43.059
9	mp_j30_a5_nr2	30	5	1	11.6	78	7.603

10	mp_j30_a5_nr3	30	5	2	25.6	109	11.104
11	mp_j30_a5_nr5	30	5	1	14	97	12.981
12	mp_j90_a10_nr3	90	10	2	38.5	225	32.477
13	mp_j90_a10_nr5	90	10	1	48.7	248	51.582
14	mp_j90_a2_nr2	90	2	1	19	117	24.042
15	mp_j90_a20_nr2	90	20	1	2.5	163	4.674
16	mp_j90_a20_nr4	90	20	3	24.8	186	28.929
17	mp_j90_a20_nr5	90	20	1	37.55	243	46.957
18	mp_j90_a5_nr5	90	5	1	11.6	151	17.038
19	mp_j120_a10_nr1	120	10	2	35.7	138	25.893
20	mp_j120_a10_nr2	120	10	1	60.9	254	53.716
21	mp_j120_a10_nr3	120	10	2	2.7	152	4.373
22	mp_j120_a10_nr5	120	10	1	170.5	502	126.615
23	mp_j120_a2_nr1	120	2	2	32.5	171	40.305
24	mp_j120_a2_nr4	120	2	3	39.5	147	10.607
25	mp_j120_a20_nr1	120	20	2	2.05	76	0.224
26	mp_j120_a5_nr3	120	5	2	61.2	216	39.720
27	mp_j120_a5_nr5	120	5	1	70.8	267	73.761
28	mp_j90_a10_nr5_AC10	90	10	4	49.6	180	43.638
29	mp_j90_a10_nr5_AC2	90	10	4	267.7	761	232.398
30	mp_j90_a10_nr5_AC3	90	10	4	49.8	269	60.494
31	mp_j90_a10_nr5_AC5	90	10	4	124.4	366	88.434
32	mp_j90_a10_nr5_AC7	90	10	4	124.6	404	123.384
33	mp_j90_a10_nr5_AC8	90	10	4	66.4	273	75.482
34	mp_j90_a10_nr5_AC9	90	10	4	78.9	235	57.717
35	mp_j90_a20_nr5_AC1	90	20	4	130.35	468	111.280
36	mp_j90_a20_nr5_AC10	90	20	4	206.4	489	138.073
37	mp_j90_a20_nr5_AC3	90	20	4	8	160	16.371
38	mp_j90_a20_nr5_AC4	90	20	4	115.6	355	82.353
39	mp_j90_a20_nr5_AC5	90	20	4	5.75	129	7.840
40	mp_j90_a20_nr5_AC6	90	20	4	55.75	256	56.550
41	mp_j90_a20_nr5_AC7	90	20	4	68.25	278	75.201
42	mp_j90_a20_nr5_AC8	90	20	4	19.1	160	30.017
43	mp_j90_a20_nr5_AC9	90	20	4	169.45	422	114.368
44	mp_j90_a2_nr5_AC1	90	2	4	61.5	191	64.347
45	mp_j90_a2_nr5_AC10	90	2	4	20	106	26.870
46	mp_j90_a2_nr5_AC2	90	2	4	172.5	330	118.087
47	mp_j90_a2_nr5_AC3	90	2	4	36	159	50.912
48	mp_j90_a2_nr5_AC4	90	2	4	174	326	103.238
49	mp_j90_a2_nr5_AC6	90	2	4	63.5	187	62.933
50	mp_j90_a2_nr5_AC8	90	2	4	37	156	52.326

51	mp_j90_a5_nr5_AC10	90	5	4	96.2	256	70.187
52	mp_j90_a5_nr5_AC5	90	5	4	87	259	70.937
53	mp_j120_a10_nr5_AC10	120	10	4	16.3	188	31.006
54	mp_j120_a10_nr5_AC2	120	10	4	100.2	396	108.309
55	mp_j120_a10_nr5_AC3	120	10	4	156.4	489	131.902
56	mp_j120_a10_nr5_AC4	120	10	4	113.4	436	113.811
57	mp_j120_a10_nr5_AC5	120	10	4	86	409	105.171
58	mp_j120_a10_nr5_AC6	120	10	4	99.5	399	116.464
59	mp_j120_a10_nr5_AC7	120	10	4	13	150	22.040
60	mp_j120_a10_nr5_AC8	120	10	4	22.9	174	34.949
61	mp_j120_a10_nr5_AC9	120	10	4	14.1	157	20.551
62	mp_j120_a20_nr5_AC1	120	20	4	95.55	395	103.045
63	mp_j120_a20_nr5_AC10	120	20	4	93.85	374	102.449
64	mp_j120_a20_nr5_AC2	120	20	4	60.5	305	70.464
65	mp_j120_a20_nr5_AC3	120	20	4	351.75	1005	281.823
66	mp_j120_a20_nr5_AC4	120	20	4	54.85	316	68.448
67	mp_j120_a20_nr5_AC5	120	20	4	78.6	379	88.258
68	mp_j120_a20_nr5_AC8	120	20	4	110.4	385	100.531
69	mp_j120_a20_nr5_AC9	120	20	4	63.6	313	76.177
70	mp_j120_a2_nr5_AC1	120	2	4	69	215	96.1665
71	mp_j120_a2_nr5_AC10	120	2	4	3.5	92	4.950
72	mp_j120_a2_nr5_AC3	120	2	4	54.5	170	54.447
73	mp_j120_a2_nr5_AC5	120	2	4	5	95	7.071
74	mp_j120_a2_nr5_AC6	120	2	4	69	212	97.581
75	mp_j120_a2_nr5_AC7	120	2	4	2	103	2.828
76	mp_j120_a2_nr5_AC8	120	2	4	57.5	180	48.790
77	mp_j120_a5_nr5_AC1	120	5	4	207.6	598	223.267
78	mp_j120_a5_nr5_AC2	120	5	4	69.2	293	75.939
79	mp_j120_a5_nr5_AC4	120	5	4	109.2	387	115.361
80	mp_j120_a5_nr5_AC6	120	5	4	213.6	596	221.218
81	mp_j120_a5_nr5_AC7	120	5	4	73.2	291	85.634

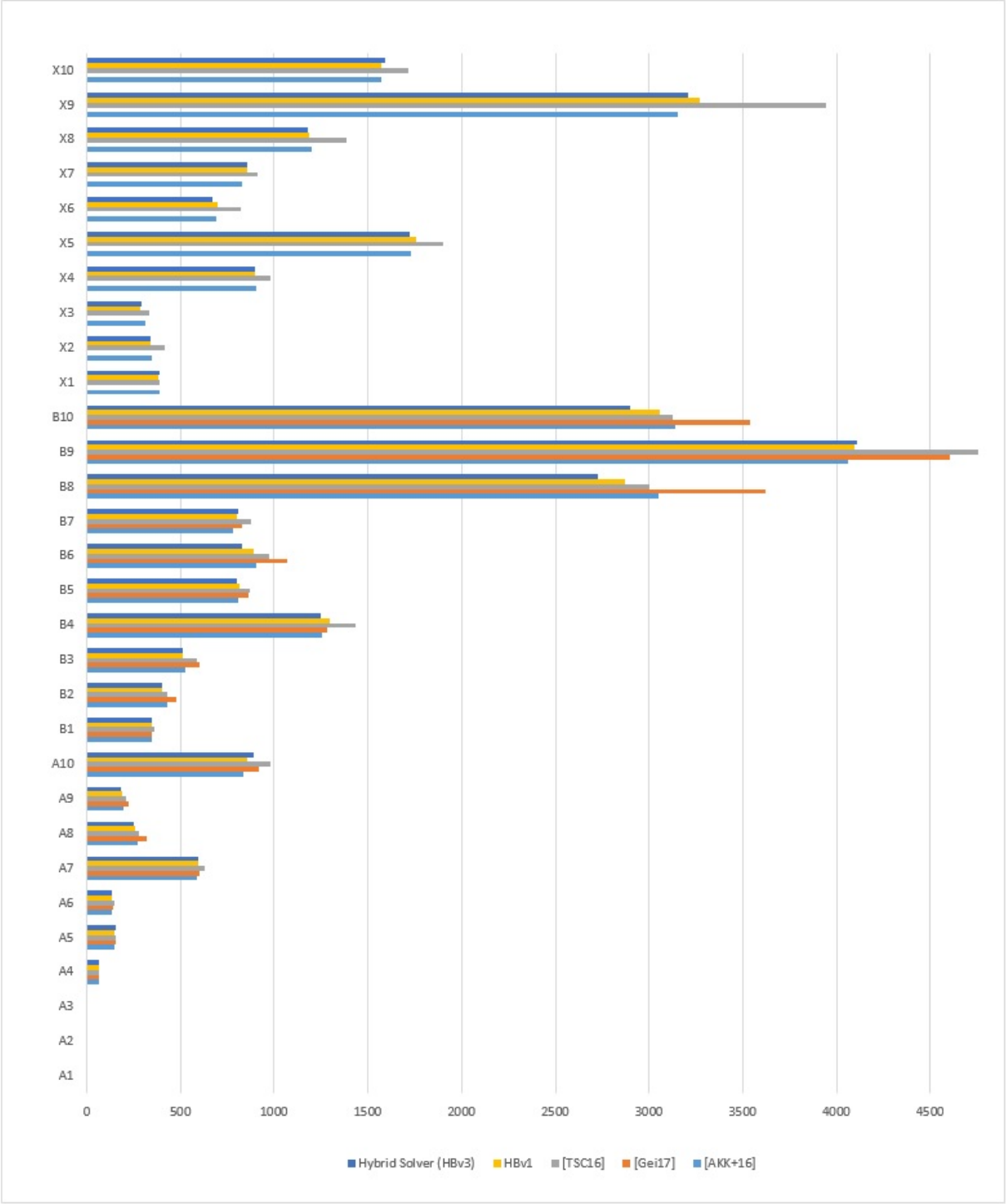

Figure 6.2: Comparison of the new hybrid solver's results (TPD/TMS) with results of the best solvers for MRCMPSP

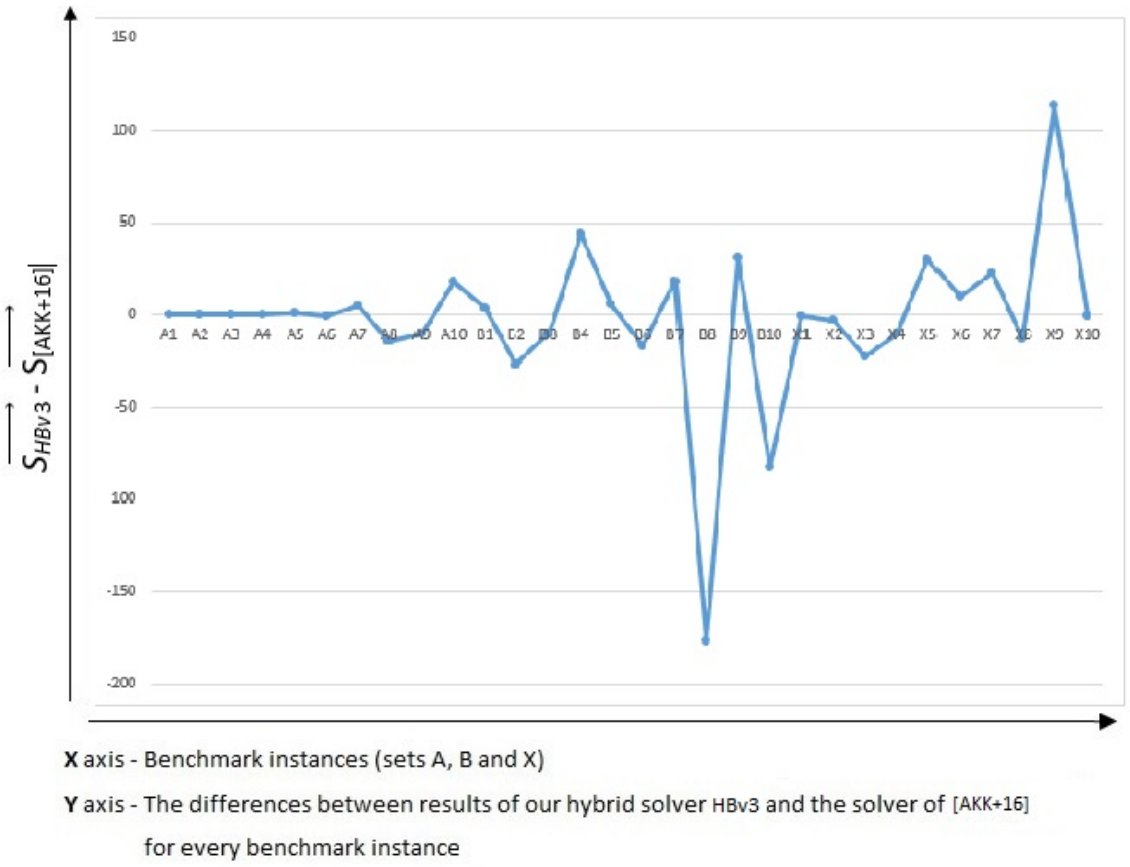

Figure 6.3: Comparison of the results of our HBv3 solver (TPD/TMS) with the results of the best solver ([AKK$^+$16]) for MRCMPSP. The algorithms are run under time relaxed conditions

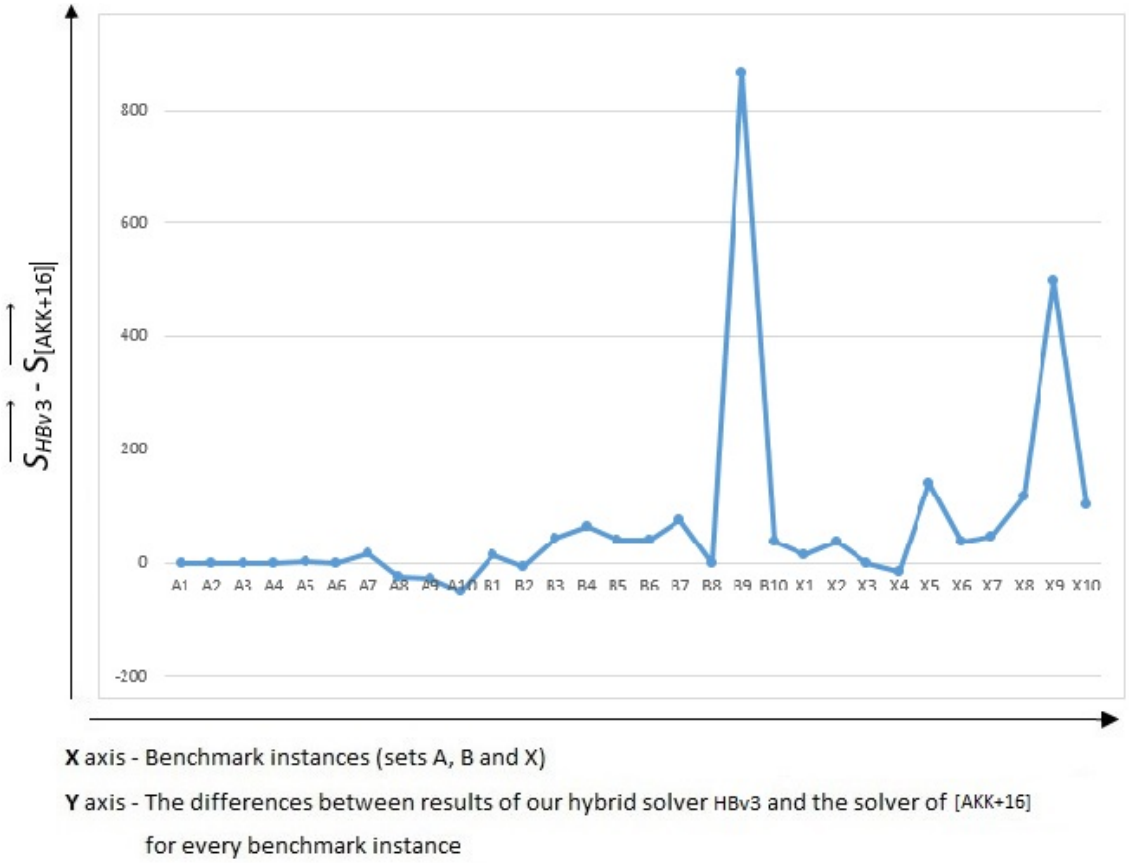

Figure 6.4: Comparison of the results of our HBv3 solver (TPD/TMS) with the results of the [AKK$^+$16] solver for MRCMPSP. The algorithms are run under time relaxed conditions

CHAPTER 7

Min-Conflicts heuristic for Vehicle Routing and Scheduling with Delivery and Installation of Machines

After applying the min-conflicts heuristic in solving project scheduling problems, we continued its experimentation and applied it in solving other categories of problems. Specifically, this chapter presents the min-conflicts heuristic in an ILS approach to solving vehicle routing and scheduling with delivery and installation of machine (DIM). It is one of the latest variants of the vehicle routing problem (VRP) introduced by the EURO Working Group in Vehicle Routing and Logistics Optimization (VeRoLog [1]) and ORTEC in the VeRoLog Solver Challenge 2019 [2] (for more details, see [GvHV19]). This work was done in collaboration with another PhD student (see [KAM21]), and we present only our part of the contribution in this chapter.

7.1 VRP with DIM description

DIM is an extension of the capacitated vehicle routing problem (CVRP) that combines routing and scheduling aspects and optimizes the delivery and consequent installation of equipment, e.g. vending machines. Vending machines must be delivered within the time window requested by customers and should be installed by a technician as soon as possible after delivery. There are different types of vending machines, each with a different size. All customer requests for machines have to be satisfied. The planning

[1] http://www.verolog.eu/
[2] https://verolog2019.ortec.com/

horizon for the deliveries and installations consists of a given number of consecutive days. A machine request contains a certain number of machines of one type. If a customer requests machines of different types, a separate request for each kind is generated. The number of machines is sufficient to serve all requests of the customer. At the beginning of the planning horizon, all the machines are located in a central warehouse. An unlimited number of identical trucks are available to transport the machines from the warehouse to the locations chosen by the customers. One truck can transport machines of different types in one delivery without exceeding the truck capacity, which is expressed in the same unit as the machine size. All machines belonging to one order must be transported at the same time, i.e. the delivery of one order cannot be split. Each truck picking up a delivery from the warehouse must finish its route to the warehouse on the same day. Trucks may return to the warehouse more than once during the day to pick up machines, but may not exceed their maximum daily distance allowed.

After each machine delivery, the installation of the machine at the customer's site must be carried out by a specialized technician as soon as possible. The installation of a machine may not take place earlier than the next day after its delivery. If the machine is not installed on the next day of delivery, a penalty set for the machine will be due, which will increase for each day that the machine is not installed. The installation of a machine is performed by a specialized technician, meaning that each technician has skills defined by the type of machines they can install. Each technician may work a maximum of five consecutive days. If the technician has worked five consecutive days, he has two days off. If the technician has worked less than five consecutive days, he will receive only one day off. Each technician's route must begin and end at their location. The total distance a technician can travel per day is limited to a certain maximum. The number of requests a technician can handle per day is also limited to a maximum. Processing a request means that all machines belonging to that request are installed.

The objective of the DIM problem is to minimize the total cost. The feasible solution consists of the machines supplied and installed throughout the planning horizon according to the customer's requirements. The total cost is composed of several components:

- Unit cost for the distance covered by a truck,
- Cost for using a truck for a day,
- Cost for using a truck at all,
- Cost per unit of the distance traveled by a technician,
- Cost for engaging a technician for a day,
- Cost for using a technician at all, and
- Cost for each day a machine is idle, specific to each type of machine.

To determine the distances traversed, integer coordinates are given for the warehouse, customer locations, and technician home locations. Every location is presented as a pair of coordinates (x, y), and the distance between any two locations, (x_i, y_i) and (x_j, y_j), is calculated as the maximum value of the Euclidean distance: $d = \lceil \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rceil$.

7.2 Description of our approach

Our approach to solving the DIM problem is based on an ILS implementation where the local search consists of several neighborhood operators using the min-conflicts heuristic, destroy-repair, and nearest neighbors heuristics. In this section, the discussion will be focused on the use of min-conflicts in this ILS approach. In the following subsections, we will discuss solution representation, local search, and neighborhood structures in more detail.

7.2.1 Solution representation

A solution in our approach is represented as a list of days, where each day contains a list of truck routes and a list of technicians routes. In addition, the truck routes for a given day contain the delivery requests, including trips to and from the warehouse on that day. The list of technician routes consists of the installation requirements of the machines. The main methods for solving the DIM problem found in the literature are based on the hierarchical decomposition of the problem into subproblems. In such an approach, requests assigned to days and vehicles are grouped into so-called clusters. Each cluster is treated as a TSP problem, e.g. work of [FJ81], [STT10], and [DFFT06] are some typical cases. Our implementation introduces an additional step in the decomposition of the problem by considering delivery and installation scheduling as two related VRP problems.

7.2.2 Local search and neighborhood operators

Our local search heuristic consists of several neighborhood operators that affect machine deliveries or installations (see Algorithm 7.1). It is embedded within an ILS framework. The search is performed by neighborhood operators, which are divided into two groups according to similar characteristics. The first group consists of operators that generate simpler neighborhoods and are faster, and the second group contains operators that generate more complex and larger neighborhoods and take longer to execute and have a more intense effect on the solution. The operators of the first group are executed in each cycle of the local search and get more and more time as long as they improve the solution. The second group is executed only after a parameterized number of failures in improving the solution by the first group.

The first group consists of the following neighborhood operators:

- MoveTrip,

Algorithm 7.1: Local search for VRP-DIM, LS(s)

Input: s a feasible solution;
Output: An improved solution;

```
1  failure = 0;
2  repeat
3      Execute alternatively: MoveTrip(s) or SwapTrips(s);
4      Specify execution order using roulette wheel for:
5      RemoveInsertDeliveriesStack(s);
6      RemoveInsertInstallationsStack(s);
7      RegionGroupingNN(s);
8      LoopSwap(s);
9      SwapInstallations(s);
10     if s < best_s then
11         best_s = s;
12         failure = 0;
13     else
14         failure +=1;
15         if failure > threshold then
16             Intensify search:
17             ReInsertDeliveriesSerial(s);
18             ReInsertInstallationsSerial(s);
19             BestSwapWithNNDays(s);
20             BestSwapWithNNDaysInst(s);
21             if s < best_s then
22                 best_s = s;
23             end
24             Perturbate s:
25             PerturbSwapTrip(s);
26             PerturbReInsertDeliveriesStack(s);
27         end
28     end
29 until timeExpired;
```

- SwapTrips,
- ReInsertDeliverStack
- ReInsertInstallStack,
- RegionGroupingNN,
- LoopSwap,
- SwapInstallations.

The execution order of the operators in this group is determined by applying the roulette wheel rule on the parameterized selection probabilities for each operator.

MoveTrip - applies the idea of min-conflict heuristics to generate the neighborhood by constructing a set of variables consisting of the trips competing for a better allocation in one of the available trucks within the allowed time window. This operator attempts to reduce the number of trucks and/or truck days.

SwapTrips - similarly applies the idea of min-conflict heuristics to generate the neighborhood by constructing a set of variables consisting of pairs of trips competing for better allocations in the available trucks within the allowed time window. This operator attempts to reduce the number of trucks and/or truck days.

ReInsertDeliverStack and ReInsertInstallStack - applies the destroy and repair heuristics to deliveries and installations, respectively. The destroy process removes a delivery from the schedule along with its installation. The repair consists of inserting the delivery and the installation into the schedule.

RegionGroupingNN - groups requests into a new trip assigned to a truck according to nearest neighbors heuristics. This operator can improve the solution by removing the relatively closest requests that are distributed among different trips and putting them into a common trip.

LoopSwap - is applied to deliveries of a given day. It performs sequential swaps of deliveries. This operator is based on the idea of the eject chain proposed by [RR96].

SwapInstallations - operates on installations. It randomly selects an installation and then swaps it in order with all other feasible installations. The operator is executed as long as there are improvements.

The second group of the neighborhood operators consists of the following:

- ReInsertDeliverSerial,
- ReInsertInstallationsSerial,
- BestSwapWithNNDays,
- BestSwapWithNNDaysInst.

ReInsertDeliverSerial and ReInsertInstallationsSerial - perform a series of shifts of the lists of deliveries and installations respectively. The evaluation of a solution is done only after the displacement is done for a complete list of deliveries and installations. The idea is to escape local optima by accepting a potentially worse solution before allocating the full list of deliveries and installations.

BestSwapWithNNDays and BestSwapWithNNDaysInst - Perform swaps between each of the two deliveries and their installations from a complete set of requests of a certain day. These operators are executed as long as they improve the solution.

7.3 Computational results

We participated in the VeRoLog Solver Challenge 2019 and tested our approach with several sets of examples presented in this competition. The competition was organized in two stages: *all-time-best challenge* and *restricted resources challenge*. In the first stage, there were no time or technology restrictions on solving the available instances. In the second stage, there were imposed restrictions expressed through the formula:

$$t = f * (10 + R) \tag{7.1}$$

where f is the factor related to the machine the algorithm is run; it is calculated by running a benchmark program provided on the challenge website. R is the number of requests, and t is the execution time limit, expressed in seconds, for a given instance.
Thirteen teams participated in the competition and submitted their solutions in the first stage. Only eight of them managed to qualify for the final round; our solver was one of them.

7.3.1 Benchmark instances

At the VeRoLog Solver Challenge 2019, three sets of instances were introduced, each containing 25 instances with different characteristics. The first set of instances, called the "early set", was introduced for the *all-time-best challenge* stage. The "early set" includes instances from the smallest, consisting of 150 requests to the largest, with up to 900 requests. Each team's results for the "early set" of instances were published and ranked online during this stage.

The second set of instances, called the "late set", was presented for the *restricted resources challenge* stage. All teams had to submit the solution for each instance from this set and the solver. The challenge organizers ran the solvers nine times per instance on their machine. For each instance, the best and worst solutions were excluded from the analysis, and the average results and the standard deviation were calculated for the remaining five. Using these results, the qualified solvers were tested on the last set, called the "hidden set". This set was made available for the *restricted resources challenge* stage and was used after the finalists of the competition were determined and the solver submission was completed.

7.3.2 Performance of our solver

Our solver was qualified for the final phase. In the "early set" of instances, our solver achieved second place for the first instance, third place for six instances, fourth place for thirteen instances, fifth place for four instances and sixth place for one instance. For seven instances, the relative difference between our solver's results and the best solver is less than 1%. In the case of the "late set", as mentioned earlier, all teams submitted their solutions for each instance from that set and their solver. The organizers of the competition ran the solvers on their machines to determine the final rank of all solvers for each instance. The final rank for each solver is calculated as the average of all its ranks. Our solver produced better results than the fourth-ranked solver for seven out of twenty-five instances.

7.3.3 Discussion and analysis

We introduced an implementation of the min-conflicts, destroy-repair, and nearest neighbors heuristics embedded in an ILS framework for Vehicle Routing and Scheduling with Deliveries and Installation of Machines problem. We participated in the VeRoLog Solver Challenge 2019, where three sets of instances were provided. The competition was organized in two stages: *all-time-best challenge* and *restricted resources challenge*. Our approach was one of the eight qualified solvers for the final stage. It was tested with different sets of instances in different stages and gave promising results for many of them. It outperformed the solver ranked fourth in four out of twenty-five instances in the final evaluation.

CHAPTER 8

Conclusion

In this thesis, we studied the MRCMPSP as one of the most general variants of the project scheduling problem with significant relevance to research and industry. We introduced several new solution approaches for MRCMPSP, including exact, meta-heuristic, and hybrid techniques. We have also evaluated our solution methods on several well-known project scheduling problems such as MRCPSP and RCMPSP. Our solution approaches improve upon state-of-the-art results and solutions for MRCMPSP, MRCPSP and RCMPSP, an achieve new upper bounds for many of the benchmark instances.

In Chapter 3, we explained the implementation and evaluation of a new meta-heuristic solution method for MRCMPSP based on the min-conflicts heuristic. The main idea of this solution approach is to efficiently explore the neighborhood of the current solution based on activities competing for the same resources. This technique is further used within an ILS framework in combination with tabu search lists. In our experiments, we found that the neighborhood operators that manipulate the execution modes of activities have a significant impact on the quality of the active schedule. Therefore, we focused mainly on neighborhood operators that manipulate modes of activities and proposed three new project-wise neighborhood operators for the MRCMPSP. Our experiments showed that the new operators are useful for this problem. We improved our meta-heuristic solver with two successive enhancements and each variant was evaluated on MRCMPSP instances introduced in the MISTA 2013 Challenge and on well-known MRCPSP MMLIB benchmark instances (MMLIB50, MMLIB100, and MMLIB+ sets). Compared to the best performing approaches for MRCMPSP, our method achieved competitive results to the third-ranked solver and provided eleven new upper bounds for MRCPSP MMLIB instances.

Chapter 4 analyses the development and testing of a CP model for MRCMPSP. CP Models have successfully solved various scheduling problems from research and industry. Our CP model is based on an existing model, which has been applied to the related problem MRCPSP. To the best of our knowledge, this is the first CP model developed for

MRCMPSP. Another important aspect of developing the CP model for MRCMPSP the configuration and fine-tuning of its parameters. Compared to state-of-the-art algorithms for MRCMPSP, our CP model achieved competitive results for many benchmark instances. From our experiments, we can conclude that the CP model performs well for MRCMPSP instances, especially for small and medium sized instances.

In Chapter 5, we presented a hybrid model consisting of the combination of the meta-heuristic with the CP model presented in the previous chapters. The starting point for this hybrid model architecture was based on several models analyzed in the literature. Our hybrid model could be classified as a high-level relay hybrid approach. This hybrid approach achieved new upper bounds and outperformed all state-of-the-art solvers for most instances of different categories of project scheduling problems under relaxed time limits. Our hybrid approach also dominates its constituent components, the meta-heuristic and the CP model.

We investigated and experimented with two hybrid approaches for MRCMPSP. The experiments showed us that improving the complementary search of the algorithms forming a potential hybrid method would be of great interest. In this context, we found that the hybrid model containing a meta-heuristic of a more stochastic nature performed better than the other, although its constituent components performed worse than the components of its counterpart.

Therefore, in Chapter 6, we considered adding a new component to our meta-heuristic. We considered implementing a simulated annealing based approach as a new meta-heuristic at this stage. On the one hand, this approach helps to avoid getting stuck in a local optimum, and on the other hand, it increases the stochastic character of the algorithm. In this chapter, we presented the implementation of the simulated annealing heuristic for the MRCMPSP in three variants, depending on the types of neighborhoods they generate: a variant based on changing the activity mode, a variant that shifts the position of a randomly selected activity in the schedule to its first predecessor, and a variant that shifts the position of a randomly selected activity in the schedule to its first successor. The simulated annealing heuristic was successfully implemented and integrated into the meta-heuristic component as an independent neighborhood operator. The new hybrid method composed of the extended meta-heuristic and the CP model is developed and evaluated on several types of project scheduling benchmark instances with different objectives: MISTA 2013 Challenge MRCMPSP, MMLIB MRCPSP, and MPSPLIB RCMPSP instances. This hybrid approach achieved new upper bounds and outperformed all state-of-the-art solvers for most instances of these problems under relaxed time limits. It also performed better than the hybrid approaches presented in Chapter 5.

In Chapter 7, we introduced an implementation of the min-conflicts heuristic combined with destroy-repair and nearest neighbors heuristics embedded in an ILS framework for the Vehicle Routing and Scheduling with Deliveries and Installation of Machines problem, even though this problem was not the main subject of this study. We participated in the VeRoLog Solver Challenge 2019. Our approach was one of the eight qualified solvers for

the final stage. It was tested with different instance sets in different stages and delivered promising results for many of them. It outperformed the solver ranked fourth for most instances. It can be seen that the min-conflicts heuristic can provide promising results for scheduling problems when combined with other local search methods.

8.1 Future work

Our main objective was to implement and evaluate different solution techniques for solving MRCMPSP. Our work culminated with a hybrid approach combining two complementary search strategies. This study introduces a successful combination of an exact model (CP model) with several meta-heuristic approaches and a particular perturbation mechanism. Considering that the hybrid models performed better than the algorithms that composed them, we believe it is very interesting to investigate different hybrid approaches that combine different search strategies to create a robust and efficient method for future work.

In particular, the role of meta-heuristics in a hybrid method should be further investigated. In our case, our hybrid model performed very well on MRCMPSP instances, as well as on MRCMPSP and RCMPSP instances, but there might still be room for improving the convergence time of the model for very large instances under very short time limits. Since the exact model relies on searching large neighborhoods and performs very well on small and medium instances, it is up to the meta-heuristic component to perform an efficient search under very short time limits to achieve excellent results even on very large instances.

Another valuable research topic could be the investigation of the architecture of the hybrid model itself. In our implementation, the meta-heuristic and the CP model are integrated in a hybrid model as black boxes and independently improve an active schedule. Further interaction between the two could be explored, e.g., using the CP model to target specific neighborhoods within the meta-heuristic or to repair infeasible and/or partial solutions.

Another interesting research topic could be the combination of some of the best hyper-heuristic and meta-heuristic solution approaches to date with the CP model for project scheduling problems. The investigation of machine learning methods for various purposes, i.e., selection of heuristics and determination of the quality of candidate solutions, could be of great interest in this context.

List of Figures

List of Tables

List of Algorithms

Acronyms

AOA Activity-on-Arc. 8

AON Activity-on-Node. 8

CP Constraint Programming. 3

CVRP capacitated vehicle routing problem. 89

DIM delivery and installation of machine. 89

ILS iterated local search. xiv, 27, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52

MIRCPSP The mode-identity and resource-constrained project scheduling problem. 13

MMLIB multi-mode instances library. 21

MPSPLIB multi-project scheduling problem library. 21

MRCMPSP The multi-mode resource-constrained multi-project scheduling problem. 13

MRCPSP The multi-mode resource-constrained project scheduling problem. 11

MRCPSP-GPR The multi-mode resource-constrained project scheduling problem with generalized precedence relations. 12

MRCPSPDCF The multi-mode resource-constrained project scheduling problem with discounted cash flows. 12

PSPLIB project scheduling problem library. 20

RCMPSP The resource-constrained multi-project scheduling problem. 12

RCPSP The resource constrained project scheduling problem. 9

RCPSPMCM The resource-constrained project scheduling problem with multiple crashable modes. 13

SRCPSP The stochastic resource-constrained project scheduling. 13

TLSP The test laboratory scheduling problem. 13

VRP vehicle routing problem. 89

Bibliography

[AE98] Taeho Ahn and S. Selcuk Erenguc. The resource constrained project scheduling problem with multiple crashable modes: A heuristic procedure. *European Journal of Operational Research*, 107(2):250–259, 1998.

[AH13] Christian Artigues and Emmanuel Hebrard. Mip relaxation and large neighborhood search for a multi-mode resource-constrained multi-project scheduling problem. In *Proceedings of the 6th Multidisciplinary International Scheduling Conference*, pages 815–819, 2013.

[AKK+16] Shahriar Asta, Daniel Karapetyan, Ahmed Kheiri, Ender Özcan, and Andrew J. Parkes. Combining monte-carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem. *Information Sciences*, 373:476–498, 2016.

[AM18] Arben Ahmeti and Nysret Musliu. Min-conflicts heuristic for multi-mode resource-constrained projects scheduling. In Hernán E. Aguirre and Keiki Takadama, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, pages 237–244. ACM, 2018.

[AM21] Arben Ahmeti and Nysret Musliu. Hybridizing constraint programming and meta-heuristics for multi-mode resource-constrained multiple projects scheduling problem. *Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT*, 1:188–206, 2021.

[AMR03] J Alcaraz, C Maroto, and R Ruiz. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, 54(6):614–626, 2003.

[APPR13] Federico Alonso-Pecina, Johnatan E. Pecero, and David Romero. A three-phases based algorithm for the multi-mode resource-constrained multi-project scheduling problem. In *Proceedings of the 6th Multidisciplinary International Scheduling Conference*, pages 812–814, 2013.

[BBK99] Tonius Baar, Peter Brucker, and Sigrid Knust. Tabu search algorithms and lower bounds for the resource-constrained project scheduling problem. In *Meta-Heuristics*, pages 1–18. Springer, 1999.

[BBU15] Umut Beşikci, Ümit Bilge, and Gündüz Ulusoy. Multi-mode resource constrained multi-project scheduling and resource portfolio problem. *European Journal of Operational Research*, 240(1):22–31, 2015.

[BBZ+13] Leonardo M. Borba, Alexander J. Benavides, Tadeu Zubaran, Germano C. Carniel, and Marcus Ritt. A simple stochastic local search for multi-mode resource-constrained multi-project scheduling. In *Proceedings of the 6th Multidisciplinary International Scheduling Conference*, pages 826–830, 2013.

[BDKS99] Jan Böttcher, Andreas Drexl, Rainer Kolisch, and Frank Salewski. Project scheduling under partially renewable resource constraints. *Management Science*, 45(4):543–559, 1999.

[BDM+99] Peter Brucker, Andreas Drexl, Rolf Möhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1999.

[BDMX13] Hermann Bouly, Duc-Cuong Dang, Aziz Moukrim, and Huang Xu. Evolutionary algorithm and post-processing to minimize the total delay in multi-project scheduling. In *Proceedings of the 6th Multidisciplinary International Scheduling Conference*, pages 831–835, 2013.

[BDP81] Roger B. Bey, Robert H. Doersch, and James H. Patterson. The net present value criterion: its impact on project scheduling. *Project Management Quarterly*, 12(2):35–45, 1981.

[biT20] Samer ben issa and Yiliu Tu. A survey in the resource-constrained project and multi-project scheduling problems. *Journal of Project Management*, pages 117–138, 01 2020.

[BKST98] Peter Brucker, Sigrid Knust, Arno Schoo, and Olaf Thiele. A branch and bound algorithm for the resource-constrained project scheduling problem1supported by the deutsche forschungsgemeinschaft, project 'komplexe maschinen-schedulingprobleme'.1. *European Journal of Operational Research*, 107(2):272–288, 1998.

[BL03] K. Bouleimen and H. Lecocq. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, 149(2):268–281, 2003. Sequencing and Scheduling.

[Bła86] Jacek Błażewicz. *Scheduling under resource constraints: Deterministic models*. JC Baltzer, 1986.

[BLK83] Jacek Blazewicz, Jan Karel Lenstra, and Alexander H.G. Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.

[BMR88] M. Bartusch, R. H. Mohring, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Ann. Oper. Res.*, 16(1–4):201–240, January 1988.

[BN96] Klaus Brinkmann and Klaus Neumann. Heuristic procedures for resource—constrained project scheduling with minimal and maximal time lags: the resource—levelling and minimum project—duration problems. *Journal of Decision Systems*, 5(1-2):129–155, 1996.

[BOC93] F. F. BOCTOR. Heuristics for scheduling projects with resource restrictions and several resource-duration modes. *International Journal of Production Research*, 31(11):2547–2558, 1993.

[BOC96a] F. F. BOCTOR. Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, 34(8):2335–2351, 1996.

[Boc96b] Fayez F. Boctor. A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. *European Journal of Operational Research*, 90(2):349–361, 1996.

[BSZ06] Francisco Ballestín, C. Schwindt, and J. Zimmermann. Resource leveling in make-to-order production : Modeling and heuristic solution method. 2006.

[BTR94] M. Bandelloni, M. Tucci, and R. Rinaldi. Optimal resource leveling using non-serial dyanamic programming. *European Journal of Operational Research*, 78(2):162–177, 1994.

[CGR07] Giuseppe Confessore, Stefano Giordani, and Silvia Rismondo. A market-based multi-agent system model for decentralized multi-project scheduling. *Annals OR*, 150:115–135, 03 2007.

[CHW08] Chuan-Wen Chiang, Yu-Qing Huang, and Wen-Yen Wang. Ant colony optimization with parameter adaptation for multi-mode resource-constrained project scheduling. *J. Intell. Fuzzy Syst.*, 19(4,5):345–358, December 2008.

[CV11] José Coelho and Mario Vanhoucke. Multi-mode resource-constrained project scheduling using rcpsp and sat solvers. *European Journal of Operational Research*, 213(1):73–82, 2011.

[CV20] José Coelho and Mario Vanhoucke. Going to the core of hard resource-constrained project scheduling instances. *Computers & Operations Research*, 121:104976, 2020.

[CZC+10] Wei Neng Chen, Jun Zhang, Henry Shu Hung Chung, Rui Zhang Huang, and Ou Liu. Optimizing discounted cash flows in project scheduling-an ant colony optimization approach. *IEEE Transactions on Systems, Man and Cybernetics: Part C*, 40(1):64–77, January 2010.

[DFFT06] Roberto De Franceschi, Matteo Fischetti, and Paolo Toth. A new ilp-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105(2-3):471–499, 2006.

[DG93] ANDREAS DREXL and JUERGEN GRUENEWALD. Nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 25(5):74–81, 1993.

[DGMM20] Philipp Danzinger, Tobias Geibinger, Florian Mischek, and Nysret Musliu. Solving the test laboratory scheduling problem with variable task grouping. In J. Christopher Beck, Olivier Buffet, Jörg Hoffmann, Erez Karpas, and Shirin Sohrabi, editors, *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, pages 357–365. AAAI Press, 2020.

[DH97] Erik L. Demeulemeester and Willy S. Herroelen. New benchmark results for the resource-constrained project scheduling problem. *Management Science*, 43(11):1485–1492, 1997.

[DH99] Bert De Reyck and Willy Herroelen. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 119(2):538–556, 1999.

[DJSL09] N. Damak, B. Jarboui, P. Siarry, and T. Loukil. Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Computers & Operations Research*, 36(9):2653–2659, 2009.

[DKG13] Olfa Dridi, Saoussen Krichen, and Adel Guitouni. Solving resource-constrained project scheduling problem by a genetic local search approach. In *2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, pages 1–5, 2013.

[DMaBL15] Aidin Delgoshaei, Mohd khairol anuar Mohd ariffin, B T Baharudin, and Z. Leman. Minimizing makespan of a resource-constrained scheduling problem: A hybrid greedy and genetic algorithms. *International Journal of Industrial Engineering Computations*, 6:503–520, 09 2015.

[DV07] Dieter Debels and Mario Vanhoucke. A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. *Operations Research*, 55(3):457–469, 2007.

[DVH03] E. Demeulemeester, M. Vanhoucke, and W. Herroelen. Rangen: A random network generator for activity-on-the-node networks. *Journal of Scheduling*, 6:17–38, 2003.

[EAC01] S. S. Erenguc, T. Ahn, and D. Conway. The resource constrained project scheduling problem with multiple crashable modes: An exact solution method. *Naval Research Logistics*, 48:107–127, 2001.

[EAM+19] Tirkolaee E.B, Goli A, Hematian M, Arun Kumar S, and Han T. Multi-objective multi-mode resource constrained project scheduling problem using pareto-based algorithms. *Computing*, 101(6):547 – 570, 2019.

[EF10] Sonda Elloumi and Philippe Fortemps. A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 205(1):31–41, 2010.

[Elm77] Salah Eldin Elmaghraby. *Activity networks : project planning and control by network models / Salah E. Elmaghraby.* Wiley New York, 1977.

[Fen67] Larry Glenn Fendley. The development of a complete multi-project scheduling system using a forecasting and sequencing technique. 1967.

[FJ81] Marshall L Fisher and Ramchandran Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.

[Gei17] Martin Josef Geiger. A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem. *European Journal of Operational Research*, 256(3):729–741, 2017.

[GLLK79] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287–326. Elsevier, 1979.

[GMM19] Tobias Geibinger, Florian Mischek, and Nysret Musliu. Investigating constraint programming for real world industrial test laboratory scheduling. In Louis-Martin Rousseau and Kostas Stergiou, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 16th International Conference, CPAIOR 2019, Thessaloniki, Greece, June 4-7, 2019, Proceedings*, volume 11494 of *Lecture Notes in Computer Science*, pages 304–319. Springer, 2019.

[GMM21] Tobias Geibinger, Florian Mischek, and Nysret Musliu. Constraint logic programming for real-world test laboratory scheduling. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6358–6366. AAAI Press, 2021.

[Gra66] R. L. Graham. Bounds for certain multiprocessing anomalies. *The Bell System Technical Journal*, 45(9):1563–1581, 1966.

[GSF17] Patrick Gerhards, Christian Stürck, and Andreas Fink. An adaptive large neighbourhood search as a matheuristic for the multi-mode resource-constrained project scheduling problem. *European Journal of Industrial Engineering*, 11(6):774–791, 2017.

[GvHV19] Joaquim Gromicho, Pim van't Hof, and Daniele Vigo. The verolog solver challenge 2019. *Journal on Vehicle Routing Algorithms*, 2(1-4):109–111, 2019.

[Har98] Sönke Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7):733–750, 1998.

[Har01] Sönke Hartmann. Project scheduling with multiple modes: A genetic algorithm. *Annals of Operations Research*, 102(1):111–135, 2001.

[HB10] Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010.

[HBS18] Farhad Habibi, Farnaz Barzinpour, and Seyed Sadjadi. Resource-constrained project scheduling problem: review of past and recent developments. *Journal of Project Management*, 3:55–88, 01 2018.

[HD98] Sönke Hartmann and Andreas Drexl. Project scheduling with multiple modes: A comparison of exact algorithms. *Networks*, 32(4):283–297, 1998.

[HDDR99] Willy Herroelen, Erik Demeulemeester, and Bert De Reyck. A classification scheme for project scheduling. In *Project scheduling*, pages 1–26. Springer, 1999.

[HHL11] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization - 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers*, pages 507–523, 2011.

[HHW21] Yukang He, Zhengwen He, and Nengmin Wang. Tabu search and simulated annealing for resource-constrained multi-project scheduling to minimize

maximal cash flow gap. *Journal of Industrial & Management Optimization*, 17(5):2451–2474, 2021.

[Hom07] Jörg Homberger. A multi-agent system for the decentralized resource-constrained multi-project scheduling problem. *International Transactions in Operational Research*, 14(6):565–589, 2007.

[Hom12] Jörg Homberger. A (μ, λ)-coordination mechanism for agent-based multi-project scheduling. *OR Spectrum*, 34:107–132, 01 2012.

[IE94] Oya Icmeli and S.Selcuk Erenguc. A tabu search procedure for the resource constrained project scheduling problem with discounted cash flows. *Computers & Operations Research*, 21(8):841–853, 1994. Heuristic, Genetic and Tabu Search.

[JDSR08] B. Jarboui, N. Damak, P. Siarry, and A. Rebai. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, 195(1):299–308, 2008.

[JMR+01] Joanna Józefowska, Marek Mika, Rafał Różycki, Grzegorz Waligóra, and Jan Węglarz. Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, 102(1-4):137–155, 2001.

[KAM21] Valon Kastrati, Arben Ahmeti, and Nysret Musliu. Solving vehicle routing and scheduling with delivery and installation of machines using ils. *Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT*, 1:207–223, 2021.

[KD82] I. Kurtulus and E. W. Davis. Multi-project scheduling: Categorization of heuristic rules performance. *Management Science*, 28(2):161–172, 1982.

[KD97a] R. Kolisch and A. Drexl. Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 29:987–999, 1997.

[KD97b] Rainer Kolisch and Andreas Drexl. Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE transactions*, 29(11):987–999, 1997.

[KDH00] Gary Knotts, M. Dror, and B. Hartman. Agent-based project scheduling. *IIE Transactions*, 32:387 – 401, 2000.

[KM20] Vladislav Korotkov and Mikhail Matveev. Individual scheduling for the multi-mode resource-constrained multi-project scheduling problem. In *CEUR Workshop Proceedings*, 2020.

[KN85] Ibrahim S. Kurtulus and S. Narula. Multi-project scheduling: Analysis of project performance. *Iie Transactions*, 17:58–66, 1985.

[Kol96] Rainer Kolisch. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2):320–333, 1996.

[KP97] Rainer Kolisch and Rema Padman. An integrated survey of project scheduling. Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 463, Kiel; Kiel, Hamburg, 1997.

[KREK19] Marimuthu Kannimuthu, Benny Raphael, Palaneeswaran Ekambaram, and Ananthanarayanan Kuppuswamy. Comparing optimization modeling approaches for the multi-mode resource-constrained multi-project scheduling problem. *Engineering, Construction and Architectural Management*, 2019.

[KS97] Rainer Kolisch and Arno Sprecher. Psplib - a project scheduling problem library: Or software - orsep operations research software exchange program. *European Journal of Operational Research*, 96(1):205–216, 1997.

[KSD95] Rainer Kolisch, Arno Sprecher, and Andreas Drexl. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, 41(10):1693–1703, 1995.

[KW59] James E. Kelley and Morgan R. Walker. Critical-path planning and scheduling. In *Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '59 (Eastern), page 160–173, New York, NY, USA, 1959. Association for Computing Machinery.

[KZV+16] Mathias Kühn, Taiba Zahid, Michael Völker, Zhugen Zhou, and Oliver Rose. Investigation of genetic operators and priority heuristics for simulation based optimization of multi-mode resource constrained multi-project scheduling problems (mmrcmpsp). In *ECMS*, 2016.

[LEF+17] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, Stefan Falkner, Andre Biedenkapp, and Frank Hutter. Smac v3: Algorithm configuration in python. `https://github.com/automl/SMAC3`, 2017.

[LIMMS13] Manuel López-Ibánez, Franco Mascia, Marie-Eléonore Marmion, and Thomas Stützle. Automatic design of a hybrid iterated local search for the multi-mode resource-constrained multi-project scheduling problem. In *Proceedings of the 6th Multidisciplinary International Scheduling Conference*, pages 820–825, 2013.

[LMS03] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. Iterated local search. *International series in operations research and management science*, pages 321–354, 2003.

[LMT00] Antonio Lova, Concepción Maroto, and Pilar Tormos. A multicriteria heuristic method to improve resource allocation in multiproject scheduling. *European Journal of Operational Research*, 127(2):408–424, 2000.

[LRSV18] Philippe Laborie, Jérôme Rogerie, Paul Shaw, and Petr Vilím. Ibm ilog cp optimizer for scheduling. *Constraints*, 23(2):210–250, April 2018.

[LTCB09] Antonio Lova, Pilar Tormos, Mariamar Cervantes, and Federico Barber. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, 117(2):302–316, 2009.

[LV16] Pieter Leyman and Mario Vanhoucke. Payment models and net present value optimization for resource-constrained project scheduling. *Computers & Industrial Engineering*, 91:139–153, 2016.

[LW15] Haitao Li and Keith Womer. Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *European Journal of Operational Research*, 246, 04 2015.

[MJPL92] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artif. Intell.*, 58(1-3):161–205, 1992.

[MM18] Florian Mischek and Nysret Musliu. The test laboratory scheduling problem. Technical report, Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling, TU Wien, 2018.

[MM21] Florian Mischek and Nysret Musliu. A local search framework for industrial test laboratory scheduling. *Annals of Operations Research*, 302:533–562, 07 2021.

[MMRB98] Aristide Mingozzi, Vittorio Maniezzo, Salvatore Ricciardelli, and Lucio Bianco. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science*, 44(5):714–729, 1998.

[MMS21] Florian Mischek, Nysret Musliu, and Andrea Schaerf. Local search approaches for the test laboratory scheduling problem with variable task grouping. *Journal of Scheduling*, pages 1–21, 2021.

[MRCF59] D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar. Application of a technique for research and development program evaluation. *Operations Research*, 7(5):646–669, 1959.

[MT97] Masao Mori and Ching Chih Tseng. A genetic algorithm for multi-mode resource constrained project scheduling problem. *European Journal of Operational Research*, 100(1):134–141, 1997.

[NI02] Koji Nonobe and Toshihide Ibaraki. *Formulation and Tabu Search Algorithm for the Resource Constrained Project Scheduling Problem*, pages 557–588. Springer US, Boston, MA, 2002.

[NZ00] K. Neumann and J. Zimmermann. Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. *European Journal of Operational Research*, 127(2):425–443, 2000.

[Ozd99] Linet Ozdamar. A genetic algorithm approach to a general category project scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 29:44–59, 01 1999.

[PSDSD97] Rema Padman, Dwight E. Smith-Daniels, and Vicki L. Smith-Daniels. Heuristic scheduling of resource-constrained projects with cash flows. *Naval Research Logistics (NRL)*, 44(4):365–381, 1997.

[PV10] Vincent Van Peteghem and Mario Vanhoucke. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201(2):409–418, 2010.

[PV11] Vincent Peteghem and Mario Vanhoucke. Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem. *J. Heuristics*, 17:705–728, 12 2011.

[PWW69] A Alan B Pritsker, Lawrence J Waiters, and Philip M Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management science*, 16(1):93–108, 1969.

[RCL18] Salim Rostami, Stefan Creemers, and Roel Leus. New strategies for stochastic resource-constrained project scheduling. *Journal of Scheduling*, 21(3):349–365, 2018.

[RDK09] Mohammad Ranjbar, Bert De Reyck, and Fereydoon Kianfar. A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *European Journal of Operational Research*, 193(1):35–48, 2009.

[RHA13] Mohammad Ranjbar, Saeed Hosseinabadi, and Foroogh Abasian. Minimizing total weighted late work in the resource-constrained project scheduling problem. *Applied Mathematical Modelling*, 37(23):9776–9785, 2013.

[RPSN21] Azin Rashidi-Pour and Majid Shakhsi-Niaei. A twofold constructive genetic algorithm for resource-constrained multi-project scheduling problem (rcmpsp). *Journal of Quality Engineering and Production Optimization*, 6(1):1–14, 2021.

[RR96] César Rego and Catherine Roucairol. A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In *Meta-Heuristics*, pages 661–675. Springer, 1996.

[Rus70] A. H. Russell. Cash flows in networks. *Management Science*, 16(5):357–373, 1970.

[SD98] Arno Sprecher and Andreas Drexl. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm1supported by the deutsche forschungsgemeinschaft.1. *European Journal of Operational Research*, 107(2):431–450, 1998.

[SH16] Alexander Schnell and Richard F. Hartl. On the efficient modeling and solution of the multi-mode resource-constrained project scheduling problem with generalized precedence relations. *OR Spectr.*, 38(2):283–303, mar 2016.

[SH17] Alexander Schnell and Richard F. Hartl. On the generalization of constraint programming and boolean satisfiability solving techniques to schedule a resource-constrained project consisting of multi-mode jobs. *Operations Research Perspectives*, 4:1–11, 2017.

[Spr96] Arno Sprecher. Solving the rcpsp efficiently at modest memory requirements. Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 425, Kiel; Kiel, Hamburg, 1996.

[SSD97] Frank Salewski, Andreas Schirmer, and Andreas Drexl. Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application. *European Journal of Operational Research*, 102(1):88–110, 1997.

[SSH08] Majid Sabzehparvar and S. Mohammad Seyed-Hosseini. A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent time lags. *J. Supercomput.*, 44(3):257–273, June 2008.

[SSW94] Roman Słowiński, Bolesław Soniewicki, and Jan Węglarz. Dss for multiobjective project scheduling. *European Journal of Operational Research*, 79(2):220–229, 1994.

[STT10] Majid Salari, Paolo Toth, and Andrea Tramontani. An ilp improvement procedure for the open vehicle routing problem. *Computers & Operations Research*, 37(12):2106–2120, 2010.

[SV93] M.Grazia Speranza and Carlo Vercellis. Hierarchical models for multiproject planning and scheduling. *European Journal of Operational Research*, 64(2):312–325, 1993. Project Management anf Scheduling.

[Sł81] Roman Słowinski. Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. *European Journal of Operational Research*, 7(3):265–273, 1981.

[Tal09] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[TC09] Lin-Yu Tseng and Shih-Chieh Chen. Two-phase genetic local search algorithm for the multimode resource-constrained project scheduling problem. *IEEE Transactions on Evolutionary Computation*, 13(4):848–857, 2009.

[TSCS16] Túlio A.M. Toffolo, Haroldo G. Santos, Marco A.M. Carvalho, and Janniele A. Soares. An integer programming approach to the multimode resource-constrained multiproject scheduling problem. *Journal of Scheduling*, 19(3):295–307, 2016.

[Tse08] Ching-Chih Tseng. Two heuristic algorithms for a multi-mode resource-constrained multi-project scheduling problem. *Journal of Science and Engineering Technology*, 4(2):63–74, 2008.

[USŞŞ01] Gündüz Ulusoy, Funda Sivrikaya-Şerifoğlu, and Şule Şahin. Four payment models for the multi-mode resource constrained project scheduling problem with discounted cash flows. *Annals of Operations research*, 102(1):237–261, 2001.

[VC18] Mario Vanhoucke and José Coelho. A tool to test and validate algorithms for the resource-constrained project scheduling problem. *Computers & Industrial Engineering*, 118:251–265, 2018.

[VCD+08] Mario Vanhoucke, José Coelho, Dieter Debels, Broos Maenhout, and Luís V. Tavares. An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, 187(2):511–524, 2008.

[VPV10] Vincent Van Peteghem and Mario Vanhoucke. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201(2):409–418, 2010.

[VPV14] Vincent Van Peteghem and Mario Vanhoucke. An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research*, 235(1):62–72, 2014.

[WF12] Ling Wang and Chen Fang. An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem. *Computers & Operations Research*, 39(2):449–460, 2012.

[WJMW11] Jan Węglarz, Joanna Józefowska, Marek Mika, and Grzegorz Waligóra. Project scheduling with finite or infinite number of activity processing modes – a survey. *European Journal of Operational Research*, 208(3):177–205, 2011.

[WKS+16] Tony Wauters, Joris Kinable, Pieter Smet, Wim Vancroonenburg, Greet Vanden Berghe, and Jannes Verstichel. The multi-mode resource-constrained

multi-project scheduling problem. *Journal of Scheduling*, 19(3):271–283, 2016.

[WVBC11] T Wauters, K Verbeeck, G Vanden Berghe, and P De Causmaecker. Learning agents for the multi-mode project scheduling problem. *Journal of the Operational Research Society*, 62(2):281–290, 2011.

[Zam19] Reza Zamani. An innovative four-layer heuristic for scheduling multi-mode projects under multiple resource constrains. *RAIRO Oper. Res.*, 53(4):1309–1330, 2019.

[ZP99] Dan Zhu and Rema Padman. A metaheuristic scheduling procedure for resource-constrained projects with cash flows. *Naval Research Logistics (NRL)*, 46(8):912–927, 1999.

[ZTL06] Hong Zhang, C. M. Tam, and Heng Li. Multimode project scheduling based on particle swarm optimization. *Computer-Aided Civil and Infrastructure Engineering*, 21(2):93–103, 2006.