# Solution Approaches for Balanced Task Planning and Employee Task Distribution

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Software Engineering & Internet Computing**

eingereicht von

**Fabian Nagler, BSc**
Matrikelnummer 11776817

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dr.techn. Nysret Musliu
Mitwirkung: Univ.Ass. Dipl.-Ing. Dr.techn. Felix Winter

Wien, 11. Dezember 2024

Fabian Nagler	Nysret Musliu

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

Informatics

# Solution Approaches for Balanced Task Planning and Employee Task Distribution

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Software Engineering & Internet Computing**

by

**Fabian Nagler, BSc**
Registration Number 11776817

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dipl.-Ing. Dr.techn. Nysret Musliu
Assistance: Univ.Ass. Dipl.-Ing. Dr.techn. Felix Winter

Vienna, December 11, 2024

Fabian Nagler　　Nysret Musliu

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

# Erklärung zur Verfassung der Arbeit

Fabian Nagler, BSc

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.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 11. Dezember 2024

Fabian Nagler

# Acknowledgements

First, I am deeply grateful to Associate Prof. Dipl.-Ing. Dr. Nysret Musliu and Dr. Felix Winter for their precious advice and valuable feedback while working on this thesis. Furthermore, I want to thank our industrial partner MCP for the good cooperation.

I deeply thank my parents for their everlasting support throughout my entire life. Last but not least, I thank my girlfriend Sandra for her loving support, motivation, and tolerance also in the more stressful phases of this thesis.

The financial support by the Austrian Federal Ministry for Labor and Economy and the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

# Kurzfassung

In dieser Diplomarbeit untersuchen wir zwei für die Praxis relevante Optimierungsprobleme aus dem Bereich der Produktionsplanung. Das erste Problem, das Balanced Task Planning Problem, wird in dieser Arbeit erstmals beschrieben. Obwohl es verwandte Probleme aus der Literatur gibt, erfordern relevante Unterschiede in der Definition der Nebenbedingungen und Zielfunktionen die Einführung einer neuen Problemspezifikation. Das Ziel ist es verschiedene Aufgaben zu Planungsperioden und Maschinen zuzuordnen, sodass die Auslastungen der Maschinen ausgewogen um einen Zielwert liegen, während kritische Aufgaben priorisiert werden. Eine formale Problemspezifikation des zweiten Problems wurde kürzlich unter dem Namen Employee Task Distribution Problem veröffentlicht. Dabei handelt es sich um ein Optimierungsproblem mit dem Ziel Bedarfe zu erfüllen, indem Mitarbeiterinnen und Mitarbeiter zu Operationen zugeordnet werden. Außerdem muss eine Reihe an Nebenbedingungen berücksichtigt werden. In der Literatur wurden bereits Lösungsansätze für das Problem untersucht. In der Praxis gibt es jedoch zusätzlich den Badarf einer interaktiven Lösungsvariante, um Entscheidungsträgerinnen und Entscheidungsträger enger in den Planungsprozess einzubinden. Dies ist vor allem relevant um auf kurzfristige Veränderungen der Mitarbeiterverfügbarkeiten oder -bedarfe reagieren zu können, während die Anpassungen des vorhandenen Plans möglichst gering gehalten werden sollen.

Wir präsentieren eine Problemspezifikation für das erste Problem und beweisen, dass eine Entscheidungsvariante des Problems NP-vollständig ist. Wir stellen ein mathematisches Modell vor, welches mit modernen Constraint Programming und Integer Programming Lösungsverfahren optimale Lösungen bereitstellen kann. Außerdem schlagen wir verschiedene metaheuristische Varianten basierend auf Tabusuche vor, um qualitative Lösungen für große Probleminstanzen zu finden. Für das zweite Problem verwenden wir eine interaktive Optimierungsmethode, die nachvollziehbare Verbesserungen eines vorhandenen Plans vorschlägt. Dafür definieren wir eine interaktive Problemvariante des Employee Task Distribution Problems und schlagen einen Lösungsansatz basierend auf Tabusuche vor, der es ermöglicht, Lösungen zu dieser Problemvariante zu finden.

Um die vorgeschlagenen Methoden zu evaluieren, erzeugen wir zufällige Probleminstanzen für das Balanced Task Planning Problem und führen eine experimentelle Auswertung der verschiedenen Lösungsansätze durch. Das Ergebnis dieser Auswertungen zeigt, dass die meisten kleinen Instanzen mithilfe der exakten Methoden gelöst werden können, wobei für einige eine optimale Lösung gefunden wurde. Bei Verwendung der Tabusuche konnten hochqualitative Lösungen auch für die überwiegende Mehrheit der großen Probleminstanzen gefunden werden. Um die Effektivität der interaktiven Optimierungsmethode zu demonstrieren, untersuchen wir praktische Anwendungsfälle und zeigen die Wirksamkeit unseres Ansatzes bei der Unterstützung menschlicher Entscheidungsträgerinnen und Entscheidungsträger.

# Abstract

In this thesis, we investigate two real-life optimization problems from the area of production planning. The first problem, the Balanced Task Planning Problem, is introduced in this thesis. While there are related problems in the literature, significant differences in the definition of constraints and objectives require the introduction of a new problem specification. The aim of the problem is to assign tasks to machines and planning periods so that the machine loads are balanced around a target value while prioritizing more critical tasks. The second problem was recently introduced to the literature and is called the Employee Task Distribution Problem. The goal is to fulfill demands by assigning employees to operations while satisfying various constraints. While promising solution approaches have been proposed in the literature, users of decision support systems that solve real-life problems often prefer more involvement in the employee planning process. This is particularly relevant for reacting to short-term changes in employee availabilities or capacity demands while keeping the adaptations as minimal as possible.

For the first problem, we provide a formal problem description and prove that a decision variant of the problem is NP-complete. We introduce a mathematical model that can be used with Constraint Programming and Integer Programming solvers to find optimal solutions. Furthermore, we propose different local search variants based on Tabu Search to find solutions for large-scale problem instances. For the second problem, we propose an interactive optimization method that provides the user with traceable suggestions for improving employee assignments. For that, we introduce an interactive variant of the Employee Task Distribution Problem and propose a Tabu Search method to find solutions for this problem variant.

To evaluate the methods, we generate random problem instances for the Balanced Task Planning Problem and perform an experimental evaluation of the different approaches. The results show that most of the small instances can be solved using the exact approach, and for some, optimal solutions can be obtained. Using Tabu Search, high-quality solutions could also be found for the large majority of large-scale instances. To demonstrate the effectiveness of the interactive optimization approach, we study practical use cases and show the usefulness in supporting human decision-makers.

# Contents

CHAPTER 1

# Introduction

In the area of production planning, assigning tasks to machines or employees is a highly relevant problem. Solving such kind of problems can be challenging, especially if they include a large number of constraints and objectives. Finding high-quality solutions, however, is of high importance. In this thesis, we approach two combinatorial optimization problems that originate from the field of production planning: the Balanced Task Planning Problem and the Employee Task Distribution Problem.

The Balanced Task Planning Problem takes a set of tasks and machines as input, and the goal is to assign the tasks to machines, while a machine's load must not exceed a given maximum but should also reach a given minimum. Furthermore, there are restrictions on the compatibility between tasks and machines, i.e., not each task can be assigned to each machine. Depending on the input, not all tasks can always be planned without exceeding a machine's maximum load. Therefore, the plan can further be divided into independent planning periods. There are three optimization objectives: First, the loads of machines within a planning period should be balanced. Second, each machine's load should be as close as possible to a given target value. Third, priorities should be respected, and tasks with similar priorities should be assigned to the same period. This way, the human planner can prioritize periods containing tasks with high priority over periods with low-priority tasks.

Related problems have already been investigated in the literature. Both, the Production Leveling Problem [Vas19, VLM$^+$22] and the Balanced Academic Curriculum Problem [CM01, CDGGS12] aim at finding assignments to periods while balancing the load of each period. However, relevant differences in the definition of constraints and objectives motivate the introduction of the Balanced Task Planning Problem in this thesis.

The second part of the thesis is dedicated to the Employee Task Distribution Problem - a multi-objective optimization problem, with the task of assigning employees to operations while respecting a variety of different constraints. It has already been introduced and formally described in the literature, together with a constraint programming approach to solve the problem [HLM+24b]. This problem originates from a real-life application and is relevant for a wide range of industrial areas. Although the existing solution approaches are capable of finding high-quality solutions to real-life problem instances, there are cases in which it is necessary to include the human decision-makers (i.e., the users of the planning tool) more closely during the optimization process. This is mainly due to two reasons: First, to the users, automatic planning is a black box, i.e., they do not understand concretely why specific decisions were made. Second, they often prefer stable plans over plans retrieved from the tool. Therefore, a common approach is to copy established plans (e.g., from previous weeks) and only make small adaptations to react to changing demands or (short-term) changes in employee availability (e.g., caused by illnesses). Applying the automatic planning tool, however, would possibly change a large portion of the plan.

## 1.1 Aims of the thesis

The goals of this thesis are:

- Introduce and formally define the Balanced Task Planning Problem.
- Analyze the computational complexity of the Balanced Task Planning Problem.
- Apply exact solution methods to find optimal solutions.
- Develop a metaheuristic method to efficiently find high-quality solutions to large-scale problem instances.
- Provide diverse problem instances by proposing a random instance generator for the Balanced Task Planning Problem.
- Compare the different solution approaches in an experimental evaluation.
- Develop an interactive method to include human decision-makers in the solution process of the Employee Task Distribution Problem.
- Propose a solution method for the interactive optimization problem.
- Demonstrate the approach in practical use cases.

## 1.2 Contributions

The contributions provided in this thesis are:

- We provide an informal problem description and mathematical definition of the Balanced Task Planning Problem.

- We prove that a decision variant of the problem is NP-complete by providing a polynomial-time reduction from a well-known NP-complete problem.
- A mathematical model is developed that can be used as an exact solution approach to acquire optimal solutions.
- As a metaheuristic approach, a local search method based on (Reactive) Tabu Search is proposed that can solve large-scale instances in a reasonable time.
- We provide an instance generator that allows the generation of random instances of the Balanced Task Planning Problem.
- Using the exact approaches, we provide optimal solutions for 24 of the 100 small instances and one medium-sized instance. The metaheuristic methods provide high-quality solutions for the majority of large-scale instances.
- For the Employee Task Distribution Problem, we adapt the problem definition from the literature to be able to model user interactions and solve this interactive problem using Tabu Search.
- We provide three concrete, practical use cases to demonstrate the applicability of the interactive approach to relevant, real-life scenarios.

## 1.3 Structure of the thesis

The remainder of the thesis is structured as follows. In Chapter 2, we introduce the Balanced Task Planning Problem and provide an informal description and a mathematical problem definition. Furthermore, related problems from the literature are summarized, and the computational complexity is analyzed. Chapter 3 is dedicated to different solution methods, starting with the exact approach, where we provide a constraint programming model. Then, the different Tabu Search methods are presented. Finally, the experimental evaluation of the solution methods concludes this chapter. In chapter 4, we describe the Employee Task Distribution problem and summarize existing solution methods. The interactive optimization method for this problem is presented in chapter 5, where we also demonstrate this method in practical use cases. Finally, in chapter 6, we summarize the findings of this thesis and mention possible future work.

CHAPTER 2

# The Balanced Task Planning Problem

In this chapter, we introduce the Balanced Task Planning Problem (BTPP). After an informal problem description, we give a formal definition of the BTPP. Furthermore, related work is summarized, and similar problems are compared. In the last section of this chapter, we analyze the computational complexity of the BTPP.

## 2.1 Problem Description

The input to the BTPP consists of a set of tasks and machines. Each task has a size, a priority, and a set of machines on which the task can be planned. The goal is to assign each task to a machine so that the load of each machine is within a given range. Furthermore, the schedule can be divided into an arbitrary number of independent planning periods. The number of periods is not given as part of the problem input but can be adjusted dynamically during the solution process.

There are three objective goals which can be calculated independently for each period:

- The first goal is to balance the sum of task sizes assigned to the machines within a period. Therefore, we want to minimize the load deviation between each machine and the machine with the highest load in the period.

- The load of each machine should be as close as possible to a given target load. Hence, the goal is to minimize the absolute difference between this target value and the load of each machine.

- The priorities of tasks assigned to the same period should be as homogeneous as possible. For each period, we consider the average priority of tasks assigned to this period, and the goal is to minimize the sum of absolute differences between each task priority and the average.

As an example, consider an instance with tasks T1-T14 and three machines, where all tasks can be planned on each machine. Furthermore, we allow machine loads between 2 and 9, and we target machine loads of 7. Figure 2.1 shows a candidate solution to this instance, containing two planning periods. In period P1, the loads of all machines lie between the minimum and maximum sizes defined by the instance. The same does not hold for period P2 because task T5 only has a size of 1.

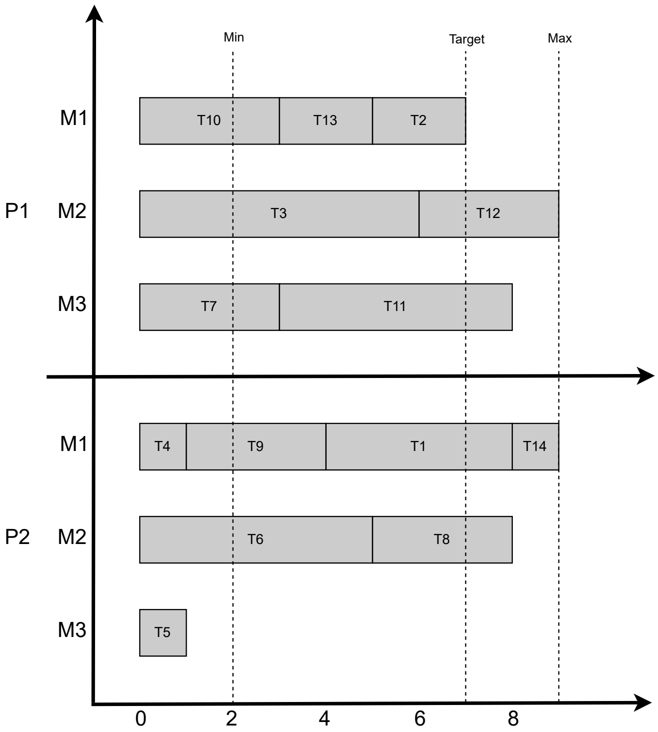


Figure 2.1: A candidate solution to an example instance of the BTPP

## 2.2 Formal Problem Definition

### 2.2.1 Problem Input

The following parameters define the input to the problem:

**Finite set of tasks**: $\mathcal{T} = \{t_1, \ldots, t_n\}$

**Finite set of machines**: $\mathcal{M} = \{m_1, \ldots, m_k\}$

**Size of task** $t$: $s_t \in \mathbb{N}^+$

**Priority of task** $t$: $pr_t \in \mathbb{N}^+$

**Minimum machine load**: $S_{min} \in \mathbb{N}$

**Maximum machine load**: $S_{max} \in \mathbb{N}^+$

**Targeted machine load**: $S_{tar} \in \mathbb{N}^+$, where $S_{min} \leq S_{tar} \leq S_{max}$

**Task-machine compatibility matrix**: $V_{t,m} \in \{0,1\}, \forall t \in \mathcal{T}, \forall m \in \mathcal{M}, V_{t,m} = 1$ iff task $t$ can be assigned to machine $m$

### 2.2.2 Variables

The following variables represent the output of the problem:

**Set of planning periods**: $\mathcal{P} = \{p_1, \ldots, p_q\}$

**Period which task** $t$ **is assigned to:** $PA_t \in \mathcal{P}, \forall t \in \mathcal{T}$

**Machine which task** $t$ **is assigned to:** $MA_t \in \mathcal{M}, \forall t \in \mathcal{P}$

The following variables act as helper variables:

**Load of a machine:** To calculate the load of a machine $m$ in period $p$, we sum up the sizes of all tasks that are assigned to $m$ and $p$.

$$MS_{p,m} = \sum_{t \in \mathcal{T}} s_t \cdot [PA_t = p \wedge MA_t = m], \forall p \in \mathcal{P}, m \in \mathcal{M}$$

**Average period priority:** To calculate the average priority of tasks assigned to a planning period $p$, we sum up the priorities of all tasks assigned to $p$ and divide this sum by the total number of tasks assigned to $p$.

$$avgPrio_p = \frac{\sum_{t \in \mathcal{T}} pr_t \cdot [PA_t = p]}{|\{t \in \mathcal{T} \mid PA_t = p\}|}, \forall p \in \mathcal{P}$$

### 2.2.3 Constraints

- The load of each machine must not be lower than the given minimum $S_{min}$. This constraint is not active for exactly one special period, which we call the *remainder period* ($p = 1$):
$$MS_{p,m} \geq S_{min}, \forall p \in \mathcal{P} \setminus 1, m \in \mathcal{M}$$

- The load of each machine must not be larger than the given maximum $S_{max}$. Note that this constraint has to be satisfied for all periods, including the remainder period:
$$MS_{p,m} \leq S_{max}, \forall p \in \mathcal{P}, m \in \mathcal{M}$$

- Each task may only be assigned to a compatible machine:
$$V_{t,MA_t} = 1, \forall t \in \mathcal{T}$$

### 2.2.4 Objective

The objective function is composed of the following three objectives:

- **Machine load deviation:** The difference between the loads of the machine with the highest load and each single machine in a period should be as low as possible:
$$f_1 = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \left( \max_{m' \in \mathcal{M}} MS_{p,m'} \right) - MS_{p,m}$$

- **Target machine load deviation:** The absolute difference between the target machine load and each machine load should be minimized:
$$f_2 = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} |S_{tar} - MS_{p,m}|$$

- **Priority deviation:** For each period, the absolute difference between the average task priority and the priorities of the single tasks assigned to this period is minimized. The average period size is rounded to the next integer to ensure the integrality of the objective value, which allows the use of Constraint Programming solvers that do not support real-valued variables. Furthermore, it enables the implementation of efficient delta evaluation procedures for the metaheuristic approaches without errors caused by the use of floating-point arithmetic:
$$f_3 = \sum_{p \in \mathcal{P}} \sum_{t \in \mathcal{T}} |\lceil avgPrio_p \rceil - pr_t| \cdot [PA_t = p]$$

The overall objective is to minimize the weighted sum of the objective components $f_1, f_2$ and $f_3$:

$$minimize\ w_1 \cdot f_1 + w_2 \cdot f_2 + w_3 \cdot f_3$$

The parameters $w_1, w_2$, and $w_3$ can be used to configure the influence of the three objective components on the overall objective value.

## 2.3 Related Work

In this section, we summarize problems from the literature that share similar properties with the BTPP.

One related problem is the Production Leveling Problem (PLP) [Vas19, VLM+22]. The goal of the PLP is to assign orders (which are comparable to tasks in the BTPP) of different types to periods while getting as close as possible to a targeted production volume and respecting order priorities. The number of periods in the PLP is fixed, while for the BTPP it is a variable that is assigned during the solution process. Another significant difference is that the PLP implicitly assumes that there is only one machine, in contrast to the BTPP, where the number of machines is part of the problem parameters. As solution methods, the authors of [Vas19, VLM+22] proposed a Mixed Integer Programming approach as well as a metaheuristic method based on Simulated Annealing.

Another problem related to the BTPP is the Balanced Academic Curriculum Problem (BACP) [CM01][CDGGS12]. The goal of this problem is to assign courses to academic periods while restricting the minimum and maximum academic load of each period. The objective is to find a curriculum with a balanced academic load among the periods. This is achieved by minimizing the maximum academic load. In another work [MSZ+07], a few more balancing criteria for the BACP were studied, including the objective to minimize the sum of deviations from the mean, which is used similarly in the BTPP for balancing the task priorities within a period. The BACP differs from the BTPP in that it lacks the notion of machines (or a similar concept) and does not allow the prioritization of courses, which is possible for the tasks of the BTPP. Furthermore, it takes the number of periods as part of the input and, in contrast to the BTPP, does not model it as a decision variable. The problem has been approached using Constraint Programming [CM01], Integer Programming, and Local Search [CDGGS12].

Restricting the BTPP to instances with only one machine yields a problem similar to the two problems above. Similarly, enforcing solutions to the BTPP to contain only one planning period leads to a generalization of the well-known and extensively studied Bin Packing Problem (BPP) [Har82]. The input to the traditional variant of the BPP consists of a finite set of items of different sizes. The objective is to assign each item to a bin while the overall number of bins should be minimized. This goal differs from the BTPP, where the number of machines is given as input, and the minimization of the number of used periods is not a direct objective. Vice versa, the Bin Packing Problem does not aim to balance the load of the bins in any way. Although the BPP was first released some decades ago, it, together with its variants, remains a relevant research topic to this day. For surveys and literature reviews, we refer to [DIM16, ME21].

Generalizations of the BPP to two or more dimensions have been extensively studied (e.g., see [CKPT17, LMMV14]). The BTPP can be seen as a special variant of two-dimensional bin packing with the machines and periods as the two dimensions. However, while the sizes of tasks in the BTPP are one-dimensional, in most of the studied two-dimensional problem variants of the BPP, the size of an item is defined by its length and width.

## 2.4 Complexity Analysis

In this section, we prove that a decision variant of the BTPP is NP-complete. This is a problem variant where the goal is to decide for an arbitrary problem instance whether there exists a solution that satisfies all constraints while not considering the optimization objectives.

**Theorem 1.** The decision variant of the *Balanced Task Planning Problem* that asks whether a feasible solution exists without considering the optimization objectives is NP-complete.

*Proof.* To show the NP-hardness of the decision variant of the BTPP, we give a polynomial time reduction from the Bin Packing Problem [Har82], which is known to be NP-complete in the strong sense. Formally, it is defined as follows:

| **Bin Packing Problem (BPP)** | |
|---|---|
| **Instance:** | A finite set $U$ of items, a size $s_u \in \mathbb{N}^+$ for each $u \in U$, a positive integer bin capacity $B$, and a positive integer $K$. |
| **Question:** | Is there a partition of $U$ into disjoint sets $U_1, \ldots, U_K$ such that the sum of the sizes of the items in each $U_i$ is $B$ or less? |

Consider an arbitrary instance of the Bin Packing problem with a set $U$ and positive integers $B$ and $K$.

We construct an instance of the BTPP in the following way:

- We use a bijective function $g$ to uniquely map each item to a task. We denote the inverse function of $g$ by $g^{-1}$:

$$\mathcal{T} = \{g(u_1), \ldots, g(u_{|U|})\}$$

- The number of machines is $K+1$: $\mathcal{M} = \{m_1, \ldots, m_{K+1}\}$
- The size of a task refers to the size of the corresponding item:

$$s_t = s_{g^{-1}(t)}, \forall t \in \mathcal{T}$$

- As the task priority is only relevant for calculating the objective value, which is not part of the considered decision variant of the BTPP, we can set it to an arbitrary value:
$$pr_t = 1, \forall t \in \mathcal{T}$$
- Each task can be assigned to each machine except for one special machine $M_0$:
$$V_{t,m} = 1, \forall t \in \mathcal{T}, \forall m \in \mathcal{M} \setminus M_0$$
$$V_{t,M_0} = 0, \forall t \in \mathcal{T}$$
- The minimum machine load is set to 1:
$$S_{min} = 1$$
- The maximum and target machine loads are set to the bin capacity
$$S_{max} = S_{tar} = B$$
- All objective weights are set to 1
$$w_0 = w_1 = w_2 = 1$$

We clearly see that this transformation from an instance of the BPP to an instance of the BTPP can be done in time polynomial to all of the input parameters.

To show that BTPP is NP-hard, it remains to show that given an instance of the Bin Packing Problem, there exists a partition of $U$ into disjoint sets $U_1, \ldots, U_K$ with $\sum_{u \in U_i} s_u \leq B$, if and only if there exists a feasible solution to the BTPP instance retrieved from the reduction.

Consider an arbitrary instance of BPP and assume that there is a partition of $U$ into disjoint sets $U_1, \ldots, U_K$ with $\sum_{u \in U_i} s_u \leq B$. We construct a solution to the BTPP instance as follows: We set $\mathcal{P} = \{p_1\}$ and assign each task to the single period, i.e. $PA_t = p_1$ for all tasks $t \in \mathcal{T}$. Furthermore, we identify each set $U_i$ from the solution to the BPP with a machine $M_i$ and assign the tasks to $M_i$ for which the corresponding items are assigned to $U_i$. Formally, this means that $MA_t = m_i$ if and only if $g^{-1}(t) \in U_i$. We know that $\sum_{u \in U_i} s_u \leq B$ and by construction of the reduction it follows that $\sum_{u \in U_i} s_{g^{-1}(u)} \leq S_{max}$. Therefore, the constraints enforcing the maximum machine sizes are satisfied. Furthermore, there is only one period, which means that the constraints enforcing the minimum machine sizes are not active, and as the task-machine compatibilities are only restricted for $M_0$ (to which no tasks are assigned), there are no constraint violations. Consequently, there is a feasible solution to the instance of the BTPP.

Now, consider an arbitrary instance of BTPP that has a feasible solution and was retrieved from a BPP instance by the reduction. This instance cannot contain any tasks that are assigned to machine $M_0$ because $V_{t,M_0} = 0$ for all $t \in \mathcal{T}$. Furthermore, the minimum machine load $S_{min}$ is set to 1, which means that the constraints that enforce the minimum machine load for $M_0$ would be violated for each period, except for the remainder period, for which this constraint is not active. Consequently, the solution can only contain one period $p_1$. Now, we construct a partition of $U$ into disjoint sets $U_1, \ldots, U_K$, by setting $U_i = \{u \mid g(u) \in M_i\}$. As there is only one period and $MS_{p_1,m} \leq S_{max}$ holds for all machines, it follows directly from the construction of the reduction that $\sum_{u \in U_i} s_u \leq B$.

Finally, it remains to show that the BTPP is a member of the complexity class NP. This is done by proving the existence of a polynomial-time verifier, which is an algorithm that, given a problem instance together with a solution, decides in polynomial time whether the solution is feasible. In a feasible solution, the number of periods is polynomial in the number of tasks because otherwise, there would be periods to which no task is assigned, which would violate the minimum machine size constraints. This means that the number of constraints enforcing the minimum and maximum machine loads is polynomial in $|\mathcal{T}| \cdot |\mathcal{M}|$. Similarly, the number of constraints for task-machine compatibility is polynomial in the number of tasks. Therefore, a polynomial-time verifier is obtained by checking each of the constraints.

□

CHAPTER 3

# Solution Approaches for the Balanced Task Planning Problem

In this chapter, we present solution methods for the BTPP. First, we propose a mathematical model to find solutions using Constraint Programming and Mixed Integer Programming solvers. Then, we introduce a construction heuristic to create initial solutions. Furthermore, we present metaheuristic approaches based on Tabu Search. Finally, we evaluate all proposed methods by conducting a large number of experiments. The experimental results are presented and discussed at the end of this chapter.

## 3.1 Constraint Programming Model

In this section, we provide a Constraint Programming model that can be used to solve the BTPP. The model is similar to the formal problem description presented in Chapter 2.2, however, there are a few differences in the definition of some of the constraints. Therefore, we only describe these differences in more detail.

### 3.1.1 Input Parameters

**Finite set of tasks**: $\mathcal{T} = \{t_1, \ldots, t_n\}$

**Finite set of machines**: $\mathcal{M} = \{m_1, \ldots, m_k\}$

**Finite set of periods**: $\mathcal{P} = \{p_1, \ldots, p_q\}$

**Size of task** $t$: $s_t \in \mathbb{N}^+$

**Priority of task** $t$: $pr_t \in \mathbb{N}^+$

**Minimum machine load**: $S_{min} \in \mathbb{N}$

**Maximum machine load**: $S_{max} \in \mathbb{N}^+$

**Targeted machine load**: $S_{tar} \in \mathbb{N}^+$, where $S_{min} \leq S_{tar} \leq S_{max}$

**Set of forbidden machines per task**: $F_t \subseteq \mathcal{M}$, for all $t \in \mathcal{T}$

Machine-task compatibilities are represented as a set of machines $F_t$ to which a task $t$ cannot be assigned, i.e., $m \in F_t$ if and only if $V_{t,m} = 0$. Furthermore, the maximum number $|\mathcal{P}|$ of periods is given as part of the input because the number of decision variables depends on $|\mathcal{P}|$. When solving an instance, we set this number to a theoretical upper bound for the number of periods. In any feasible solution, the minimum load in each period, except for the remainder period, is $S_{min} \cdot |\mathcal{M}|$. Consequently, ignoring the remainder period, there cannot be more than $\frac{\sum_{t \in \mathcal{T}} s_t}{S_{min} \cdot |\mathcal{M}|}$ periods. Adding the remainder period, we set $|\mathcal{P}| = \left\lfloor \frac{\sum_{t \in \mathcal{T}} s_t}{S_{min} \cdot |\mathcal{M}|} \right\rfloor + 1$.

### 3.1.2 Decision Variables

**Period which task** $t$ **is assigned to:** $PA_t \in \mathcal{P}, \forall t \in \mathcal{T}$

**Machine which task** $t$ **is assigned to:** $MA_t \in \mathcal{M}, \forall t \in \mathcal{T}$

**Load of a machine:** $MS_{p,m} \in \mathbb{N}, \forall p \in \mathcal{P}, m \in \mathcal{M}$

**Load of a period:** $PS_p \in \mathbb{N}, \forall p \in \mathcal{P}$

### 3.1.3 Constraints

In this section we list all constraints of the Constraint Programming model. As most of them are similar to the constraints definded in Section 2.2.3, we only discuss the differences in more detail.

- The load of machine $m$ in period $p$ is the sum of all tasks assigned to $t$ and $p$:

$$MS_{p,m} = \sum_{t \in \mathcal{T}} s_t \cdot [PA_t = p \wedge MA_t = m], \forall p \in \mathcal{P}, m \in \mathcal{M}$$

- The size of a period is the maximum load of its machines:

$$PS_p = \max_{m \in \mathcal{M}} MS_{p,m}, \forall p \in \mathcal{P}$$

- The load of each machine must not be lower than the given minimum $S_{min}$. This constraint is not active for the *remainder period* ($p = 1$):

$$MS_{p,m} \geq S_{min}, \forall p \in \mathcal{P} \setminus 1, m \in \mathcal{M}$$

- The size of each machine must not be larger than the given maximum $S_{max}$. Note that this constraint has to be satisfied for all periods, including the remainder period:

$$MS_{p,m} \leq S_{max}, \forall p \in \mathcal{P}, m \in \mathcal{M}$$

- If a machine $m$ is contained in the set of forbidden resources $F_t$ of a task $t$, the machine $m$ cannot be assigned to $t$. This replaces the constraint from the formal description, which enforces the task-machine compatibilities using the matrix $V_{t,m}$.

$$MA_t \neq m, \forall t \in \mathcal{T}, \forall m \in F_t$$

- To enhance the model's performance, we add the following symmetry breaking constraint. It eliminates symmetries by ensuring that the periods are ordered by their loads (ignoring the first period, which is always the remainder period). This constraint is redundant, i.e., in contrast to the other constraints, it is not required for the correctness of the model.

$$PS_p \leq PS_{p-1}, \forall p \in \{3, \ldots, |\mathcal{P}|\}$$

### 3.1.4 Objective

As objective, we minimize the same function described in Section 2.2.4.

## 3.2 Construction Heuristic

In this section, we present a construction heuristic that is used to generate solutions to instances of the BTPP. In the upcoming sections, we will use this approach to create initial solutions for the metaheuristic approaches. Furthermore, it is used by the random instance generator described in Section 3.7. It is a greedy algorithm that, in each iteration, assigns a task to a period and machine. At each stage of the algorithm, we consider just one period. In each iteration, we find the machine with the least load, which the task can be assigned to without violating any constraint, and plan the task on this machine. If no such machine exists, a new period is created, which the current and subsequent tasks are assigned to.

**Algorithm 3.1:** Construction heuristic for the BTPP

```
   Input: Sorted list of tasks $\mathcal{T}$, set $\mathcal{M}$ of machines, Task-machine compatibility
          matrix $V$, and positive integer $S_{max}$
   Output: Number of periods, Period assignments $PA_t$, and machine assignments
           $MA_t$ for each task $t$
 1 machine_load_m ← 0, for all machines $m \in \mathcal{M}$
 2 current_period ← 1
 3 for $t$ in $\mathcal{T}$ do
 4     selected_machine ← $\arg\min_{m \in \{m' \in \mathcal{M} | V_{t,m'}=1\}}$ machine_load_m
 5     free_space ← $S_{max}$ − machine_load_selected_machine
 6     if free_space < $s_t$ then
 7         current_period ← current_period + 1
 8         machine_load_m ← 0, for all machines $m \in \mathcal{M}$
 9         selected_machine ← first machine $m$, with $V_{t,m} = 1$
10     end
11     $PA_t$ = current_period
12     $MA_t$ = selected_machine
13     machine_load_selected_machine ← machine_load_selected_machine +
       $s_t$
14 end
15 return current_period, $PA_t$, and $MA_t$ for each task $t$
```

Algorithm 3.1 shows the construction heuristic in more detail. As input, it takes the tasks $\mathcal{T}$ and machines $\mathcal{M}$, as well as the task-machine compatibility matrix $V$, and the maximum allowed load of a machine $S_{max}$. Note that the tasks are given as a sorted list, so that the user of the algorithm can influence the order in which the tasks are assigned. To find good solutions (e.g. as initial solutions for local search), we first sort the tasks by their priorities. This way, the tasks with similar priorities tend to be assigned to the same period. The number of periods and assignments of each task to a period and a machine are returned as output. First, the algorithm initializes the load of each machine to 0 and sets the current period to 1. Then, we iterate over the tasks in the given order. In each iteration, the machine with the least load, on which the task can be planned according to the matrix $V$, is stored in the variable `selected_machine`. In line 5, the free space on this machine is calculated as the difference between the maximum load $S_{max}$ and the load of the selected machine. If there is not enough space for the current task, a new period is added by incrementing the period counter (`current_period`) and resetting the load of all machines. Furthermore, a new machine is selected by finding the first one compatible with the current task. In lines 11-12, the task is assigned to the current period and the selected machine. Finally, the load of the selected machine is increased by the size of the assigned task.

## 3.3 Tabu Search for the Balanced Task Planning Problem

In this section, we propose a metaheuristic method based on Tabu Search to solve the BTPP.

### 3.3.1 Solution Representation

For the local search approach, we represent a solution as a list of planning periods. Each of these periods is represented as a list of machines, and each machine as a list of tasks that are assigned to the enclosing machine and period. Furthermore, a single integer representing the index of the remainder period is stored with each candidate solution.

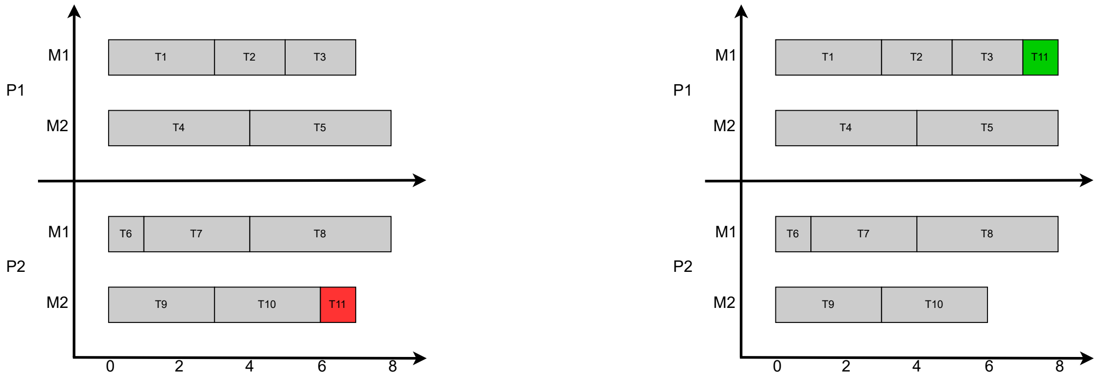
We ensure that the constraints enforcing the maximum load $S_{max}$ and the task-machine compatibilities are satisfied in each stage of the algorithm by the definition of the move operators (see next section). In contrast, the constraints concerning the minimum allowed machine loads can be violated during the search process. For each violation of these constraints, we add a penalty to the cost function multiplied by a constant value bigM.

$$f_{\mathcal{L}} = w_1 \cdot f_1 + w_2 \cdot f_2 + w_3 \cdot f_3 + \text{bigM} \cdot \text{penalty}$$
$$\text{penalty} = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}} \max(0, MS_{p,m} - S_{min})$$

The constant bigM is chosen so that each solution with at least one constraint violation has a cost higher than each feasible solution. To achieve that, we calculate an upper bound for each of the three objectives:

- **Machine load deviation**: In the worst case, each period has exactly one assigned task. Then, the number of periods is equal to the number of tasks, and each machine, except for the machine to which the single task in the period is assigned, has a deviation of $s_t$ to the loaded machine.
$$max_1 = \sum_{t \in \mathcal{T}} s_t \cdot (|\mathcal{M}| - 1)$$
- **Target machine load deviation**: As above, in the worst case, each task is assigned to its own period. The maximum deviation from $S_{tar}$ per machine is either the space between 0 and $S_{tar}$ or $S_{tar}$ and $S_{max}$.
$$max_2 = |\mathcal{T}| \cdot |\mathcal{M}| \cdot max(S_{tar}, S_{max} - S_{tar})$$
- **Priority deviation**: The deviation of each task priority to the average priority of the period cannot be larger than the difference between the largest and lowest priority.
$$max_3 = |\mathcal{T}| \cdot \max_{t \in \mathcal{T}} pr_t - \min_{t \in \mathcal{T}} pr_t$$

Therefore, we set $\text{bigM} = max_1 + max_2 + max_3 + 1$.



(a) Before the shift move is applied

(b) After the shift move is applied

Figure 3.1: An example of a shift task move

### 3.3.2 Neighborhood Structure

In this section, we present the search moves we used to generate the neighborhood of a solution.

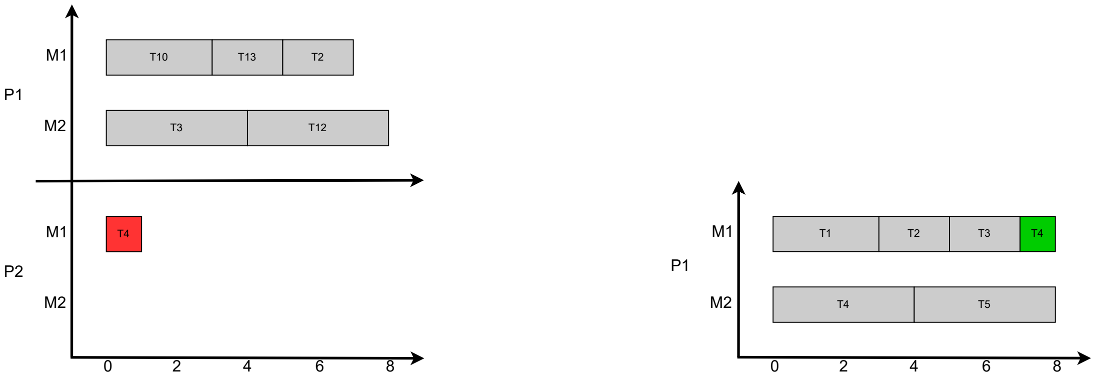
**Shift Task Neighborhood**

A *Shift Task* move shifts a task $t$ to a machine $m$ and period $p$. This means that task $t$ is removed from its current machine $m_0$ and period $p_0$ and assigned to machine $m$ and period $p$. Note that a shift move can also be applied if the current machine is equal to the machine the task is shifted to, and the same holds for the periods. However, we require a change in either the machine or the period, i.e., $m_0 \neq m$ or $p_0 \neq p$, as otherwise, the move would not change the solution. Figure 3.1 shows an example of a shift move, where task $T11$ is moved from machine $M2$ of period $P2$ to machine $M1$ of period $P1$.

A special occurrence of a *Shift Task* move is the shift of the only task of a period to another period. In this case, the period, which is empty after the move is applied, is removed from the solution (see Figure 3.2). Furthermore, if a task is shifted to a period that is not yet part of the solution, it will be added.

**Swap Tasks Neighborhood**

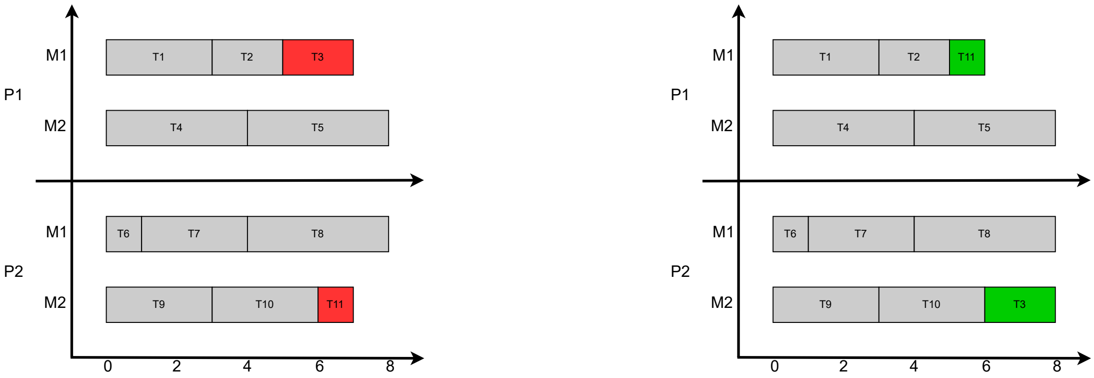
A *Swap Task* move exchanges the positions of two tasks $t_1$ and $t_2$. If $t_1$ is assigned to machine $m_1$ and period $p_1$, and task $t_2$ is assigned to machine $m_2$ and period $p_2$ before applying the move, $t_1$ is assigned to $m_2$ and $p_2$, while $t_2$ is assigned to $m_1$ and $p_1$ after the application. This type of move is only applicable if the two tasks $t_1$ and $t_2$ are not assigned to the same period and machine (i.e., $m_1 = m_2$ and $p_1 = p_2$) because applying the move would not have an effect. An example of this move can be seen in Figure 3.3, where tasks $T3$ and $T11$ are exchanged.



(a) Before the shift move is applied

(b) After the shift move is applied

Figure 3.2: An example of a shift task move where the single task of the second period is removed



(a) Before the swap move is applied

(b) After the swap move is applied

Figure 3.3: An example of a swap tasks move

### 3.3.3 Tabu Search

In this section, we propose a local search algorithm to solve the BTPP. A basic local search approach can be seen in Algorithm 3.2. The algorithm takes a problem instance and an initial solution to this instance as input and then tries to iteratively improve the solution by replacing it with the best solution in its neighborhood. As exit condition, we use a wall-clock time limit. This type of local search algorithm is also known as *hill climber*.

However, one problem with this basic approach is that it improves the solution only until no more solution in the neighborhood is better than the current solution. Then, the search process gets stuck in a local optimum. To avoid this problem, more advanced local search algorithms are applied. Tabu Search (TS) [Glo86] is one of the most prominent approaches for escaping local optima. The idea behind Tabu Search is to avoid returning

**Algorithm 3.2:** A basic local search procedure (hill climber) to improve a solution to the BTPP

```
Input: An instance $I$ of the BTPP and an initial solution $S$ to this instance
Output: An improved solution to instance $I$
current_solution = $S$
best_solution = $S$
while exit condition not met do
    neighborhood ← generate_neighborhood()
    current_solution ← find_best_solution(neighborhood)
    if cost(current_solution) < cost(best_solution) then
        best_solution ← current_solution
    end
end
return best_solution
```

to parts of the search space that have already been seen. This is achieved using the notion of *tabu lists*. In each iteration of the search process, the current solution is added to the tabu list. In future iterations, the solutions contained in the tabu list will not be considered when searching the neighborhood. Storing solutions explicitly in the tabu list is computationally expensive, because each solution in the neighborhood would have to be compared to each solution in the tabu list. Therefore, in most tabu list implementations, only attributes of the solution or the move applied to reach the solution are stored in the tabu list. To control the duration for which these attributes are tabu, a meta-parameter, the *tabu tenure*, is introduced to specify the length of the tabu list. In other words, the tabu tenure controls the number of iterations in which added solutions (or its attributes) are tabu. Furthermore, in our implementation, we do not explicitly remove a solution/attribute from the tabu list once it is no longer tabu. Rather, we store the iteration until it is tabu alongside the solutions/attributes in the tabu list, and can then check whether it is still tabu in the current iteration.

For the BTPP, we propose the following two types of tabu lists:

**Tabu List 1: Fix task**
The idea of this type of tabu list is to consider the move that is used to reach a solution. Whenever a move is applied, the tasks touched by the move are added to the tabu list. For a *SwapTasks* move, this means that both tasks that are exchanged are made tabu. As long as a task is part of the tabu list, the assignments of this task to a machine and period cannot be changed, i.e., they are fixed.

**Tabu List 2: Forbid task source**
In the second tabu list, we again consider the tasks affected by the applied move. Let $t$ be such a task, and $m_0$ and $p_0$ the machine and period, which $t$ is assigned to before the move application. When the move is applied, we add the tuple $(t, m_0, p_0)$ to the tabu list, which means that a move is considered tabu if $t$ would again be moved to machine $m_0$

and period $p_0$. In other words, we forbid the task from returning to the exact position to which it was assigned before the move was applied.

Furthermore, there can be conditions that justify the tabu status of a solution or move to be overwritten, also known as *aspiration criteria*. Often, a solution is accepted if it is better than the best solution found so far, even if it is contained in the tabu list. This is also the aspiration criterion we apply in our Tabu Search approach.

**Algorithm 3.3:** A Tabu Search algorithm for the BTPP

```
   Input: An instance $I$ of the BTPP and an initial solution $S$ to this instance
   Output: An improved solution to instance $I$
 1 current_solution = $S$
 2 best_solution = $S$
 3 tabu_list = []
 4 while exit condition not met do
 5     neighborhood ← generate_neighborhood()
 6     best_neighbor ← null
 7     for neighbor in neighborhood do
 8         if cost(neighbor) < cost(current_neighbor) and
            (nottabu_list.contains(neighbor) or cost(neighbor) <
            cost(best_neighbor)) then
 9             best_neighbor ← neighbor
10         end
11     end
12     if best_neighbor ≠ null then
13         current_solution ← best_neighbor
14         if cost(current_solution) < cost(best_solution) then
15             best_solution ← current_solution
16         end
17         tabu_list.add(current_solution)
18     end
19 end
20 return best_solution
```

Algorithm 3.3 shows our Tabu Search approach in more detail. First, the current and best solution found so far are set to the initial solution, and the tabu list is initialized. Then, the following steps are repeated in a loop until a stopping criterion is met (i.e., a time-limit is reached). In each iteration, we first generate the neighborhood of the current solution and then search for the best neighbor, that is not tabu. This is done in the loop starting at line 7. Note that in line 8, we apply the aspiration criterion by overwriting the tabu status of a neighbor if it is better than the best solution found so far. In line 12, we check whether a valid neighbor was found, i.e., a solution in the neighborhood that is not tabu or satisfies the aspiration criterion. If this is not the case, the current solution is not changed in this iteration. Otherwise, we set the current solution to the best neighbor, and if it is better than the best solution found so far, we also update the value of `best_solution`. Finally, we add the current solution to the tabu list.

### 3.3.4 Reactive Tabu Search

An extension of the Tabu Search is the Reactive Tabu Search (RTS) [BT94]. As discussed above, standard Tabu Search relies on the tabu tenure parameter, which defines the number of iterations a solution remains in the tabu list. The value of the tabu tenure highly influences the performance of Tabu Search [PMSB14]. Furthermore, the optimal setting of the tabu tenure often depends on properties of the instance space, like the size of an instance. Reactive Tabu Search tries to solve these issues by adapting the tabu tenure during the search process. This is done by keeping track of the solutions found during the search process. Whenever we return to an already seen solution, we increase the tabu tenure. If no cycles are detected for a certain period of time, the tabu tenure is decreased again. This approach does not only have the advantage that it is not required to set the tabu tenure before the execution of the algorithm but also adapts the tabu tenure to a value beneficial for the current part of the search space.

Another component of RTS is a more radical strategy to escape from local optima. While the tabu list allows the escape from short cycles in the search, it often struggles with larger cycles. Therefore, a sequence of random moves is applied whenever the number of encountered cycles exceeds a certain threshold. This is often referred to as *diversification strategy* and is an important part of high-quality Tabu Search implementations [GL97].

Algorithm 3.4 shows this approach in more detail. As before, the algorithm starts with initializing the current and best solution. Additionally, the variables `chaotic`, `iterations_since_last_change`, and `average_cycle_length` are set to the initial value 0. These three variables are explained in more detail below. Note that the tabu tenure is initialized with 1 in line 7. In line 8, a data structure is initialized, which stores the history of solutions encountered during the search process. This data structure is implemented as a hash table, which uses a solution as key and stores an object as values, which provides information about the last iteration (`last_iteration`) and the number of times (`repetitions`) a solution was encountered. The mechanism of RTS that changes the tabu tenure if necessary is done by `adapt_tabu_tenure` and is explained below. This function returns a flag indicating whether it is necessary to execute the more radical escape strategy. If this is not the case, the algorithm proceeds like the standard tabu search presented above, with an additional else case in line 27, where the tabu tenure is decreased if no applicable neighbor is found.

Algorithm 3.5 shows the reaction mechanism. First, it increments the variable `iterations_since_last_change`, which stores the number of iterations that elapsed since the tabu tenure has been changed. Then, the hash table `solution_history` is searched to find potential cycles. If a cycle is found, its length is calculated by the difference between the current iteration and the last iteration, in which this solution has been encountered. The last iteration is then set to the current iteration, and the number

**Algorithm 3.4:** A Reactive Tabu Search algorithm for the BTPP

```
Input: An instance $I$ of the BTPP and an initial solution $S$ to this instance
Output: An improved solution to instance $I$
current_solution = $S$
best_solution = $S$
chaotic ← 0
iterations_since_last_change ← 0
average_cycle_length ← 0
tabu_list = []
tabu_list.tenure ← 1
solution_history = {}
while exit condition not met do
    is_escape_necessary ← adapt_tabu_tenure()
    if is_escape_necessary then
        escape()
    else
        neighborhood ← generate_neighborhood()
        best_neighbor ← null
        for neighbor in neighborhood do
            if cost(neighbor) < cost(current_neighbor) and
             (not tabu_list.contains(neighbor) or cost(neighbor) <
             cost(best_neighbor)) then
                best_neighbor ← neighbor
            end
        end
        if best_neighbor ≠ null then
            current_solution ← best_neighbor
            if cost(current_solution) < cost(best_solution) then
                best_solution ← current_solution
            end
            tabu_list.add(current_solution)
        else
            tabu_list.tenure ← tabu_list.tenure · DECREASE_FACTOR
            iterations_since_last_change ← 0
        end
    end
end
return best_solution
```

**Algorithm 3.5:** The reaction mechanism that adapts the tabu tenure in RTS

```
function adapt_tabu_tenure() begin
    iterations_since_last_change ← iterations_since_last_change + 1
    cycle ← solution_history.find(current_solution)
    if cycle was found then
        cycle_length ← current_iteration − cycle.last_iteration
        cycle.last_iteration ← current_iteration
        cycle.repetitions ← cycle.repetitions +1
        if cycle.repetitions > REPETITION_THRESHOLD then
            chaotic ← chaotic + 1
            if chaotic > CHAOS then
                chaotic ← 0
                return true
            end
        end
        tabu_list.tenure ← tabu_list.tenure · INCREASE_FACTOR
        iterations_since_last_change ← 0
        average_cycle_length ← 0.9 · average_cycle_length + 0.1 · cycle_length
    else
        solution_history.add(key: current_solution, value: (last_iteration: current_iteration, repetitions: 1))
    end
    if iterations_since_last_change > average_cycle_length then
        tabu_list.tenure ← tabu_list.tenure · DECREASE_FACTOR
        iterations_since_last_change ← 0
    end
    return false
end
```

of repetitions of the current solution is incremented. In line 8, we check whether this number of repetitions exceeds a threshold, which is given to the algorithm as parameter `REPETITION_THRESHOLD`. If this is the case, the counter `chaotic` is incremented, and if it is larger than the parameter `CHAOS`, it is reset to 0, and the algorithm returns *true*, indicating that it is necessary to apply the escape strategy. In the case that no escape is required, we increase the tabu tenure by multiplying it with the parameter `INCREASE_FACTOR`. Furthermore, we adopt the average cycle length, which is calculated as the moving average, where the current value is multiplied by 0.9 and the length of the cycle observed in this iteration, multiplied by 0.1, is added. If no cycle is detected in an iteration, this means that the current solution has not been seen before. In this case, the solution is added to the solution history, where the last iteration is set to the current iteration and the number of repetitions is initialized with 1 (line 19). If the tabu tenure is unchanged for a number of iterations larger than the average cycle length, we decrease it by multiplying it with the parameter `DECREASE_FACTOR`. Finally, we return *false* to indicate that no application of the escape strategy is required.

**Algorithm 3.6:** The escape strategy used in RTS

```
1  function escape() begin
2      solution_history ← {}
3      random_moves ← $1 + (1 + \texttt{rand()}) \cdot \frac{\texttt{average\_cycle\_length}}{2}$
4      for i in 1 ... random_moves do
5          current_solution ← find_random_neighbor()
6          if cost(current_solution) < cost(best_solution) then
7              best_solution ← current_solution
8          end
9          tabu_list.add(current_solution)
10     end
11 end
```

In Algorithm 3.6, we present the strategy to escape from local optima. First, the solution history is reinitialized, i.e., all solutions are removed from the hash table. Then, the number of random moves that shall be applied is calculated with the formula in line 3, where the function `rand()` returns a random number in the interval $[0, 1)$. In this way, the number of random moves will always be between $\frac{\texttt{average\_cycle\_length}}{2} + 1$ and $\texttt{average\_cycle\_length} + 1$. In the loop, we first randomly choose a solution from the neighborhood as the new value for the current solution. As above, we replace the best solution with the current solution if it is better and add the new solution to the tabu list.

### 3.3.5 Restrict neighborhood

In each iteration, standard Tabu Search algorithms explore the entire neighborhood to find the best neighbor. This means that each of the solutions in the neighborhood has to be evaluated to decide which neighbor is the best. This can become very time-consuming, especially for large-scale instances. In this section, we propose two approaches to solve this issue.

**Only explore fraction of neighborhood**
The most basic way is to only search a restricted portion of the neighborhood. Every time we have to search for the best neighbor in the neighborhood, we consider each neighbor only with a probability $F_{exp}$, which is a parameter used to control the fraction of the neighborhood that should be explored.

**Elite Candidate List**
Another way to restrict the number of evaluated solutions is to use a candidate list approach [GL97]. In this thesis, we use an Elite Candidate List, which is based on the assumption that a promising move in the current iteration will also yield good neighbors in the subsequent iterations. In the first iteration, the entire neighborhood is explored, and the best $k$ neighbors are added to the candidate list. In the succeeding iterations, only the moves stored in the candidate list are used to generate the neighborhood. The best move is then removed from the candidate list and applied to the current solution. Note that it can happen that between the creation of the candidate list and the current iteration, the solution changed in a way that not each move from the candidate list is applicable. If this is the case, this specific move is ignored when searching for the best move. If the candidate list is empty, none of the moves are applicable, or the quality of the best move in the neighborhood does not reach a certain threshold, the candidate list is rebuilt as described above. We set this threshold to the quality of the worst move at the point in time the candidate list is constructed. This approach to set the quality threshold was also used by [AHA09].

## 3.4 Experimental Evaluation

In this section, we experimentally evaluate the solution methods presented in the last sections. First, we propose a random instance generator to create problem instances we can use for the experimental evaluation and then describe the experimental setting. In Section 3.4.3, we examine the results of the experiments conducted using the exact solution approaches. Section 3.4.4 is devoted to the experimental evaluation of the metaheuristic approaches. Finally, in Section 3.4.5, we make an overall comparison of the results obtained by exact and metaheuristic approaches.

### 3.4.1 Problem Instances

In this section, we propose a random instance generator to create instances of the BTPP, which we can use for the experimental evaluation of our approaches. Algorithm 3.7 shows our approach to generate one random instance. As input, it takes upper and lower bounds for the number of tasks ($t_{min}, t_{max}$), number of machines ($m_{min}, m_{max}$), sizes of tasks ($s_{min}, s_{max}$), and the number of different priority levels ($pr_{min}, pr_{max}$). Furthermore, it receives an upper bound $ms_{max}$ for the maximum machine load. First, we randomly determine some properties of the instances using the function rand$(x, y)$, which returns a random integer between $x$ and $y$. The minimum machine load ($S_{min}$) gets assigned a random value between the upper bound of the task size and the upper bound of the machine load. This ensures that the largest task still fits in a machine. Then, the maximum machine load ($S_{max}$) is chosen to be between $S_{min}$ and the upper bound. Now, the target machine load ($S_{tar}$) has to lie between $S_{min}$ and $S_{max}$. The number of tasks (task_n), machines (machine_n) and priority levels (priority_levels) are randomly chosen from the intervals $[t_{min}, t_{max}]$, $[m_{min}, m_{max}]$, and $[pr_{min}, pr_{max}]$, respectively. In the loop starting at line 8, we generate tasks with randomly chosen sizes between $s_{min}$ and $s_{max}$ and priorities between 1 and priority_levels, and add them to the list of tasks. In line 15, the task-machine compatibilities are temporarily set to 1 for all tasks and machines. This way, we can apply the construction heuristic from Algorithm 3.1 to generate a reference solution. Note that the construction heuristic processes the tasks in the order they are generated, i.e., we do not sort them by the priorities and sizes as we do when generating initial solutions for the local search. The result of the greedy algorithm is the number of periods (period_n), period assignments $PA$, and machine assignments $MA$. However, this solution may not be feasible as it can contain machines with a load lower than $S_{min}$. This means, it is not guaranteed that the generated instance is satisfiable. Therefore, we apply a procedure to fill these gaps, such that each machine load is at least as high as the lower bound $S_{min}$. Afterward, we set the valid machines of each task to 0 for the machines it is not assigned to in the reference solution (line 19). Then, we randomly choose the probability that an arbitrary task is compatible with an arbitrary machine. Therefore, for each task-machine pair $(t, m)$, we set its compatibility $V_{t,m}$ to 1 with this probability.

Algorithm 3.8 shows the method we apply to fill the gaps between the load of a machine and its lower bound $S_{min}$. This is done by checking whether the current machine load $MS_{\texttt{period,machine}}$ is lower than $S_{min}$. If this is the case, we randomly choose a machine load between $S_{min}$ and $S_{max}$ and calculate the size of the gap to fill, which is the difference between the desired and actual machine load. Then, we do the following steps while there is still a gap, i.e., gap_size $> 0$. We create a task by randomly choosing its size and priority in the same way as above. If this task is larger than the gap, we set it to the size of the gap. Finally, we assign this task to the current machine and period and subtract the size of the task from the gap size.

### 3.4.2 Experimental setting

Using the random instance generator, we created a total of 345 problem instances, which are evenly divided into small, medium, and large instances. Small instances contain between 20 and 200 tasks, medium instances between 201 and 600, and large instances between 601 and 1000 tasks. Furthermore, the weights $w_1 - w_3$ of the three objective components are all set to 1. The remaining value ranges can be seen in Table 3.1. From each instance set (small, medium, large), we chose 100 instances for the experimental evaluation and 15 for tuning the local search parameters.

**Algorithm 3.7:** Random instance generator

```
Input: Intervals of task number $[t_{min}, t_{max}]$, machine number $[m_{min}, m_{max}]$,
       task sizes $[s_{min}, s_{max}]$, priority levels $[pr_{min}, pr_{max}]$ and maximum
       machine load $ms_{max}$
Output: A randomly generated instance
$S_{min} \leftarrow$ rand($s_{max}, ms_{max}$)
$S_{max} \leftarrow$ rand($S_{min}$, $ms_{max}$)
$S_{tar} \leftarrow$ rand($S_{min}$, max_machine_size)
task_n $\leftarrow$ rand($t_{min}, t_{max}$)
machine_n $\leftarrow$ rand($m_{min}, m_{max}$)
priority_levels $\leftarrow$ rand($pr_{min}, pr_{max}$)
tasks $\leftarrow$ []
for $i$ in $1 \ldots$ task_n do
    $s \leftarrow$ rand($s_{min}, s_{max}$)
    $pr \leftarrow$ rand(1, priority_levels)
    task $\leftarrow$ create new task of size $s$ and priority $pr$
    tasks.add(task)
end
machines $\leftarrow$ {1, ..., machine_n}
$V_{t,m} \leftarrow 1$, for each task $t$ and machine $m$
period_n, $PA$, $MA \leftarrow$ construct_solution(tasks, machines, $V$, $S_{max}$)
fill_gaps()
$V_{t,m} \leftarrow 0$, for each task $t$ and machine $m$ where $MA_t \neq m$
valid_machine_probability $\leftarrow$ rand()
for $t$ in tasks do
    for $m$ in machines do
        if valid_machine_probability > rand() then
            $V_{t,m} \leftarrow 1$
        end
    end
end
```

**Algorithm 3.8:** Algorithm to fill gaps to minimum size

```
function fill_gaps() begin
    for period in $1 \dots$ period_n do
        for machine in machines do
            if $MS_{period,machine} < S_{min}$ then
                machine_load ← rand($S_{min}$, $S_{max}$)
                gap_size ← machine_load - $MS_{machine}$
                while gap_size > 0 do
                    $s$ ← rand($s_{min}$, $s_{max}$)
                    if $s$ > gap_size then
                        $s$ ← gap_size
                    end
                    gap_size ← gap_size - $s$
                    $pr$ ← rand(1, priority_levels)
                    task ← create new task of size $s$ and priority $pr$
                    tasks.add(task)
                    $PA_{task}$ ← period
                    $MA_{task}$ ← machine
                end
            end
        end
    end
end
```

| Machines | Task size | Priority levels | Machine size (upper bound) |
|---|---|---|---|
| $[1, 10]$ | $[1, 100]$ | $[1, 20]$ | 500 |

Table 3.1: The valid value ranges for the generation of random problem instances

All experiments, including parameter tuning runs, were executed on a computation cluster with 13 nodes, each with 20.48 GB RAM and an Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz processor with 24 cores. For the exact methods, we used eight cores, as some of the used solvers can benefit from running the solution process in parallel. For the evaluation of the metaheuristic approaches, we conducted each experiment five times using a single CPU core. Each algorithm run, both for the exact and metaheuristic methods, was executed using a wall-clock time limit of 10 minutes.

### 3.4.3 Evaluation of the exact methods

We implemented the Constraint Programming model proposed in Section 3.1 using the solver-independent modeling language MiniZinc [NSB+07] and conducted experiments using the solvers Gurobi 11.0.3 [Gur24], Google OR-Tools CP-SAT 9.10 [PD24], Coin-Or

| | Gurobi | OR-Tools | Coin | Chuffed |
|---|---|---|---|---|
| Small | 86% | 99% | 27% | 62% |
| Medium | 7% | 47% | 0% | 4% |
| Large | 0% | 15% | 0% | 0% |

Table 3.2: Percentages of instances for which a feasible solution was found using the exact methods

| | Gurobi | OR-Tools | Coin | Chuffed |
|---|---|---|---|---|
| Small | 24% | 19% | 9% | 3% |
| Medium | 1% | 1% | 0% | 0% |
| Large | 0% | 0% | 0% | 0% |

Table 3.3: Percentages of instances for which an optimal solution was found using exact methods

2.10.11 [Lou03] and Chuffed 0.13.2. [CSS$^+$]. With the exception of Chuffed, these solvers support parallel execution, and we therefore assign them eight CPU cores for the experiments. We applied each solver to all 300 instances from the validation set. The detailed results can be seen in Appendix A. Table 3.2 shows for each solver the percentages of instances for which a feasible solution was found. For the small instances, feasible solutions to all but one instance could be found using OR-Tools. Gurobi managed to find feasible solutions to 86 of the small problem instances, followed by Chuffed with 62 and Coin with 27 solved instances. For the medium-sized problem instances, only OR-Tools could find feasible solutions for nearly half of the instances, while Gurobi and Chuffed managed to find solutions only for a small portion of the medium-sized instances, and Coin could not find a single feasible solution. Only with OR-Tools could feasible solutions be found for large-scale instances, which, however, could be achieved for only 15 of the large instances. Gurobi and OR-Tools found the largest number of optimal solutions among all the solvers. However, even these solvers did not find optimal solutions for any of the large-scale instances and only for a single medium-sized instance. The exact percentages of optimal solutions found for each instance set can be seen in Table 3.3.

There are 25 small instances for which all four solvers found a feasible solution. Figure 3.4 shows the relative objective values (i.e., the objective value found using the specific method divided by the best objective found using any solver for the respective instance) for the four solvers on these 25 small instances. It can be seen that Gurobi and OR-Tools clearly produced improved results compared to Chuffed and Coin for the large majority of the instances. To compare Gurobi and OR-Tools in more detail, we considered the 86 small instances for which both solvers found a feasible solution. The results of this experiment are shown in the boxplot in Figure 3.5. It can be seen that Gurobi and OR-Tool perform similarly well on these instances, with a slightly better performance of OR-Tools.

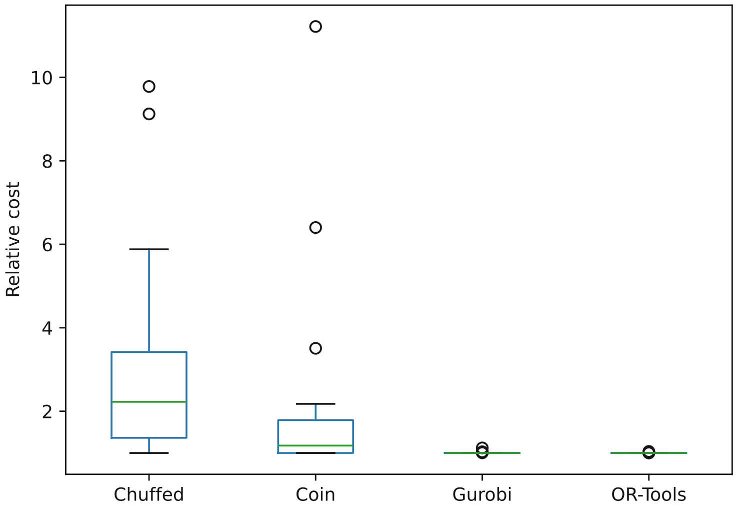


Figure 3.4: A boxplot showing the relative objective value for the 25 small instances for which all solvers could find a feasible solution

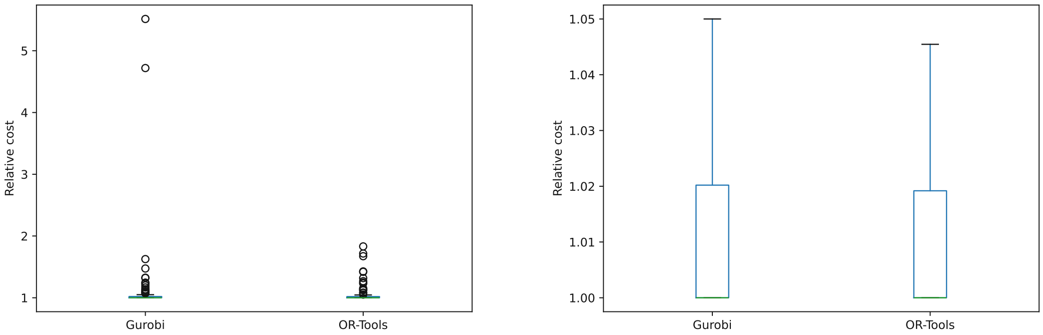


Figure 3.5: A boxplot showing the relative objective value for the 86 small instances for which Gurobi and OR-Tools could find a feasible solution with (left) and without (right) extreme outliers

### 3.4.4 Evaluation of the metaheuristic methods

In this section, we evaluate the performance of the proposed metaheuristic methods. More precisely, we compare a basic hill-climbing algorithm (HC), standard Tabu Search (TS), Reactive Tabu Search (RTS), and Reactive Tabu Search with restricted neighborhood exploration (RTS-RN).

| Parameter | Methods | Type | Range |
|---|---|---|---|
| Tabu list type ($TL$) | TS, RTS, RTS-RN | Categorical | (TL1, TL2) |
| Tabu tenure ($T$) | TS | Integer | 1-1000 |
| Chaos ($C$) | RTS, RTS-RN | Integer | 1-10 |
| Repetition threshold ($RT$) | RTS, RTS-RN | Integer | 1-10 |
| Increase factor ($\alpha_+$) | RTS, RTS-RN | Real | 1-10 |
| Decrease factor ($\alpha_-$) | RTS, RTS-RN | Real | 0-1 |
| Candidate list size ($S_{CL}$) | RTS-RN | Integer | 1-1000 |
| Explore neighborhood fraction ($F_{exp}$) | RTS-RN | Real | 0-1 |

Table 3.4: The value ranges for the single parameters

|  | $TL$ | $T$ | $C$ | $RT$ | $\alpha_+$ | $\alpha_-$ | $S_{CL}$ | $F_{exp}$ |
|---|---|---|---|---|---|---|---|---|
| Small, TS | TL2 | 31 | - | - | - | - | - | - |
| Medium, TS | TL2 | 4266 | - | - | - | - | - | - |
| Large, TS | TL1 | 86 | - | - | - | - | - | - |
| Small, RTS | TL1 | - | 7 | 3 | 7.2319 | 0.5651 | - | - |
| Medium, RTS | TL1 |  | 8 | 2 | 6.2994 | 0.8849 | - | - |
| Large, RTS | TL1 | - | 2 | 6 | 7.3013 | 0.9544 | - | - |
| Small, RTS-RN | TL2 | - | 7 | 1 | 3.4759 | 0.509 | 51 | 0.9084 |
| Medium, RTS-RN | TL2 | - | 1 | 2 | 4.3866 | 0.1685 | 131 | 0.7849 |
| Large, RTS-RN | TL2 | - | 1 | 1 | 7.3512 | 0.1013 | 924 | 0.634 |

Table 3.5: The tuned parameter settings for the single methods and instance sets found by irace

As discussed above, the behavior of these methods depends on the settings of various algorithm parameters. To find high-quality parameter settings, we use the parameter tuning tool irace 3.5 [LDPC$^+$16]. For each metaheuristic method (except for the hill climber, which does not rely on parameter settings), we used irace to tune the parameters for each instance set. Table 3.4 shows a summary of the single parameters, together with their data types and the value ranges given to irace. Using the instances from the training set, we executed a single tuning run for each combination of method and data set. The number of algorithm runs for each tuning run was set to 10,000, and the time limit for each algorithm run was set to 10 minutes. The resulting parameter settings retrieved from irace are depicted in Table 3.5.

To compare the performance of the single metaheuristic methods, we conducted experiments using all 300 instances from the test set using the tuned parameter settings. As some of the methods contain randomized components, we repeated each algorithm run five times using different random seeds. If, for a method, at least one of these runs does not return a feasible solution, we report that the respective instance could not be solved using this method. The detailed results of these experiments can be seen in Appendix B.

Table 3.6 shows the number of instances for which a feasible solution was found in each

| | HC | TS | RTS | RTS-RN |
|---|---|---|---|---|
| Small | 97% | 99% | 100% | 100% |
| Medium | 96% | 98% | 98% | 98% |
| Large | 94% | 97% | 97% | 98% |

Table 3.6: Percentages of instances for which a feasible solution was found using the exact methods

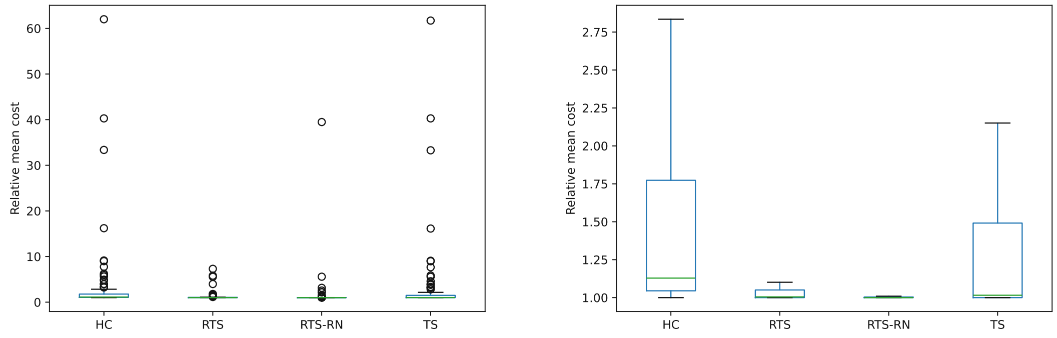


Figure 3.6: Boxplots showing the relative mean cost for the metaheuristic methods on the 97 small instances for which all methods found feasible solutions in each run with (left) and without (right) extreme outliers

of the five algorithm runs. It can be seen that using the metaheuristic methods, most of the instances could be solved. There were only two medium and two large instances for which none of the methods could find a feasible solution.

To compare the metaheuristic approaches, we calculated the relative mean costs for each instance by dividing the mean objective value for each method by the best mean objective value achieved for this instance using any of the metaheuristic approaches. This means that the method with the lowest mean objective value has a relative mean cost of 1.0. The relative costs for all problem instances can be seen in Table B.2 in the appendix. We make the comparison based on the results of the 97 small, 96 medium, and 94 large instances, for which all four metaheuristic methods managed to find a feasible solution in each of the five algorithm runs.

The boxplots in Figure 3.6 show the relative mean costs for the small instances. It can be seen that the three methods based on Tabu Search managed to produce better results than the basic hill climber. Furthermore, more high-quality results could be found using the reactive variants of Tabu Search. Restricting the search neighborhood could further improve the results. However, there are some outliers. For instance *small_86*, for example, the best mean solution is found by RTS, while the mean cost of solutions found using RTS-RN was almost 40 times higher. Another outlier worth mentioning is instance *small_34*, for which the best mean result can be observed using TS, while RTS

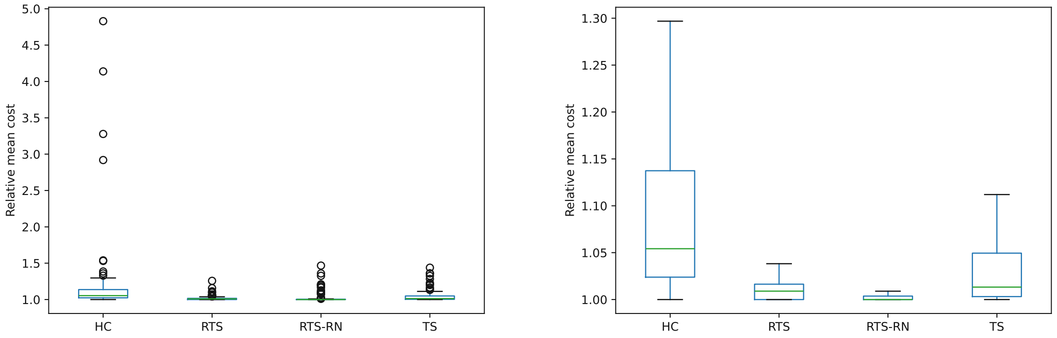


Figure 3.7: Boxplots showing the relative mean cost for the metaheuristic methods on the 96 medium instances for which all methods found feasible solutions in each run with (left) and without (right) extreme outliers

and RTS-RN both have a relative mean cost of around 13.

Figure 3.7 shows the boxplots visualizing the results for the medium-sized instances. Again, RTS-RN produces the best results, followed by RTS and TS, which work considerably better than the hill climber. Furthermore, the observed outliers are not as severe as for the small instances.

The results for the large instances can be seen in the boxplots in Figure 3.8. As expected, the restriction of the search neighborhood has the most effect on the large instances. Furthermore, it can be observed that the quality of the solutions found by RTS and TS are comparably good. A more fine-grained comparison between the three Tabu Search variants is shown in Figure 3.9 It can be seen that the TS even performs slightly better than the RTS.

### 3.4.5 Overall comparison

In this section, we compare the experimental results for exact methods and the metaheuristic approaches. For the metaheuristic methods, we consider the best results found over all of the five runs. For the large majority of instances, a feasible solution could be found. Regarding the instances *medium_26*, *medium_86*, and *large_10*, none of the proposed methods was able to obtain a solution. Furthermore, a feasible solution to instance *large_69* could only be found using RTS-RN in one of the five algorithm runs.

Table 3.7 shows for each solution method the percentages of instances for which the best known solution could be found. It can be seen that RTS-RN provides the largest number of best upper bounds. For the small instances, OR-Tools and Gurobi performed comparably well regarding the number of best known solutions found. A boxplot visualizing the relative best cost (i.e., the cost divided by the cost of the best known solution) for the small instances using the best metaheuristic approach RTS-RN and the OR-Tools solver

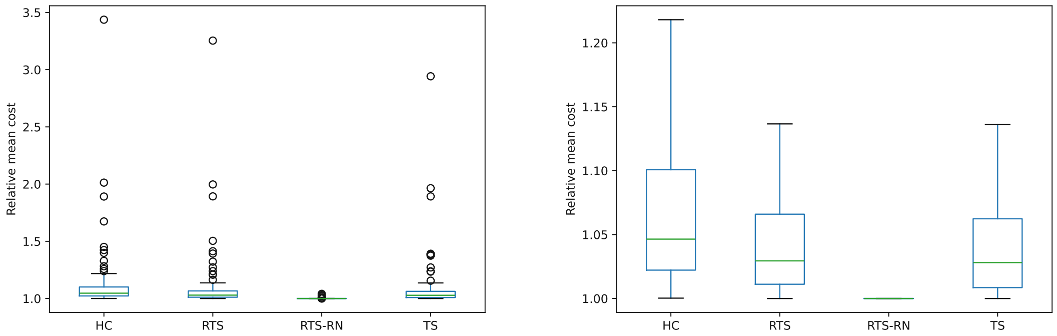


Figure 3.8: Boxplots showing the relative mean cost for the metaheuristic methods on the 94 large instances for which all methods found feasible solutions in each run with (left) and without (right) extreme outliers

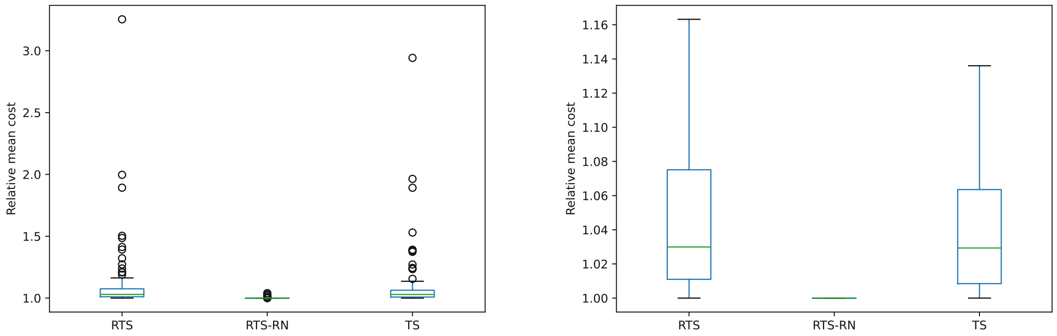


Figure 3.9: Boxplots showing the relative mean cost for the Tabu Search based methods on the 97 large instances for which all three methods found feasible solutions in each run with (left) and without (right) extreme outliers

(which is the solver that produced the most feasible solution) can be seen in Figure 3.10. It shows that these two methods produced comparably good solutions on a large number of the small instances. However, there is a respectable number of instances for which OR-Tools found considerably better solutions than RTS-RN. A similar behavior could be observed for the 47 medium-sized instances, for which both methods provided feasible solutions (see Figure 3.11). An optimal solution to 24 of the small instances could be found. Solutions with the same optimal cost could be found for 18 instances using RTS-RN, 17 using RTS, 16 using TS, and 4 using HC. Regarding the medium-sized instances, one optimal solution could be obtained using the exact methods. For the large instances, none of the approaches could provide optimal solutions.

| | Gurobi | OR-Tools | Chuffed | Coin | HC | TS | RTS | RTS-RN |
|---|---|---|---|---|---|---|---|---|
| Small | 44% | 47% | 9% | 3% | 4% | 24% | 28% | 49% |
| Medium | 2% | 26% | 0% | 0% | 1% | 6% | 15% | 57% |
| Large | 0% | 0% | 0% | 0% | 0% | 4% | 3% | 94% |

Table 3.7: Percentages of instances for which the best known solution was found by the single solution approaches

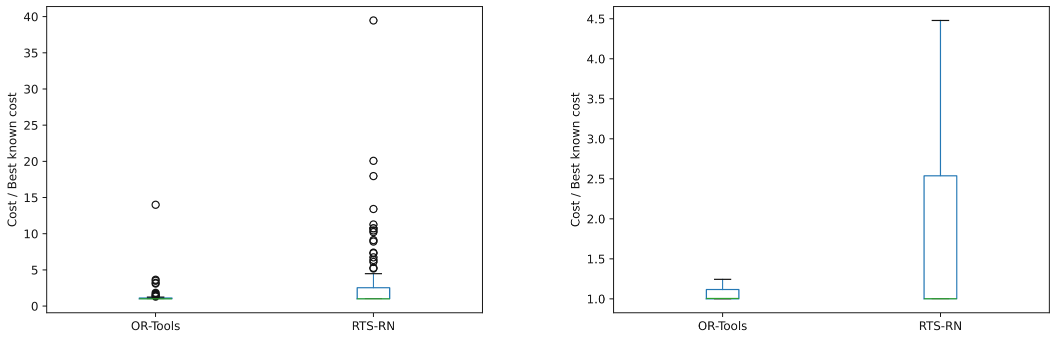


Figure 3.10: Boxplots showing the relative best cost for the OR-Tools and RTS-RN on the 99 small instances for which a feasible solution could be found by both methods with (left) and without (right) extreme outliers

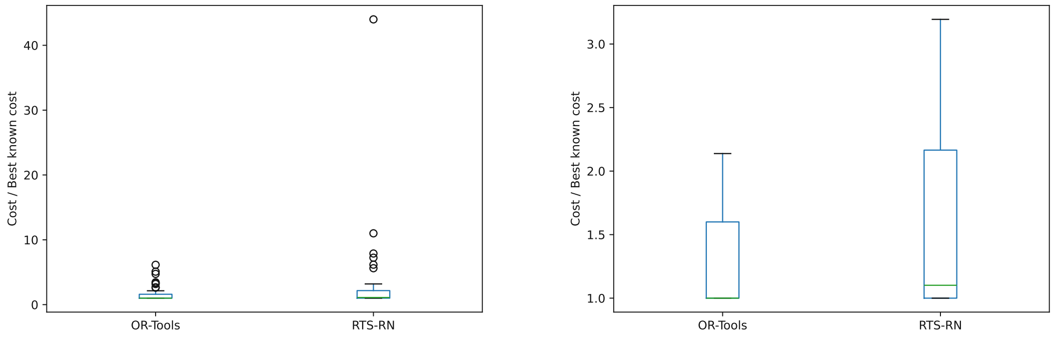


Figure 3.11: Boxplots showing the relative best cost for the OR-Tools and RTS-RN on the 47 medium instances for which a feasible solution could be found by both methods with (left) and without (right) extreme outliers

# CHAPTER 4

# The Employee Task Distribution Problem

In this chapter, we introduce the Employee Task Distribution Problem. First, we give an informal problem description followed by a mathematical definition. Furthermore, a state-of-the-art metaheuristic solution approach using local search is presented.

## 4.1 Problem Description

The Employee Task Distribution Problem (ETDP) [HLM$^{+}$24b] is a multi-objective optimization problem where the goal is to assign employees to operations while fulfilling several constraints. The input to the ETDP consists of sets of employees, operations, time buckets, and qualifications. A solution is represented as an assignment of a capacity to each triple of time bucket, operation, and employee. In a feasible solution, employees must be qualified for the operations they are assigned to. A supplied capacity is given for each employee in each time bucket, which must not be exceeded by the total capacity assigned to this employee in the respective bucket. Similarly, in each time bucket, an operation demands a certain amount of capacity, which must not be exceeded. Furthermore, an operation can have a minimum capacity, which has to be assigned for the operation to be active in a time bucket. In other words, the sum of assigned capacities in the bucket is either at least at the given minimum or no capacities are assigned to the operation in this bucket. In addition, a maximum number of parallel operations in a bucket can be given for an operation. Then, for each employee assigned to this operation, the total number of operations it is assigned to must not exceed this maximum in any time bucket.

The ETDP is a multi-objective optimization problem. Therefore, a variety of different objectives are optimized at the same time. The basic goal is to maximize the total sum of capacities, i.e., the satisfied capacity demands. Priorities can be assigned to employees such that certain employees can be lexicographically preferred over others. In practice, this is relevant for considering the distinction between permanent and temporary staff, where unfulfilled supplies on the former should have a greater impact. Similarly, priorities can be assigned to operations, which can be used to model the higher importance of bottleneck operations. Furthermore, for each operation-employee pair, a qualification score is given, defining the level of qualification an employee has for the operation. In addition, if an employee is assigned to an operation in a time bucket, there can be cases where it is preferable that the employee is assigned to the same operation in the subsequent time bucket. The last objective is to minimize the total number of (non-zero) capacity assignments. The purpose of this objective is to ensure that solutions in which employees are assigned to fewer operations are preferred, which in practice should satisfy employees and lead to a better outcome of their work.

## 4.2 Formal Problem Definition

In this section, we give a formal definition of the ETDP, which was first proposed in [HLM+24b].

### 4.2.1 Problem Input

The following parameters define the input to the problem:

**Finite set of employees**: $\mathcal{E} = \{1, 2, \dots, m\}$

**Finite set of operations**: $\mathcal{O} = \{1, 2, \dots, n\}$

**Finite set of time buckets**: $\mathcal{B} = \{1, 2, \dots, u\}$

**Supplied capacity of employee $e$ in bucket $b$ :** $sc(b, e) \in \mathbb{N}$

**Priority of employee $e$ in bucket $b$**: $ep(b, e) \in \mathbb{N}^+$

**Demanded capacity of operation $o$ in bucket $b$**: $sc(b, o) \in \mathbb{N}$

**Priority of operation $o$ in bucket $b$**: $op(b, o) \in \mathbb{N}^+$

**Minimum capacity assigned to operation $o$ in bucket $b$**: $mc(b, o) \in \mathbb{N}$

**Maximum total number of operation each employee assigned to operation** $o$ **can be assigned to in bucket** $b$: $mp(b, o) \in \mathbb{N}$

**Qualification matrix**: $Q$, $Q(o, e) \in \mathbb{N}$, $Q(o, e) > 0$ iff employee $e$ is qualified for operation $o$

### 4.2.2 Variables

The following variables represent the output of the problem:

**Amount of capacity assigned to employee** $e$ **at operation** $o$ **in bucket** $b$: $A(b, o, e) \in \mathbb{N}$

### 4.2.3 Constraints

- If a positive capacity is assigned to employee $e$ at operation $o$ in any time bucket $b$, employee $e$ must be qualified for operation $o$:

$$A(b, o, e) > 0 \implies Q(o, e) > 0, \forall b \in \mathcal{B}, o \in \mathcal{O}, e \in \mathcal{E}$$

- The sum of capacities assigned to an employee in a time bucket must not exceed the supplied capacity of the employee in the bucket:

$$\sum_{o \in \mathcal{O}} A(b, o, e) \leq sc(b, e), \forall b \in \mathcal{B}, \forall e \in \mathcal{E}$$

- The sum of capacities assigned to an operation in a time bucket must not exceed the demanded capacity of the operation in the bucket:

$$\sum_{e \in \mathcal{E}} A(b, o, e) \leq dc(b, o), \forall b \in \mathcal{B}, \forall o \in \mathcal{O}$$

- If a positive capacity is assigned to operation $o$ in time bucket $b$, the sum of capacity assigned to operation $o$ in bucket $b$ must not be lower than the given minimum $sc(b, o)$:

$$\sum_{e \in \mathcal{E}} A(b, o, e) = 0 \vee \sum_{e \in \mathcal{E}} A(b, o, e) \geq mc(b, o), \forall b \in \mathcal{B}, \forall o \in \mathcal{O}$$

- For each operation $o$ with a positive maximum number of parallel operations in bucket $b$ (i.e., $mp(b, o) > 0$), the total number of employees assigned to operation $o$ in bucket $b$ must not exceed the maximum $mp(b, o)$:

$$|\{i \in \mathcal{O} \mid A(b, i, e) > 0\}| \leq mp(b, o), \forall b \in \mathcal{B}, \forall o \in \mathcal{O}, \forall e \in \mathcal{E},$$
$$\text{with } mp(b, o) = 0 \wedge A(b, o, e) > 0$$

### 4.2.4 Objective

The objective function is composed of the six components $f_1 - f_6$.

Regarding the objective concerning the lexicographic optimization of employee priority levels ($f_2$), an assignment $A_1$ is considered better (i.e., has a larger value for $f_2$), if for any priority level $p$, the sum of capacities assigned to employees are equal at priority levels smaller than $p$ (more important employees) and greater for $A_1$ for the employees at each priority level larger than $p$ (less important employees). The lexicographic optimization of employee priorities ($f_3$) uses the same concept. This is achieved using big-M constants $\mathrm{bigM}^e(b,p)$ and $\mathrm{bigM}^o(b,p)$ for each bucket $b$ and priority level $p$, which are multiplied to the sum of assigned capacities at the respective priority level. We refer to [HLM+24a] for a detailed definition of these big-M constants.

- **Maximize sum of assigned capacities:**: The basic goal is to maximize the total sum of assigned capacities

$$f_1 = \sum_{b \in \mathcal{B}} \sum_{o \in \mathcal{O}} \sum_{e \in \mathcal{E}} A(b,o,e)$$

- **Employee priorities:** Lexicographic optimization of employee priorities; $p_E$ denotes the maximum employee priority:

$$f_2 = \sum_{b \in \mathcal{B}} \sum_{p=1}^{p_E} \mathrm{bigM}^e(b,p) \left( \sum_{o \in \mathcal{O}} \sum_{e \in \mathcal{E}} A(b,o,e) \cdot [ep(b,e) = p] \right)$$

- **Operation priorities:** Lexicographic optimization of operation priorities; $p_O$ denotes the maximum operation priority:

$$f_3 = \sum_{b \in \mathcal{B}} \sum_{p=1}^{p_O} \mathrm{bigM}^o(b,p) \left( \sum_{e \in \mathcal{E}} \sum_{o \in \mathcal{O}} A(b,o,e) \cdot [op(b,o) = p] \right)$$

- **Maximize qualification scores**: Maximize the sum of assigned capacities weighted by qualification scores of the involved employee and operation:

$$f_4 = \sum_{b \in \mathcal{B}} \sum_{o \in \mathcal{O}} \sum_{e \in \mathcal{E}} (A(b,o,e) \cdot Q(o,e))$$

- **Minimize time bucket changes**: Minimize the number of times an employee has positive assigned capacities in a time bucket $b$ but not the consecutive bucket $b+1$.

$$f_5 = \sum_{b \in \{1,\ldots,u-1\}} \sum_{o \in \mathcal{O}} \sum_{e \in \mathcal{E}} |[(A(b,o,e) > 0] - [A(b+1,o,e) > 0]|$$

- **Minimize assignment count**: Minimize the number of bucket-operation-employee triples with positive assigned capacity:

$$f_6 = |\{b \in \mathcal{B}, o \in \mathcal{O}, e \in \mathcal{E} \mid A(b,o,e) > 0\}|$$

To combine $f1 - f6$ to one objective function, which is then supposed to be minimized, the maximization goals must be converted to minimization goals. To achieve this, upper bounds are calculated, from which the maximization objectives are subtracted, yielding a non-negative objective value.

- **Upper bound for sum of assigned capacities**: For each time bucket, the sum of assigned capacities cannot be larger than the total sum of demanded capacities nor the sum of supplied capacities:

$$max_1 = \sum_{b \in \mathcal{B}} min\left(\sum_{o \in \mathcal{O}} dc(b,o), \sum_{e \in \mathcal{E}} sc(b,e)\right)$$

- **Upper bound for employee priorities objective**: Both the sum of supplied capacities and the sum of demanded capacities for an employee and operation are upper bounds for the sum of assigned capacities, from which the smaller bound is chosen:

$$max_2 = \sum_{b \in \mathcal{B}} \sum_{p=1}^{p_E} \text{bigM}^e(b,p) \cdot \min\left(\sum_{o \in \mathcal{O}} dc(b,o), \sum_{e \in \mathcal{E}} sc(b,e) \cdot [ep(b,e) = p]\right)$$

- **Upper bound for operation priorities objective**: Again, the supplies and demands are used to calculate the upper bound:

$$max_3 = \sum_{b \in \mathcal{B}} \sum_{p=1}^{p_O} \text{bigM}^o(b,p) \cdot \min\left(\sum_{e \in \mathcal{E}} sc(b,e), \sum_{o \in \mathcal{O}} dc(b,o) \cdot [op(b,o) = p]\right)$$

- **Upper bound for qualification score objective**: The upper bound for the qualification score objective is calculated similarly as for the basic objective, with the difference that additionally, the maximum qualification score over all employees and operations is multiplied to the sum of demands and supplies, respectively:

$$max_4 = \sum_{b \in \mathcal{B}} min\left(\sum_{o \in \mathcal{O}} (dc(b,o) \cdot \max_{e \in \mathcal{E}} Q(o,e)), \sum_{e \in \mathcal{E}} (sc(b,e) \cdot \max_{o \in \mathcal{O}} Q(o,e))\right)$$

The aggregated objective value is then calculated in the following way:

$$f = w_1 \cdot (max_1 - f_1) + w_2 \cdot (max_2 - f_2) + w_3 \cdot (max_3 - f_3) + w_4 \cdot (max_4 - f_4) + w_5 \cdot f_5 + w_6 \cdot f_6$$

, where $w_1 - w_6$ are the weights with which the influence of the single components can be controlled.

## 4.3 Local Search

In the same work that first introduced the ETDP [HLM+24b], exact methods using a constraint programming approach were proposed. In another (at the time of the finalization of this thesis) unpublished work, the same authors also developed a local search algorithm to solve the problem. This algorithm is based on Simulated Annealing [KGV83] and is used by our industrial partner to solve practical problem instances. In this section, we present the solution representation and neighborhood structure they used in their local search approach.

### 4.3.1 Solution Representation

For local search, a solution is represented as a three-dimensional array $A$, with the time buckets as first, operations as second, and employees as third dimension. This means that if a capacity of $c$ is assigned to the employee $e$ at operation $o$ in time bucket $b$, then $A[b][o][e] = c$. The moves of the neighborhood structure ensure that no employee is assigned to an operation for which they are not qualified. Similarly, a move does not cause the supplied or demanded capacities to be violated for an employee or operation. This, however, does not hold for the constraints enforcing the minimum capacity $mc(b, a)$ and maximum total number of operations $mp(b, o)$, which can be violated throughout the search process. Therefore, a penalty is added to the objective function for each violation of these constraints:

- The penalty term for the minimum capacity constraints is the number of bucket-operation pairs for which the constraint is violated.

$$\text{penalty}_{mc} = \sum_{b \in \mathcal{B}} \sum_{o \in \mathcal{O}} \left[ 0 < \sum_{e \in \mathcal{E}} A(b, o, e) < m(b, o) \right]$$

- For the penalty regarding the maximum total number of operations, first, the number $ao(b, e)$ of operations that employee $e$ is assigned to in time bucket $b$ is calculated. For each bucket-operation-employee triple that violates the constraint, the penalty is the difference between the actual number of operations $ao(b, e)$ and the allowed maximum $mp(b, o)$.

$$ao(b, e) = |\{o \in \mathcal{O} \mid A(b, o, e) > 0\}|, \forall b \in \mathcal{B}, \forall e \in \mathcal{E}$$

$$\text{penalty}_{mp} = \sum_{b \in \mathcal{B}} \sum_{o \in \mathcal{O}} \sum_{e \in \mathcal{E}} \begin{cases} \max(ao(b, e) - mp(b, o), 0), & \text{if } mp(b, o) > 0 \wedge A(b, o, e) > 0 \\ 0, & \text{otherwise} \end{cases}$$

To ensure that each feasible solution has an objective value less than each solution with constraint violations, a big-M constant is multiplied to the penalty term. To calculate this constant, the total upper bound of the objective value of a feasible solution is required. In Chapter 4.2.4, upper bounds $max_1 - max_4$ for the first four objectives were presented. The remaining upper bounds are calculated as follows:

| | E1 | E2 | E3 | E4 |
|---|---|---|---|---|
| **O1** | 4 | 0 | 0 | 0 |
| **O2** | 4 | 8 | 0 | 4 |
| **O3** | 0 | 0 | 0 | 4 |

(a) Before the *Change Single Assignment* move is applied

| | E1 | E2 | E3 | E4 |
|---|---|---|---|---|
| **O1** | 4 | 0 | 0 | 0 |
| **O2** | 4 | 5 | 0 | 4 |
| **O3** | 0 | 0 | 0 | 4 |

(b) After the *Change Single Assignment* move is applied

Figure 4.1: An example of a *Change Single Assignment* move in a time bucket $B1$

- **Upper bound for time bucket changes objective**: For each operation-employee pair $(o, e)$, the fact that $e$ is assigned to $o$ can at most change $|\mathcal{B}| - 1$ times between time buckets:
$$max_5 = (|\mathcal{B}| - 1) \cdot |\mathcal{O}| \cdot |\mathcal{E}|$$
- **Upper bound for assignment count objective**: An upper bound is given by the total number of assignments:
$$max_6 = |\mathcal{B}| \cdot |\mathcal{O}| \cdot |\mathcal{E}|$$

The big-M value is, therefore, the sum of the upper bounds of all objective components, to which 1 is added to get a value larger than the theoretical maximal objective:

$$\text{bigM} = max_1 + max_2 + max_3 + max_4 + max_5 + max_6 + 1$$

The objective function used for the local search $f_L$ is then calculated by adding the penalties multiplied by the big-M constant to the objective function $f$ from Chapter 4.2.4:

$$f_L = f + bigM \cdot (\text{penalty}_{mc} + \text{penalty}_{mp})$$

### 4.3.2 Neighborhood Structure

The neighborhood of a solution is generated using three types of moves.

**Change single assignment** A *Change Single Assignment* move adds or subtracts capacity from a bucket-operation-employee triple. This type of move is specified by a tuple $(b, o, e, c)$ and has the effect that when applied to an assignment $A$, the resulting neighbor $A'$ has $A'(b, o, e) = A(b, o, e) + c$. For all other values, $A'$ has the same assignments as $A$. Figure 4.1 shows an example of the application of a move $(B1, O2, E2, -3)$, where in the time bucket $B1$, the capacity which is assigned to employee $E2$ in operation $O2$ is deduced by 3.

**Shift with fixed employee**: A move of this type is defined by a tuple $(b, e, o_1, o_2, c)$ and has the effect that in bucket $b$ for employee $e$ a capacity of $c$ is shifted from operation $o_1$ to $o_2$. In Figure 4.2, an example can be seen where the move $(B1, E2, O2, O3, 5)$ is

|  | E1 | E2 | E3 | E4 |
|---|---|---|---|---|
| **O1** | 4 | 0 | 0 | 0 |
| **O2** | 4 | 8 | 0 | 4 |
| **O3** | 0 | 0 | 0 | 4 |

(a) Before the shift move is applied

|  | E1 | E2 | E3 | E4 |
|---|---|---|---|---|
| **O1** | 4 | 0 | 0 | 0 |
| **O2** | 4 | 2 | 0 | 4 |
| **O3** | 0 | 6 | 0 | 4 |

(b) After the shift move is applied

Figure 4.2: An example of a *Shift* move with fixed employee in a time bucket $B1$

|  | E1 | E2 | E3 | E4 |
|---|---|---|---|---|
| **O1** | 4 | 0 | 0 | 0 |
| **O2** | 4 | 8 | 0 | 4 |
| **O3** | 0 | 0 | 0 | 4 |

(a) Before the shift move is applied

|  | E1 | E2 | E3 | E4 |
|---|---|---|---|---|
| **O1** | 4 | 0 | 0 | 0 |
| **O2** | 4 | 0 | 8 | 4 |
| **O3** | 0 | 6 | 0 | 4 |

(b) After the shift move is applied

Figure 4.3: An example of a *Shift* move with fixed operation in a time bucket $B1$

applied, which takes 6 of the 8 time units assigned to employee $E2$ and shifts them from operation $O2$ to operation $O3$.

**Shift with fixed operation**: In a similar fashion, for an operation $o$ in bucket $b$, $c$ time units can be shifted from one employee $e_1$ to another employee $e_2$, which is represented as a tuple $(b, o, e_1, e_2, c)$. Figure 4.3 shows an example, where the move $(B1, O2, E2, E3, 8)$ is applied so that for operation $O2$, all 8 time units assigned to employee $E2$ are shifted to employee $E3$.

CHAPTER 5

# An Interactive Optimization Method for the Employee Task Distribution Problem

In this chapter, we introduce an interactive variant of the Employee Task Distribution Problem. First, we motivate and describe the approach used to involve the decision-makers in the optimization process. Then, we adapt the problem definition to be able to model interactive optimization scenarios. Finally, we demonstrate the effectiveness of our approach in practical use cases.

## 5.1 Interactive Optimization

Approaches that solve the ETDP as presented in Chapter 4 find very good solutions to real-life problem instances [HLM$^{+}$24b]. In practice, however, there are cases where users of automated optimization procedures prefer to have control and want to know why specific decisions are made. This is particularly relevant if existing assignments have to be changed due to short-term events like absences caused by illnesses or short-term changes in demands. This motivates the use of methods that involve the user in the optimization process. A comprehensive review of interactive optimization methods can be found in [MKF$^{+}$15]. Two properties are particularly important for interactive optimization: responsiveness and stability [HGQ$^{+}$12]. Responsiveness refers to the time it takes to present a new solution to the user. Due to frequent executions of the solving procedure, this is especially important for interactive decision support systems. The notion of stability requires solutions to be similar to previous solutions so that decision-makers can retrace proposed changes.

With that in mind, we designed an interactive decision support system for the ETDP using the following approach:

1. For an available solution to a problem instance, which is either created by the user manually or with the help of an automated solution approach, the system should propose changes that improve the solution.

2. These changes should be as minimal as possible, such that the decision-maker is able to understand the effect of the suggested changes

3. Then, the user decides whether the suggested change should be applied.

4. If yes, the change is applied to the current solution, and the next improvement is proposed.

5. Otherwise, the change is discarded and will not be suggested again, and another change is proposed to the user.

## 5.2 Problem Statement

In this section, we formally define the interactive variant of the ETDP. To achieve that, we adapt the problem definition from Chapter 4, so that it includes the requirements for the user interaction.

### 5.2.1 Problem Input

The problem input has to be extended in two ways.

First, the initial solution for which improvements are to be found is given as part of the input. The initial solution is represented by an assignment function $A_0(b, o, e) \in \mathbb{N}$.

Second, a method is needed to model the situation in which a user declines a suggested change. In this case, it is required that the same (or similar changes) are not presented to the user again. For example, suppose the system suggests that for an employee, 4 time units should be moved from operation $O1$ to operation $O2$, and the user declines this change. In that case, it is expected that the following suggestions do not include the move of 3 time units from $O1$ to $O2$ for the same employee. Therefore, the input is extended by a function $F(b, o, e) \in \{0, 1\}$ indicating whether the assignment to the triple $(b, o, e)$ can be changed during the solution process. If a user declines a change including the assignment $(b, o, e)$, the respective value $F(b, o, e)$ is set to 1 causing this particular assignment to be fixed, i.e. $A(b, o, e) = A_0(b, o, e)$.

### 5.2.2 Soft Constraints

In the high-level procedure presented in the previous section, it can happen that the initial solution given by the user has constraint violations, i.e., it is not feasible according to the problem definition from Chapter 4.2. This can happen because the user is unaware that constraints are violated or fixing them by hand would be too challenging due to the complexity of the problem. In this case, it is the task of the interactive optimization tool to suggest changes to satisfy the violated constraints. This means our solution method has to be capable of handling constraint violations.

Such violations can occur for all of the constraints from Section 4.2.3, except for the constraints ensuring that the capacities supplied by employees are not exceeded. This has two reasons: First, the application in which the optimization method is used by our industrial partner does not allow the assignment of more capacity to an employee than supplied. Second, it enables an upper bound to the total sum of assigned capacities, which is relevant for calculating theoretical maxima for the objectives.

This means we have to reformalize the problem so that the constraints are modeled as soft constraints, in the same way it was done for two constraints for the local search approach in Chapter 4.3.1. Therefore, we add a penalty term to the objective function also for these constraints.

- If an employee is assigned to an operation he/she is not qualified for, the penalty is equal to the number of time units assigned to this operation-employee pair.

$$\text{penalty}_q = \sum_{b \in \mathcal{B}} \sum_{o \in \mathcal{O}} \sum_{e \in \mathcal{E}} \begin{cases} \text{A(b,o,e)}, & \text{if } A(b,o,e) > 0 \land Q(o,e) = 0 \\ 0, & \text{otherwise} \end{cases}$$

- If, in a time bucket, more capacity is assigned to an operation than demanded, the difference between assigned and demanded capacity is added as a penalty:

$$\text{penalty}_{dc} = \sum_{b \in \mathcal{B}} \sum_{o \in \mathcal{O}} \max\left(\sum_{e \in \mathcal{E}} A(b,o,e) - dc(b,o), 0\right)$$

### 5.2.3 Objectives

- **Maximize sum of assigned capacities**: After the changes from the last section, the demands of operations can now be overfulfilled. Therefore, the basic goal is adapted so that only the assigned capacities up to the demands are counted when calculating the sum of assigned capacities:

$$f_1^{\mathcal{I}} = \sum_{b \in \mathcal{B}} \sum_{o \in \mathcal{O}} \min\left(\sum_{e \in \mathcal{E}} A(b,o,e), dc(b,o)\right)$$

- **Operation priorities:** Similar, for the operation priority objective, the counted capacity must not exceed the demand given for each bucket-operation pair:

$$f_3^{\mathcal{I}} = \sum_{b \in \mathcal{B}} \sum_{p=1}^{p_O} \text{bigM}^o(b,p) \left(\sum_{o \in \mathcal{O}} [op(b,o) = p] \cdot \min\left(\sum_{e \in \mathcal{E}} A(b,o,e), dc(b,o)\right)\right)$$

- **Maximize qualification scores**: In the same way, the qualification score objective is adapted:

$$f_4^{\mathcal{I}} = \sum_{b \in \mathcal{B}} \sum_{o \in \mathcal{O}} \min\left(\sum_{e \in \mathcal{E}} A(b,o,e), dc(b,o)\right) \cdot Q(o,e)$$

- The remaining objectives are unchanged:

$$f_2^{\mathcal{I}} = f_2$$

$$f_5^{\mathcal{I}} = f_5$$

$$f_6^{\mathcal{I}} = f_6$$

The intermediate objective value, which only considers the single objectives without soft constraints, is then calculated in the same way it was done for the objective function $f$ in Chapter 4.2.4:

$$f^{\mathcal{I}} = w_1 \cdot (max_1 - f_1^{\mathcal{I}}) + w_2 \cdot (max_2 - f_2^{\mathcal{I}}) + w_3 \cdot (max_3 - f_3^{\mathcal{I}}) + w_4 \cdot (max_4 - f_4^{\mathcal{I}}) + w_5 \cdot f_5^{\mathcal{I}} + w_6 \cdot f_6^{\mathcal{I}}$$

Adding the penalty terms from the last section together with the big-M constant from Chapter 4.3.1 yields the following function $f_c^{\mathcal{I}}$ to calculate the quality of a solution:

$$f_c^{\mathcal{I}} = f^{\mathcal{I}} + \text{bigM} \cdot (\text{penalty}_q + \text{penalty}_{sc} + \text{penalty}_{dc} + \text{penalty}_{mc} + \text{penalty}_{mp})$$

As mentioned above, we have to ensure that the solutions are similar to the initial solution so that the user can retrace the suggested improvements. To achieve that, we add another objective component, which calculates the distance between the solution and the given initial solution:

- **Minimize distance between solutions (stability objective)**: The distance between two solutions is the number of assignments that differ between the solutions.

$$f_d = \sum_{b \in \mathcal{B}} \sum_{o \in \mathcal{O}} \sum_{e \in \mathcal{E}} [|A(b,o,e) - A_0(b,o,e)| > 0]$$

The final objective function of a solution to the interactive problem variant is the quality $f_c^{\mathcal{I}}$ of the solution, together with the distance to the initial solution $f_d$ weighted by $w_d$.

$$f_*^{\mathcal{I}} = f_c^{\mathcal{I}} + w_d \cdot f_d$$

### 5.2.4 Hard Constraints

Depending on the choices of the weights, it can happen that the solution quality of the solution $A$ is worse or equal to the quality of the initial assignments $A_0$. In other words, the solution suggested to the user would not be an improvement. Furthermore, we have to ensure that fixed assignments are not overwritten. Therefore, we add the following two hard constraints:

- The quality $f_c(A)$ of the solution has to be better (i.e., lower) than the quality of the initial solution:
$$f_c(A) < f_c(A_0)$$
- If an assignment is fixed, it should be the same as given in the initial solution:
$$A(b,o,e) = A_0(b,o,e), \forall b \in \mathcal{B}, \forall o \in \mathcal{O}, \forall e \in \mathcal{E}, \text{ with } F(b,o,e) = 1$$

### 5.2.5 Quality vs. stability

One problem that arises in practice is the choice of objective weights. For the interactive problem variant, the choice of the weight $w_d$ is of particular importance. On the one hand, the proposed improved solution should be as similar to the initial solution as possible. On the other hand, the quality $f_c^{\mathcal{I}}$ of the solution, i.e., the objective value without the distance objective, is of high significance. We propose two ways to choose these weights.

**Users manually choose weight $\mathbf{w_d}$**
The most basic approach is to let the decision-maker determine the importance of stable solutions. Although this provides the most flexibility, it is difficult to find a tradeoff between solution quality and stability.

**Prefer stability over quality**
To overcome the drawback of having to choose the weight of the stability objective manually, we propose another way. The idea is to always prefer stability over quality. In other words, we choose the weight $w_d$, so that for solutions $A_1$ and $A_2$, where $A_1$ is more stable than $A_2$, also the total objective is better for the first solution, i.e., $f_d(A_1) < f_d(A_2)$ implies $f(A_1) < f(A_2)$. To achieve that, we use a big-M constant as weight $w_d$, as we did for the penalty terms. For that, we need an upper bound for the solution quality $f_c^{\mathcal{I}}$. Upper bounds for the objectives $f_1 - f_6$ were already presented in Chapter 4.2 and they still apply for the adapted objectives $f_1^{\mathcal{I}} - f_6^{\mathcal{I}}$. The same has to be done for the penalty terms:

- **Upper bound for penalty$\mathbf{_q}$**: In the worst case, the entire capacity is assigned to pairs of operations and employees without qualification. This means the penalty is equal to the maximum total capacity that can be assigned. And, for each time bucket, this sum cannot exceed the total capacity supplied by each employee.
$$\max_q = \sum_{b \in \mathcal{B}} \sum_{e \in \mathcal{E}} sc(b,e)$$

- **Upper bound for penalty$_{\mathbf{mc}}$**: In the worst case, each operation with a positive minimum capacity does not reach this minimum.

$$\max_{mc} = \sum_{b \in \mathcal{B}} \sum_{o \in \mathcal{O}} \begin{cases} 1, & mc(b,o) > 0 \\ 0, & \text{otherwise} \end{cases}$$

- **Upper bound for penalty$_{\mathbf{mp}}$**: In each time bucket, the maximum total number of operations an employee $e$ can be assigned to in parallel is $|\mathcal{O}|$ (i.e., the employee is assigned to all operations). Therefore, for each bucket-operation-employee triple $(b,o,e)$, the maximum number of parallel operations $mp(b,o)$ can be exceeded by at most $|\mathcal{O}| - mp(b,o)$.

$$\max_{mp} = \sum_{b \in \mathcal{B}} \sum_{o \in \mathcal{O}} \sum_{e \in \mathcal{E}} \begin{cases} \max(|\mathcal{O}| - mp(b,o), 0), & \text{if } mp(b,o) > 0 \\ 0, & \text{otherwise} \end{cases}$$

- **Upper bound for penalty$_{\mathbf{dc}}$**: The total overfulfillment of capacities cannot be larger than the theoretical maximum sum of assigned capacities:

$$\max_{dc} = \sum_{b \in \mathcal{B}} \sum_{e \in \mathcal{E}} sc(b,e)$$

## 5.3 Tabu Search

To solve instances of the interactive ETDP, we apply the Reactive Tabu Search approach presented in Chapter 3.3. In this section, we propose three different types of tabu lists for the ETDP.

**Fix Assignment Tabu List** The idea of this type of tabu list is that after a move is applied, the assignments to all affected bucket-operation-employee triples are fixed. In other words, whenever a move is applied that changes the assignment of a triple $(b,o,e)$, all moves are tabu, which would change the same assignment $A(b,o,e)$.

**Fix Employee Tabu List** This type of tabu list considers the time bucket $b$ and employee $e$ affected by the applied move. All moves that would change an assignment $A(b,o,e)$, for arbitrary $o \in \mathcal{O}$, are considered tabu in the succeeding iterations.

**Fix Operation Tabu List** What the previous type of tabu list did for employees can also be applied to operations. Whenever a move changes an assignment for operation $o$ in bucket $b$, all moves that would change assignments affecting the same bucket-operation pair $(b,o)$ are added to the tabu list.

## 5.4 Use Cases

In this section, we demonstrate the effectiveness of the interactive optimization approach in three use cases, which were acquired in cooperation with our industrial partner. These use cases are based on the instances from industry provided by [HLM$^+$24b, HLM$^+$24a].

| | Capacity | Max. operations | Min. capacity |
|---|---|---|---|
| Operation 1 | 8.00 h | - | - |
| Operation 2 | 2.50 h | - | - |
| Operation 3 | 3.00 h | 2 | - |
| Operation 4 | 1.00 h | - | - |
| Operation 5 | 8.00 h | - | - |
| Operation 6 | 6.50 h | 1 | - |
| Operation 7 | 3.00 h | - | - |
| Operation 8 | 5.75 h | - | - |
| Operation 9 | 4.00 h | - | 4.00 h |
| Operation 10 | 6.50 h | 3 | - |
| Operation 11 | 4.25 h | - | - |
| Operation 12 | 8.00 h | 2 | - |

Table 5.1: The operations used to demonstrate the use cases

For all of these use cases, we have a total of 11 employees and 12 operations. For ease of presentation, we only consider planning one shift at one day, i.e., the instance contains only a single time bucket. However, the insights gathered from the use cases can be generalized to a larger number of buckets. Table 5.1 shows the operations considered in the use cases, together with their demanded capacities, maximum total number of parallel operations, and minimum assigned capacity. The used employees can be seen in Table 5.2, where the supplied capacity is listed for each employee. Note that the capacities (demands and supplies) are displayed in hours and include fractionals, while capacities are integral according to the problem definition. This problem is solved by multiplying each capacity by 4 to retrieve an integer so that a precision from up to a quarter of an hour can be modeled. The qualification matrix is displayed in Table 5.3, where the qualification scores are listed - the higher the score, the better qualified is the employee on the respective operation. Each run of the Reactive Tabu Search algorithm was executed with a time limit of 5 seconds, using manually tuned algorithm parameters. As optimization objectives, we considered the maximization of the sum of assignments ($f_1^{\mathcal{I}}$), maximization of the qualification score ($f_4^{\mathcal{I}}$), and minimization of the assignment count ($f_6^{\mathcal{I}}$). We chose the corresponding weights $w_1, w_4, w_6$, so that $f_1^{\mathcal{I}}$ is lexicographically more important than $f_4^{\mathcal{I}}$, which itself is more important than $f_6^{\mathcal{I}}$. The weight $w_d$ for the distance objective was chosen so that stability is strictly preferred over quality, as described in Section 5.2.5.

### 5.4.1 Use Case 1: Changing employee supplies

In practice, the employee planning is completed a few days in advance. This means the decision-maker either manually assigns the employees to operations or uses the automated planning tool based on the algorithmic approach described in Chapter 4. It often happens, however, that employee supplies change in the short term, caused by absences, for example, due to illnesses.

| | Capacity |
|---|---|
| Turing, Alan | 8.00 h |
| Lovelace, Ada | 8.00 h |
| Gödel, Kurt | 8.00 h |
| Babbage, Charles | 8.00 h |
| Dijkstra, Edsger | 8.00 h |
| von Neumann, John | 8.00 h |
| Johnson, Katherine | 8.00 h |
| Engelbart, Douglas | 8.00 h |
| Hopper, Grace | 8.00 h |
| Hilbert, David | 2.00 h |
| Zuse, Konrad | 2.00 h |

Table 5.2: The employees used to demonstrate the use cases

| | Operations | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Turing, Alan | 30 | - | - | - | - | - | - | - | - | - | - | 30 |
| Lovelace, Ada | - | - | - | 20 | - | - | - | - | - | - | 15 | - |
| Gödel, Kurt | - | - | - | - | 20 | - | 40 | 20 | 20 | - | - | 30 |
| Babbage, Charles | 30 | - | - | - | 20 | - | - | - | - | - | - | - |
| Dijkstra, Edsger | - | - | - | - | 30 | - | - | - | - | - | - | - |
| von Neumann, John | - | - | - | - | - | - | - | - | - | 30 | 40 | - |
| Johnson, Katherine | - | - | 30 | - | - | - | - | 20 | 40 | 20 | - | - |
| Engelbart, Douglas | - | - | - | - | 20 | 40 | - | - | - | - | - | - |
| Hopper, Grace | - | - | - | - | - | - | - | - | - | - | 20 | - |
| Hilbert, David | - | 15 | - | - | - | - | - | - | - | - | - | - |
| Zuse, Konrad | - | 30 | - | - | - | - | - | - | - | - | - | 20 |

Table 5.3: The qualification scores used for the demonstration of the use cases

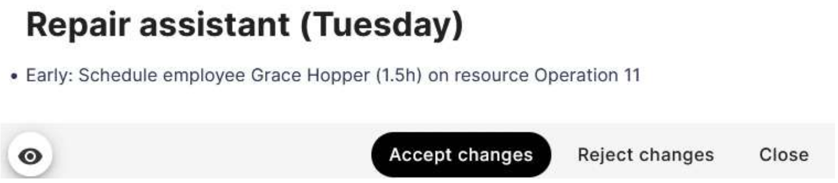
As a concrete use case, assume that the employees *Turing, Alan* and *von Neumann, John* are now absent. This would mean that the demands of the operations they are currently assigned to are no longer fulfilled. Concretely, this means that operations 10 and 12 now do not have any capacities assigned, and only 2.75 h of the demanded 4.25 h are fulfilled. The task of the interactive optimization tool is now to suggest changes to the employee plan so that a large number of demands can be fulfilled by other employees.

The Reactive Tabu Search algorithm is applied to this instance of the interactive problem variant. As result, it returns a solution where the capacity of employee *Hopper, Grace* is increased to 1.5 h on operation 11. Figure 5.1 shows how this change is presented to the user in the real-life application of our industrial partner MCP GmbH [MCP24]. The idea of the change is to substitute the capacity previously provided by the absent employee *von Neumann, John* with the (less qualified) employee *Hopper, Grace*. After accepting this

| | |
|---|---|
| Operation 1 | Babbage, Charles (8 h) |
| Operation 2 | Hilbert, David (0.5 h)<br>Zuse, Konrad (2 h) |
| Operation 3 | Johnson, Katherine (3 h) |
| Operation 4 | Lovelace, Ada (1 h) |
| Operation 5 | Dijkstra, Edsger (8 h) |
| Operation 6 | Engelbart, Douglas (6.5 h) |
| Operation 7 | Gödel, Kurt (3 h) |
| Operation 8 | Gödel, Kurt (0.75 h)<br>Johnson, Katherine (5 h) |
| Operation 9 | Gödel, Kurt (4 h) |
| Operation 10 | von Neumann, John (6.5 h) |
| Operation 11 | von Neumann, John (1.5 h)<br>Hopper, Grace (2.75 h) |
| Operation 12 | Turing, Alan (8 h) |

Table 5.4: The initial solution to the instance of the use cases

change, the improvement is applied to the current solution, and the interactive algorithm is called again, with the adapted solution as the initial solution. The screenshots in Figure 5.2 show the visualization of the solution in the tool of our industrial partner before and after the move is applied. Next, it is suggested to increase the capacity of *Hilbert, David* on operation 2 by 1.5 h and move *Zuse, Konrad* to operation 12 for 1.5 h. This way, the absent employee *Turing, Alan* is partially substituted by *Zuse, Konrad*, who in turn is substituted by *Hilbert, David* on operation 2. Again, we accept these changes and receive the suggestion to move employee *Johnson, Katherine* from operation 8 to operation 10 so that the remaining capacity of 0.25 h for *Gödel, Kurt* can be added to operation 8. The adapted solution, after accepting the suggested improvements, can be seen in Table 5.5.



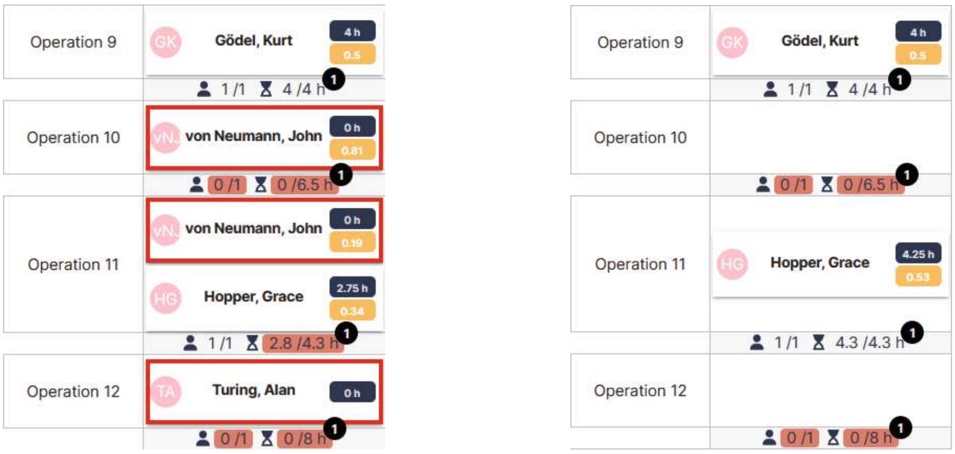
Repair assistant (Tuesday)

- Early: Schedule employee Grace Hopper (1.5h) on resource Operation 11

Figure 5.1: A screenshot from the tool where the interactive approach is used

### 5.4.2 Use Case 2: Modified demands

Like employee supplies, short-term changes in capacity demands for operations are a recurring event in practice. If such changes happen, the decision-maker is responsible for



(a) The plan before the improvement is applied. (b) The plan after the improvement is applied.

Figure 5.2: A screenshot from the tool of our industrial partner showing a part of the solution before and after the first improvement from use case 1 was applied. The absence of an employee is indicated by the red borders.

| | |
|---|---|
| Operation 1 | Babbage, Charles (8 h) |
| Operation 2 | Hilbert, David (0.5 h + 1.5 h)<br>Zuse, Konrad (2 h - 1.5 h ) |
| Operation 3 | Johnson, Katherine (3 h) |
| Operation 4 | Lovelace, Ada (1 h) |
| Operation 5 | Dijkstra, Edsger (8 h) |
| Operation 6 | Engelbart, Douglas (6.5 h) |
| Operation 7 | Gödel, Kurt (3 h) |
| Operation 8 | Gödel, Kurt (0.75 h + 0.25 h)<br>~~Johnson, Katherine (5 h)~~ |
| Operation 9 | Gödel, Kurt (4 h) |
| Operation 10 | ~~von Neumann, John (6.5 h)~~<br>Johnson, Katherine (5 h) |
| Operation 11 | ~~von Neumann, John (1.5 h)~~<br>Hopper, Grace (2.75 h + 1.5 h ) |
| Operation 12 | ~~Turing, Alan (8 h)~~<br>Zuse, Konrad (1.5 h) |

Table 5.5: The solution after the improvements suggested for use case 1 were applied

| | |
|---|---|
| Operation 1 | Babbage, Charles (8 h) |
| Operation 2 | Hilbert, David (0.5 h)<br>Zuse, Konrad (2 h) |
| Operation 3 | Johnson, Katherine (3 h) |
| Operation 4 | Lovelace, Ada (1 h) |
| Operation 5 | Dijkstra, Edsger (8 h)<br>Engelbart, Douglas (8 h) |
| Operation 6 | ~~Engelbart, Douglas (6.5 h)~~ |
| Operation 7 | Gödel, Kurt (3 h) |
| Operation 8 | Gödel, Kurt (0.75 h)<br>Johnson, Katherine (5 h) |
| Operation 9 | Gödel, Kurt (4 h) |
| Operation 10 | von Neumann, John (6.5 h) |
| Operation 11 | von Neumann, John (1.5 h)<br>Hopper, Grace (2.75 h) |
| Operation 12 | Turing, Alan (8 h) |

Table 5.6: The solution after the accepted improvement suggested for use case 2 was applied

incorporating these changes into the existing plan.

Again, consider the solution from Table 5.4, and assume that the requested demand for operation 5 increases to 16 h. After applying the interactive algorithm, the first suggestion is to assign the free capacities from *Gödel, Kurt* (0.25 h) to operation 5. Suppose now the user decides to reject this change. In that case, the system proposes to assign 8 h of the employee *Engelbart, Douglas* to operation 5, where 6.5 h are moved from operation 6, while 1.5 h were still unassigned for this employee. Note that only adding the free capacity of 1.5 h would not be an improvement because the maximum number of parallel operations for all employees assigned to operation 6 is set to 1. Therefore, *Engelbart, Douglas* has to be moved from operation 6 to avoid a violation of this constraint. The resulting solution can be seen in Table 5.6.

### 5.4.3 Use Case 3: Changes to default plan

In practice, the decision-makers often have a default plan as a starting point for their planning. This should keep the plan stable for the employees from week to week. However, as demands and employee availabilities change, these default plans must be adapted to the current demands and employee supplies.

To demonstrate the usage of the interactive planning tool for this use case, let the solution depicted in Table 5.4 be the default plan. The assumed demands and supplies, however, changed in the following ways:

- The demand for operation 8 deduces to 2 h.
- The demand for operation 10 increases to 8 h.
- The demand for operation 11 reduces to 3 h.
- Employee *Gödel, Kurt* is on vacation.

When applying the interactive optimization algorithm, the following changes are suggested, which - as we assume - are all accepted:

1. Shift 1.25 h of employee *von Neumann, John* from operation 11 to operation 10.
2. For employee *Johnson, Katherine*, shift 4 h from operation 8 to operation 9 and remove the remaining 1 h from operation 8
3. Shift 0.25 h of *von Neumann, John* from operation 11 to operation 10 and add 0.25 h to *Hopper, Grace* on operation 10

With the first improvement, the excess capacity from operation 11 is shifted to operation 10, which has unfulfilled demands. Similarly, the second change is to take the unneeded capacity from operation 8 to satisfy the demands of operation 9. Note that keeping the remaining 1 h at operation 8 is not possible because *Johnson, Katherine* is also assigned to operation 3, which has a maximum number of parallel operations of 2. Finally, the remaining demands for operation 10 are fulfilled by applying the third suggestion. As *Gödel, Kurt* is absent and is the only employee qualified for operation 7, these demands cannot be fulfilled. Therefore, no further changes are suggested. The solution after applying all suggested changes can be seen in Table 5.7.

| | |
|---|---|
| Operation 1 | Babbage, Charles (8 h) |
| Operation 2 | Hilbert, David (0.5 h)<br>Zuse, Konrad (2 h) |
| Operation 3 | Johnson, Katherine (3 h) |
| Operation 4 | Lovelace, Ada (1 h) |
| Operation 5 | Dijkstra, Edsger (8 h) |
| Operation 6 | Engelbart, Douglas (6.5 h) |
| Operation 7 | ~~Gödel, Kurt (3 h)~~ |
| Operation 8 | ~~Gödel, Kurt (0.75 h)~~<br>~~Johnson, Katherine (5 h)~~ |
| Operation 9 | ~~Gödel, Kurt (4 h)~~<br>Johnson, Katherine (4 h) |
| Operation 10 | von Neumann, John (6.5 h + 1.5 h) |
| Operation 11 | ~~von Neumann, John (1.5 h)~~<br>Hopper, Grace (2.75 h + 0.25 h ) |
| Operation 12 | Turing, Alan (8 h) |

Table 5.7: The solution after the suggestions from use case 3 were adapted

# CHAPTER 6

# Conclusion

In this thesis, we introduced the Balanced Task Planning Problem. We analyzed its computational complexity and proved that a decision variant is NP-complete by providing a reduction from the well-known NP-complete Bin Packing Problem. We introduced a mathematical model for the BTPP, which we used as an exact approach together with state-of-the-art Constraint Programming and Mixed Integer Programming solvers. Furthermore, we proposed a greedy construction heuristic and different variants of local search-based metaheuristics, which can be used to efficiently solve large-scale problem instances in a reasonable time. More precisely, we used Reactive Tabu Search, an extension of standard Tabu Search that adapts the tabu list length during the search process and introduces a more radical strategy to escape from local optima. Additionally, we used two strategies to reduce the size of the neighborhood explored in each iteration. To evaluate our approaches, we introduced a set of problem instances, which we created using a random instance generator developed in this thesis. The experimental results show that, among the investigated methods, the exact approach produces the best results for small problem instances and solves 24% of the instances to optimality. Using the different Tabu Search variants, we could find high-quality solutions for most medium-sized and large-scale problem instances. The experiments indicate that for the evaluated test set, on average, the Reactive Tabu Search is capable of finding better solutions than the standard variant of Tabu Search. Furthermore, restricting the neighborhood led to improved results in our experiments, especially for the larger instances.

In the second part of the thesis, we investigated interactive optimization methods for the Employee Task Distribution Problem. We proposed a method that suggests stepwise improvements so that the decision-maker can follow and possibly adapt the suggestions. We adapted the formal problem definition from the literature to model the special requirements for user interaction. To solve the interactive problem variant, we proposed a Tabu Search method with three different tabu list implementations. Finally, we demonstrated our approach in three practically relevant use cases. This showed that the proposed method is capable of suggesting reasonable improvements to solutions, which can help decision-makers react to changes in problem instances after an initial (automatic or manual) optimization.

In future work, it would be interesting to examine further solution approaches for the Balanced Task Planning Problem, e.g., using other metaheuristic methods like Simulated Annealing, Genetic Algorithms or Large Neighborhood Search. A more detailed investigation of exact methods could provide optimal solutions for more problem instances. For the Employee Task Distribution Problem, it would be interesting to apply the proposed Reactive Tabu Search approach also for the non-interactive problem variant and compare the results to the methods from the literature.

APPENDIX A

# Experimental results of exact methods

Table A.1: The objective values for each of the used solvers and the best dual bound found by one of the solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed | Dual Bound |
|---|---|---|---|---|---|
| small_1 | 885.0 | 924.0 | - | - | 9.0 |
| small_2 | 236.0* | 236.0* | 236.0* | 390.0 | 236.0 |
| small_3 | 606.0 | 637.0 | - | - | 336.0 |
| small_4 | - | 276.0 | - | - | 0.0 |
| small_5 | 146.0 | 146.0 | 159.0 | 199.0 | 138.12 |
| small_6 | 144.0 | 141.0 | - | 350.0 | 93.0 |
| small_7 | 534.0* | 534.0* | 687.0 | 894.0 | 534.0 |
| small_8 | 890.0* | 890.0* | - | 1466.0 | 890.0 |
| small_9 | 338.0 | 379.0 | - | - | 153.0 |
| small_10 | 959.0 | 943.0 | - | 2242.0 | 278.7 |
| small_11 | 53.0 | 40.0 | - | 135.0 | 16.0 |
| small_12 | 170.0* | 170.0* | 170.0* | - | 170.0 |
| small_13 | 95.0 | 174.0 | - | - | 0.0 |
| small_14 | 457.0* | 457.0* | 457.0* | 457.0* | 457.0 |
| small_15 | 281.0* | 281.0* | 313.0 | 921.0 | 281.0 |
| small_16 | 193.0 | 274.0 | - | - | 74.0 |
| small_17 | 74.0 | 66.0 | 118.0 | 357.0 | 38.0 |
| small_18 | 104.0 | 107.0 | - | 187.0 | 0.0 |
| small_19 | 101.0 | 82.0 | - | 691.0 | 11.0 |
| small_20 | 203.0* | 203.0* | 212.0 | - | 203.0 |
| small_21 | 134.0 | 230.0 | - | - | 0.0 |
| small_22 | 146.0* | 146.0* | 146.0* | 1332.0 | 146.0 |
| small_23 | - | 484.0 | - | 1965.0 | 16.0 |
| small_24 | 2658.0 | 482.0 | - | - | 5.0 |
| small_25 | 610.0 | 583.0 | - | - | 488.23 |
| small_26 | 536.0 | 547.0 | - | 1003.0 | 422.0 |
| small_27 | 563.0 | 495.0 | - | 1449.0 | 369.0 |
| small_28 | - | 685.0 | - | - | 1.0 |

Continued on next page

Table A.1: The objective values for each of the used solvers and the best dual bound found by one of the solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed | Dual Bound |
|---|---|---|---|---|---|
| small_29 | 761.0 | 764.0 | 1150.0 | 1036.0 | 663.98 |
| small_30 | 487.0* | 487.0* | 487.0* | 2695.0 | 487.0 |
| small_31 | 124.0* | 124.0* | 270.0 | 276.0 | 124.0 |
| small_32 | 420.0 | 420.0 | - | - | 0.0 |
| small_33 | 1177.0 | 891.0 | - | - | 4.0 |
| small_34 | 311.0* | 311.0* | - | - | 311.0 |
| small_35 | 403.0 | 393.0 | - | - | 215.0 |
| small_36 | 197.0* | 197.0* | - | 956.0 | 197.0 |
| small_37 | 661.0 | 657.0 | - | 950.0 | 190.97 |
| small_38 | 935.0 | 801.0 | - | - | 19.5 |
| small_39 | 226.0 | 226.0 | - | 2056.0 | 0.0 |
| small_40 | 340.0 | 337.0 | - | - | 159.77 |
| small_41 | 742.0 | 739.0 | - | 2025.0 | 286.0 |
| small_42 | 286.0 | 408.0 | - | 3190.0 | 12.0 |
| small_43 | 356.0* | 356.0* | 356.0* | 404.0 | 356.0 |
| small_44 | - | 971.0 | - | - | 199.0 |
| small_45 | 413.0 | 445.0 | - | 5003.0 | 218.0 |
| small_46 | 1162.0 | 1169.0 | - | 4553.0 | 1156.0 |
| small_47 | - | 831.0 | - | 2531.0 | 12.0 |
| small_48 | 336.0* | 336.0* | 408.0 | 920.0 | 336.0 |
| small_49 | 279.0 | 280.0 | 1786.0 | 2729.0 | 63.0 |
| small_50 | - | 201.0 | - | 245.0 | 82.0 |
| small_51 | 1119.0* | 1119.0* | - | 3171.0 | 1119.0 |
| small_52 | 1176.0 | 798.0 | - | 5153.0 | 492.04 |
| small_53 | 158.0 | 158.0 | - | 302.0 | 145.58 |
| small_54 | 320.0 | 398.0 | - | 664.0 | 6.0 |
| small_55 | 184.0* | 186.0 | 239.0 | 281.0 | 184.0 |
| small_56 | 391.0 | 314.0 | - | 2466.0 | 5.0 |
| small_57 | 142.0 | 138.0 | 212.0 | 472.0 | 96.0 |
| small_58 | 408.0 | 466.0 | - | - | 38.0 |
| small_59 | 159.0 | 266.0 | - | 434.0 | 8.0 |
| small_60 | 1075.0 | 1417.0 | - | 1650.0 | 0.0 |
| small_61 | 294.0 | 280.0 | - | - | 182.0 |
| small_62 | 222.0 | 185.0 | - | - | 3.0 |
| small_63 | 501.0 | 468.0 | - | 3828.0 | 8.0 |
| small_64 | - | 225.0 | - | 1059.0 | 151.0 |
| small_65 | 613.0 | 377.0 | - | - | 4.0 |
| small_66 | 133.0 | 138.0 | 1492.0 | 782.0 | 81.0 |
| small_67 | 472.0 | 470.0 | 490.0 | 553.0 | 460.0 |
| small_68 | 224.0 | 284.0 | - | - | 9.0 |
| small_69 | 319.0* | 319.0* | 319.0* | 319.0* | 319.0 |
| small_70 | - | 387.0 | - | 1024.0 | 211.0 |
| small_71 | 285.0 | 321.0 | - | - | 95.0 |
| small_72 | 3653.0 | 774.0 | - | - | 2.0 |
| small_73 | 664.0 | 608.0 | - | - | 5.0 |
| small_74 | - | 995.0 | - | - | 5.0 |
| small_75 | 242.0 | 242.0 | 285.0 | 582.0 | 177.16 |
| small_76 | - | 1499.0 | - | - | 2.0 |
| small_77 | - | 4368.0 | - | - | 0.0 |
| small_78 | 955.0 | 1035.0 | - | - | 220.67 |
| small_79 | 142.0 | 175.0 | - | 859.0 | 129.0 |
| small_80 | 259.0 | 260.0 | - | 485.0 | 101.0 |
| small_81 | 202.0* | 203.0 | 381.0 | 280.0 | 202.0 |
| small_82 | 366.0 | 339.0 | - | 1381.0 | 89.62 |

Continued on next page

Table A.1: The objective values for each of the used solvers and the best dual bound found by one of the solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed | Dual Bound |
|---|---|---|---|---|---|
| small_83 | 517.0 | 510.0 | 1789.0 | 1771.0 | 354.45 |
| small_84 | 395.0 | 401.0 | - | 1040.0 | 330.0 |
| small_85 | 491.0 | 486.0 | - | 1770.0 | 349.7 |
| small_86 | 90.0 | 90.0 | - | - | 9.01 |
| small_87 | - | 1469.0 | - | - | 12.0 |
| small_88 | 339.0* | 339.0* | 339.0* | 339.0* | 339.0 |
| small_89 | 180.0 | 178.0 | - | 431.0 | 171.0 |
| small_90 | 1122.0 | 1293.0 | - | - | 3.0 |
| small_91 | 2353.0 | 2418.0 | - | - | 26.5 |
| small_92 | 894.0* | 894.0* | 894.0* | 2382.0 | 894.0 |
| small_93 | 87.0 | 79.0 | - | 741.0 | 26.0 |
| small_94 | 198.0* | 207.0 | - | - | 198.0 |
| small_95 | - | 649.0 | - | - | 3.0 |
| small_96 | 386.0* | 386.0* | - | 472.0 | 386.0 |
| small_97 | 86.0* | 86.0 | - | 822.0 | 86.0 |
| small_98 | 71.0* | 71.0 | 80.0 | 104.0 | 71.0 |
| small_99 | 131.0 | 128.0 | 252.0 | 326.0 | 74.0 |
| small_100 | - | - | - | - | 0 |
| medium_1 | - | 1037.0 | - | - | 0.0 |
| medium_2 | - | - | - | - | 0 |
| medium_3 | - | - | - | - | 0 |
| medium_4 | - | - | - | - | 0 |
| medium_5 | - | 2195.0 | - | - | 195.27 |
| medium_6 | - | - | - | - | 0 |
| medium_7 | - | 5446.0 | - | - | 4.0 |
| medium_8 | - | 1186.0 | - | 1822.0 | 4.0 |
| medium_9 | 1528.0 | 1088.0 | - | - | 0.0 |
| medium_10 | - | 7270.0 | - | - | 0 |
| medium_11 | - | - | - | - | 0 |
| medium_12 | - | - | - | - | 0 |
| medium_13 | - | - | - | - | 0 |
| medium_14 | - | - | - | - | 0 |
| medium_15 | - | 1569.0 | - | - | 0 |
| medium_16 | - | - | - | - | 0 |
| medium_17 | - | - | - | - | 54.0 |
| medium_18 | - | 378.0 | - | - | 0.0 |
| medium_19 | - | 501.0 | - | - | 0.0 |
| medium_20 | - | - | - | - | 0 |
| medium_21 | - | 1706.0 | - | - | 3.0 |
| medium_22 | - | - | - | - | 0 |
| medium_23 | - | - | - | - | 0.0 |
| medium_24 | - | 5000.0 | - | - | 0.0 |
| medium_25 | 1229.0 | 1235.0 | - | - | 2.0 |
| medium_26 | - | - | - | - | 28.0 |
| medium_27 | - | 1067.0 | - | - | 1.0 |
| medium_28 | - | - | - | - | 0 |
| medium_29 | - | 7704.0 | - | - | 0 |
| medium_30 | - | 5049.0 | - | - | 0 |
| medium_31 | - | 9176.0 | - | - | 1.0 |
| medium_32 | 11.0* | 11.0* | - | 35.0 | 11.0 |
| medium_33 | - | - | - | - | 0 |
| medium_34 | - | - | - | - | 0.0 |
| medium_35 | - | - | - | - | 0 |
| medium_36 | - | - | - | - | 0.0 |

Continued on next page

Table A.1: The objective values for each of the used solvers and the best dual bound found by one of the solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed | Dual Bound |
|---|---|---|---|---|---|
| medium_37 | - | 3102.0 | - | - | 0 |
| medium_38 | - | 5711.0 | - | - | 0 |
| medium_39 | - | 12885.0 | - | - | 0 |
| medium_40 | - | - | - | - | 0 |
| medium_41 | - | - | - | - | 0 |
| medium_42 | - | 1374.0 | - | - | 1.0 |
| medium_43 | - | - | - | - | 0 |
| medium_44 | - | - | - | - | 0 |
| medium_45 | - | - | - | - | 0 |
| medium_46 | 2008.0 | - | - | - | 0.0 |
| medium_47 | - | - | - | - | 0 |
| medium_48 | - | 4899.0 | - | - | 4.0 |
| medium_49 | - | - | - | - | 0 |
| medium_50 | - | 3826.0 | - | - | 0 |
| medium_51 | - | - | - | - | 0 |
| medium_52 | - | - | - | - | 125.0 |
| medium_53 | - | - | - | - | 0 |
| medium_54 | - | 1219.0 | - | - | 2.0 |
| medium_55 | - | 4402.0 | - | - | 0.0 |
| medium_56 | 315.0 | 245.0 | - | - | 0 |
| medium_57 | - | 11875.0 | - | - | 0 |
| medium_58 | - | 4817.0 | - | - | 1.0 |
| medium_59 | - | - | - | - | 0 |
| medium_60 | - | - | - | - | 0 |
| medium_61 | - | - | - | - | 0.0 |
| medium_62 | - | 13833.0 | - | - | 0 |
| medium_63 | - | 3476.0 | - | - | 2.0 |
| medium_64 | - | - | - | - | 0 |
| medium_65 | - | - | - | - | 0 |
| medium_66 | - | 2257.0 | - | - | 0.0 |
| medium_67 | - | 4376.0 | - | - | 0 |
| medium_68 | - | - | - | - | 0 |
| medium_69 | - | 9831.0 | - | - | 0 |
| medium_70 | - | 749.0 | - | - | 0.0 |
| medium_71 | - | - | - | - | 0 |
| medium_72 | - | 1917.0 | - | - | 5.0 |
| medium_73 | - | 11540.0 | - | - | 0 |
| medium_74 | - | - | - | - | 0 |
| medium_75 | - | - | - | - | 0 |
| medium_76 | - | 521.0 | - | - | 0.0 |
| medium_77 | - | - | - | - | 0.0 |
| medium_78 | - | - | - | - | 0 |
| medium_79 | - | 1051.0 | - | - | 4.0 |
| medium_80 | - | - | - | - | 0 |
| medium_81 | - | - | - | - | 0 |
| medium_82 | - | 2234.0 | - | 5216.0 | 79.0 |
| medium_83 | - | 776.0 | - | - | 0.0 |
| medium_84 | - | - | - | - | 0 |
| medium_85 | - | - | - | - | 0 |
| medium_86 | - | - | - | - | 0 |
| medium_87 | - | - | - | - | 0.0 |
| medium_88 | 365.0 | 343.0 | - | - | 295.0 |
| medium_89 | - | - | - | - | 0 |
| medium_90 | - | - | - | - | 0 |

Continued on next page

Table A.1: The objective values for each of the used solvers and the best dual bound found by one of the solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed | Dual Bound |
|---|---|---|---|---|---|
| medium_91 | - | 5679.0 | - | - | 1.0 |
| medium_92 | - | - | - | - | 0.0 |
| medium_93 | 932.0 | 1188.0 | - | 1599.0 | 180.0 |
| medium_94 | - | - | - | - | 0 |
| medium_95 | - | 1128.0 | - | - | 601.11 |
| medium_96 | - | - | - | - | 0.0 |
| medium_97 | - | 4620.0 | - | - | 0 |
| medium_98 | - | - | - | - | 0 |
| medium_99 | - | 2789.0 | - | - | 6.0 |
| medium_100 | - | 1828.0 | - | - | 0 |
| large_1 | - | - | - | - | 0 |
| large_2 | - | - | - | - | 0 |
| large_3 | - | - | - | - | 0 |
| large_4 | - | - | - | - | 0 |
| large_5 | - | - | - | - | 0 |
| large_6 | - | 12269.0 | - | - | 0 |
| large_7 | - | 14368.0 | - | - | 0 |
| large_8 | - | - | - | - | 0 |
| large_9 | - | - | - | - | 0 |
| large_10 | - | - | - | - | 0 |
| large_11 | - | - | - | - | 0 |
| large_12 | - | - | - | - | 0 |
| large_13 | - | - | - | - | 0 |
| large_14 | - | 6281.0 | - | - | 0 |
| large_15 | - | - | - | - | 0 |
| large_16 | - | - | - | - | 0 |
| large_17 | - | - | - | - | 0 |
| large_18 | - | - | - | - | 0 |
| large_19 | - | - | - | - | 0 |
| large_20 | - | 8378.0 | - | - | 0 |
| large_21 | - | - | - | - | 0 |
| large_22 | - | - | - | - | 0 |
| large_23 | - | - | - | - | 0 |
| large_24 | - | - | - | - | 0 |
| large_25 | - | - | - | - | 0 |
| large_26 | - | - | - | - | 0 |
| large_27 | - | - | - | - | 0 |
| large_28 | - | - | - | - | 0 |
| large_29 | - | 9347.0 | - | - | 0 |
| large_30 | - | - | - | - | 0 |
| large_31 | - | - | - | - | 0 |
| large_32 | - | - | - | - | 0 |
| large_33 | - | - | - | - | 0.0 |
| large_34 | - | - | - | - | 0 |
| large_35 | - | - | - | - | 0 |
| large_36 | - | - | - | - | 0 |
| large_37 | - | - | - | - | 0 |
| large_38 | - | 4441.0 | - | - | 0 |
| large_39 | - | - | - | - | 0 |
| large_40 | - | 4097.0 | - | - | 0 |
| large_41 | - | 6109.0 | - | - | 0 |
| large_42 | - | - | - | - | 0 |
| large_43 | - | 5229.0 | - | - | 0 |
| large_44 | - | - | - | - | 0 |

Continued on next page

Table A.1: The objective values for each of the used solvers and the best dual bound found by one of the solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed | Dual Bound |
|---|---|---|---|---|---|
| large_45 | - | - | - | - | 0 |
| large_46 | - | - | - | - | 0 |
| large_47 | - | - | - | - | 0 |
| large_48 | - | 6852.0 | - | - | 0 |
| large_49 | - | - | - | - | 0 |
| large_50 | - | - | - | - | 0 |
| large_51 | - | - | - | - | 0 |
| large_52 | - | - | - | - | 0 |
| large_53 | - | - | - | - | 0 |
| large_54 | - | - | - | - | 0 |
| large_55 | - | 9847.0 | - | - | 0 |
| large_56 | - | - | - | - | 0 |
| large_57 | - | - | - | - | 0 |
| large_58 | - | - | - | - | 0 |
| large_59 | - | - | - | - | 0 |
| large_60 | - | - | - | - | 0 |
| large_61 | - | - | - | - | 0 |
| large_62 | - | - | - | - | 0 |
| large_63 | - | - | - | - | 0 |
| large_64 | - | - | - | - | 0 |
| large_65 | - | - | - | - | 0 |
| large_66 | - | - | - | - | 0 |
| large_67 | - | - | - | - | 0 |
| large_68 | - | - | - | - | 0 |
| large_69 | - | - | - | - | 700.0 |
| large_70 | - | - | - | - | 0 |
| large_71 | - | - | - | - | 0 |
| large_72 | - | - | - | - | 0 |
| large_73 | - | - | - | - | 0 |
| large_74 | - | - | - | - | 37.0 |
| large_75 | - | - | - | - | 0 |
| large_76 | - | 9778.0 | - | - | 0 |
| large_77 | - | - | - | - | 0 |
| large_78 | - | - | - | - | 0 |
| large_79 | - | - | - | - | 0 |
| large_80 | - | - | - | - | 0 |
| large_81 | - | - | - | - | 0 |
| large_82 | - | - | - | - | 0 |
| large_83 | - | - | - | - | 0 |
| large_84 | - | - | - | - | 0 |
| large_85 | - | - | - | - | 0 |
| large_86 | - | - | - | - | 0 |
| large_87 | - | 19490.0 | - | - | 2.0 |
| large_88 | - | 17447.0 | - | - | 0 |
| large_89 | - | - | - | - | 0 |
| large_90 | - | - | - | - | 0 |
| large_91 | - | - | - | - | 0 |
| large_92 | - | - | - | - | 0 |
| large_93 | - | - | - | - | 0 |
| large_94 | - | - | - | - | 0 |
| large_95 | - | - | - | - | 0 |
| large_96 | - | - | - | - | 0 |
| large_97 | - | - | - | - | 0 |
| large_98 | - | 8695.0 | - | - | 0 |

Continued on next page

Table A.1: The objective values for each of the used solvers and the best dual bound found by one of the solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed | Dual Bound |
|---|---|---|---|---|---|
| large_99 | - | - | - | - | 772.0 |
| large_100 | - | - | - | - | 0 |

Table A.2: The relative objective values for each of the used solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed |
|---|---|---|---|---|
| small_1 | **1.0** | 1.0441 | - | - |
| small_2 | **1.0** | **1.0** | **1.0** | 1.6525 |
| small_3 | **1.0** | 1.0512 | - | - |
| small_4 | - | **1.0** | - | - |
| small_5 | **1.0** | **1.0** | 1.089 | 1.363 |
| small_6 | 1.0213 | **1.0** | - | 2.4823 |
| small_7 | **1.0** | **1.0** | 1.2865 | 1.6742 |
| small_8 | **1.0** | **1.0** | - | 1.6472 |
| small_9 | **1.0** | 1.1213 | - | - |
| small_10 | 1.017 | **1.0** | - | 2.3775 |
| small_11 | 1.325 | **1.0** | - | 3.375 |
| small_12 | **1.0** | **1.0** | **1.0** | - |
| small_13 | **1.0** | 1.8316 | - | - |
| small_14 | **1.0** | **1.0** | **1.0** | **1.0** |
| small_15 | **1.0** | **1.0** | 1.1139 | 3.2776 |
| small_16 | **1.0** | 1.4197 | - | - |
| small_17 | 1.1212 | **1.0** | 1.7879 | 5.4091 |
| small_18 | **1.0** | 1.0288 | - | 1.7981 |
| small_19 | 1.2317 | **1.0** | - | 8.4268 |
| small_20 | **1.0** | **1.0** | 1.0443 | - |
| small_21 | **1.0** | 1.7164 | - | - |
| small_22 | **1.0** | **1.0** | **1.0** | 9.1233 |
| small_23 | - | **1.0** | - | 4.0599 |
| small_24 | 5.5145 | **1.0** | - | - |
| small_25 | 1.0463 | **1.0** | - | - |
| small_26 | **1.0** | 1.0205 | - | 1.8713 |
| small_27 | 1.1374 | **1.0** | - | 2.9273 |
| small_28 | - | **1.0** | - | - |
| small_29 | **1.0** | 1.0039 | 1.5112 | 1.3614 |
| small_30 | **1.0** | **1.0** | **1.0** | 5.5339 |
| small_31 | **1.0** | **1.0** | 2.1774 | 2.2258 |
| small_32 | **1.0** | **1.0** | - | - |
| small_33 | 1.321 | **1.0** | - | - |
| small_34 | **1.0** | **1.0** | - | - |
| small_35 | 1.0254 | **1.0** | - | - |
| small_36 | **1.0** | **1.0** | - | 4.8528 |
| small_37 | 1.0061 | **1.0** | - | 1.446 |
| small_38 | 1.1673 | **1.0** | - | - |
| small_39 | **1.0** | **1.0** | - | 9.0973 |
| small_40 | 1.0089 | **1.0** | - | - |
| small_41 | 1.0041 | **1.0** | - | 2.7402 |
| small_42 | **1.0** | 1.4266 | - | 11.1538 |
| small_43 | **1.0** | **1.0** | **1.0** | 1.1348 |
| small_44 | - | **1.0** | - | - |
| small_45 | **1.0** | 1.0775 | - | 12.1138 |
| small_46 | **1.0** | 1.006 | - | 3.9182 |
| small_47 | - | **1.0** | - | 3.0457 |
| small_48 | **1.0** | **1.0** | 1.2143 | 2.7381 |
| small_49 | **1.0** | 1.0036 | 6.4014 | 9.7814 |
| small_50 | - | **1.0** | - | 1.2189 |
| small_51 | **1.0** | **1.0** | - | 2.8338 |
| small_52 | 1.4737 | **1.0** | - | 6.4574 |
| small_53 | **1.0** | **1.0** | - | 1.9114 |
| small_54 | **1.0** | 1.2438 | - | 2.075 |
| small_55 | **1.0** | 1.0109 | 1.2989 | 1.5272 |

Continued on next page

Table A.2: The relative objective values for each of the used solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed |
|---|---|---|---|---|
| small_56 | 1.2452 | **1.0** | - | 7.8535 |
| small_57 | 1.029 | **1.0** | 1.5362 | 3.4203 |
| small_58 | **1.0** | 1.1422 | - | - |
| small_59 | **1.0** | 1.673 | - | 2.7296 |
| small_60 | **1.0** | 1.3181 | - | 1.5349 |
| small_61 | 1.05 | **1.0** | - | - |
| small_62 | 1.2 | **1.0** | - | - |
| small_63 | 1.0705 | **1.0** | - | 8.1795 |
| small_64 | - | **1.0** | - | 4.7067 |
| small_65 | 1.626 | **1.0** | - | - |
| small_66 | **1.0** | 1.0376 | 11.218 | 5.8797 |
| small_67 | 1.0043 | **1.0** | 1.0426 | 1.1766 |
| small_68 | **1.0** | 1.2679 | - | - |
| small_69 | **1.0** | **1.0** | **1.0** | **1.0** |
| small_70 | - | **1.0** | - | 2.646 |
| small_71 | **1.0** | 1.1263 | - | - |
| small_72 | 4.7196 | **1.0** | - | - |
| small_73 | 1.0921 | **1.0** | - | - |
| small_74 | - | **1.0** | - | - |
| small_75 | **1.0** | **1.0** | 1.1777 | 2.405 |
| small_76 | - | **1.0** | - | - |
| small_77 | - | **1.0** | - | - |
| small_78 | **1.0** | 1.0838 | - | - |
| small_79 | **1.0** | 1.2324 | - | 6.0493 |
| small_80 | **1.0** | 1.0039 | - | 1.8726 |
| small_81 | **1.0** | 1.005 | 1.8861 | 1.3861 |
| small_82 | 1.0796 | **1.0** | - | 4.0737 |
| small_83 | 1.0137 | **1.0** | 3.5078 | 3.4725 |
| small_84 | **1.0** | 1.0152 | - | 2.6329 |
| small_85 | 1.0103 | **1.0** | - | 3.642 |
| small_86 | **1.0** | **1.0** | - | - |
| small_87 | - | **1.0** | - | - |
| small_88 | **1.0** | **1.0** | **1.0** | **1.0** |
| small_89 | 1.0112 | **1.0** | - | 2.4213 |
| small_90 | **1.0** | 1.1524 | - | - |
| small_91 | **1.0** | 1.0276 | - | - |
| small_92 | **1.0** | **1.0** | **1.0** | 2.6644 |
| small_93 | 1.1013 | **1.0** | - | 9.3797 |
| small_94 | **1.0** | 1.0455 | - | - |
| small_95 | - | **1.0** | - | - |
| small_96 | **1.0** | **1.0** | - | 1.2228 |
| small_97 | **1.0** | **1.0** | - | 9.5581 |
| small_98 | **1.0** | **1.0** | 1.1268 | 1.4648 |
| small_99 | 1.0234 | **1.0** | 1.9688 | 2.5469 |
| small_100 | - | - | - | - |
| medium_1 | - | **1.0** | - | - |
| medium_2 | - | - | - | - |
| medium_3 | - | - | - | - |
| medium_4 | - | - | - | - |
| medium_5 | - | **1.0** | - | - |
| medium_6 | - | - | - | - |
| medium_7 | - | **1.0** | - | - |
| medium_8 | - | **1.0** | - | 1.5363 |
| medium_9 | 1.4044 | **1.0** | - | - |
| medium_10 | - | **1.0** | - | - |

Continued on next page

Table A.2: The relative objective values for each of the used solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed |
|---|---|---|---|---|
| medium_11 | - | - | - | - |
| medium_12 | - | - | - | - |
| medium_13 | - | - | - | - |
| medium_14 | - | - | - | - |
| medium_15 | - | **1.0** | - | - |
| medium_16 | - | - | - | - |
| medium_17 | - | - | - | - |
| medium_18 | - | **1.0** | - | - |
| medium_19 | - | **1.0** | - | - |
| medium_20 | - | - | - | - |
| medium_21 | - | **1.0** | - | - |
| medium_22 | - | - | - | - |
| medium_23 | - | - | - | - |
| medium_24 | - | **1.0** | - | - |
| medium_25 | **1.0** | 1.0049 | - | - |
| medium_26 | - | - | - | - |
| medium_27 | - | **1.0** | - | - |
| medium_28 | - | - | - | - |
| medium_29 | - | **1.0** | - | - |
| medium_30 | - | **1.0** | - | - |
| medium_31 | - | **1.0** | - | - |
| medium_32 | **1.0** | **1.0** | - | 3.1818 |
| medium_33 | - | - | - | - |
| medium_34 | - | - | - | - |
| medium_35 | - | - | - | - |
| medium_36 | - | - | - | - |
| medium_37 | - | **1.0** | - | - |
| medium_38 | - | **1.0** | - | - |
| medium_39 | - | **1.0** | - | - |
| medium_40 | - | - | - | - |
| medium_41 | - | - | - | - |
| medium_42 | - | **1.0** | - | - |
| medium_43 | - | - | - | - |
| medium_44 | - | - | - | - |
| medium_45 | - | - | - | - |
| medium_46 | **1.0** | - | - | - |
| medium_47 | - | - | - | - |
| medium_48 | - | **1.0** | - | - |
| medium_49 | - | - | - | - |
| medium_50 | - | **1.0** | - | - |
| medium_51 | - | - | - | - |
| medium_52 | - | - | - | - |
| medium_53 | - | - | - | - |
| medium_54 | - | **1.0** | - | - |
| medium_55 | - | **1.0** | - | - |
| medium_56 | 1.2857 | **1.0** | - | - |
| medium_57 | - | **1.0** | - | - |
| medium_58 | - | **1.0** | - | - |
| medium_59 | - | - | - | - |
| medium_60 | - | - | - | - |
| medium_61 | - | - | - | - |
| medium_62 | - | **1.0** | - | - |
| medium_63 | - | **1.0** | - | - |
| medium_64 | - | - | - | - |
| medium_65 | - | - | - | - |

Continued on next page

Table A.2: The relative objective values for each of the used solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed |
|---|---|---|---|---|
| medium_66 | - | **1.0** | - | - |
| medium_67 | - | **1.0** | - | - |
| medium_68 | - | - | - | - |
| medium_69 | - | **1.0** | - | - |
| medium_70 | - | **1.0** | - | - |
| medium_71 | - | - | - | - |
| medium_72 | - | **1.0** | - | - |
| medium_73 | - | **1.0** | - | - |
| medium_74 | - | - | - | - |
| medium_75 | - | - | - | - |
| medium_76 | - | **1.0** | - | - |
| medium_77 | - | - | - | - |
| medium_78 | - | - | - | - |
| medium_79 | - | **1.0** | - | - |
| medium_80 | - | - | - | - |
| medium_81 | - | - | - | - |
| medium_82 | - | **1.0** | - | 2.3348 |
| medium_83 | - | **1.0** | - | - |
| medium_84 | - | - | - | - |
| medium_85 | - | - | - | - |
| medium_86 | - | - | - | - |
| medium_87 | - | - | - | - |
| medium_88 | 1.0641 | **1.0** | - | - |
| medium_89 | - | - | - | - |
| medium_90 | - | - | - | - |
| medium_91 | - | **1.0** | - | - |
| medium_92 | - | - | - | - |
| medium_93 | **1.0** | 1.2747 | - | 1.7157 |
| medium_94 | - | - | - | - |
| medium_95 | - | **1.0** | - | - |
| medium_96 | - | - | - | - |
| medium_97 | - | **1.0** | - | - |
| medium_98 | - | - | - | - |
| medium_99 | - | **1.0** | - | - |
| medium_100 | - | **1.0** | - | - |
| large_1 | - | - | - | - |
| large_2 | - | - | - | - |
| large_3 | - | - | - | - |
| large_4 | - | - | - | - |
| large_5 | - | - | - | - |
| large_6 | - | **1.0** | - | - |
| large_7 | - | **1.0** | - | - |
| large_8 | - | - | - | - |
| large_9 | - | - | - | - |
| large_10 | - | - | - | - |
| large_11 | - | - | - | - |
| large_12 | - | - | - | - |
| large_13 | - | - | - | - |
| large_14 | - | **1.0** | - | - |
| large_15 | - | - | - | - |
| large_16 | - | - | - | - |
| large_17 | - | - | - | - |
| large_18 | - | - | - | - |
| large_19 | - | - | - | - |
| large_20 | - | **1.0** | - | - |

Continued on next page

Table A.2: The relative objective values for each of the used solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed |
|---|---|---|---|---|
| large_21 | - | - | - | - |
| large_22 | - | - | - | - |
| large_23 | - | - | - | - |
| large_24 | - | - | - | - |
| large_25 | - | - | - | - |
| large_26 | - | - | - | - |
| large_27 | - | - | - | - |
| large_28 | - | - | - | - |
| large_29 | - | **1.0** | - | - |
| large_30 | - | - | - | - |
| large_31 | - | - | - | - |
| large_32 | - | - | - | - |
| large_33 | - | - | - | - |
| large_34 | - | - | - | - |
| large_35 | - | - | - | - |
| large_36 | - | - | - | - |
| large_37 | - | - | - | - |
| large_38 | - | **1.0** | - | - |
| large_39 | - | - | - | - |
| large_40 | - | **1.0** | - | - |
| large_41 | - | **1.0** | - | - |
| large_42 | - | - | - | - |
| large_43 | - | **1.0** | - | - |
| large_44 | - | - | - | - |
| large_45 | - | - | - | - |
| large_46 | - | - | - | - |
| large_47 | - | - | - | - |
| large_48 | - | **1.0** | - | - |
| large_49 | - | - | - | - |
| large_50 | - | - | - | - |
| large_51 | - | - | - | - |
| large_52 | - | - | - | - |
| large_53 | - | - | - | - |
| large_54 | - | - | - | - |
| large_55 | - | **1.0** | - | - |
| large_56 | - | - | - | - |
| large_57 | - | - | - | - |
| large_58 | - | - | - | - |
| large_59 | - | - | - | - |
| large_60 | - | - | - | - |
| large_61 | - | - | - | - |
| large_62 | - | - | - | - |
| large_63 | - | - | - | - |
| large_64 | - | - | - | - |
| large_65 | - | - | - | - |
| large_66 | - | - | - | - |
| large_67 | - | - | - | - |
| large_68 | - | - | - | - |
| large_69 | - | - | - | - |
| large_70 | - | - | - | - |
| large_71 | - | - | - | - |
| large_72 | - | - | - | - |
| large_73 | - | - | - | - |
| large_74 | - | - | - | - |
| large_75 | - | - | - | - |

Continued on next page

Table A.2: The relative objective values for each of the used solvers

| Instance | Gurobi | OR-Tools | Coin | Chuffed |
|---|---|---|---|---|
| large_76 | - | **1.0** | - | - |
| large_77 | - | - | - | - |
| large_78 | - | - | - | - |
| large_79 | - | - | - | - |
| large_80 | - | - | - | - |
| large_81 | - | - | - | - |
| large_82 | - | - | - | - |
| large_83 | - | - | - | - |
| large_84 | - | - | - | - |
| large_85 | - | - | - | - |
| large_86 | - | - | - | - |
| large_87 | - | **1.0** | - | - |
| large_88 | - | **1.0** | - | - |
| large_89 | - | - | - | - |
| large_90 | - | - | - | - |
| large_91 | - | - | - | - |
| large_92 | - | - | - | - |
| large_93 | - | - | - | - |
| large_94 | - | - | - | - |
| large_95 | - | - | - | - |
| large_96 | - | - | - | - |
| large_97 | - | - | - | - |
| large_98 | - | **1.0** | - | - |
| large_99 | - | - | - | - |
| large_100 | - | - | - | - |

# APPENDIX B

# Experimental results of metaheuristic methods

Table B.1: The objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | **Objective (median / mean / std)** | | | |
|---|---|---|---|---|
| | HC | TS | RTS | RTS-RN |
| small_1 | 4204 / 4204.0 / 0.0 | 4184 / 4184.0 / 0.0 | 77 / 77.0 / 0.0 | 68 / 67.8 / 1.1 |
| small_2 | 4196 / 4196.0 / 0.0 | 4176 / 4176.0 / 0.0 | 267 / 267.0 / 0.0 | 257 / 258.8 / 2.49 |
| small_3 | 1689 / 1689.0 / 0.0 | 1679 / 1679.0 / 0.0 | 1675 / 1675.0 / 0.0 | 1685 / 1682.4 / 6.47 |
| small_4 | 2920 / 2920.0 / 0.0 | 2910 / 2910.0 / 0.0 | 2904 / 2904.0 / 0.0 | 192 / 730.0 / 1214.2 |
| small_5 | 1133 / 1133.0 / 0.0 | 1120 / 1120.0 / 0.0 | 149 / 148.8 / 0.45 | 146 / 146.2 / 0.45 |
| small_6 | 1318 / 1318.0 / 0.0 | 1261 / 1261.0 / 0.0 | 1272 / 1272.0 / 0.0 | 1263 / 1261.2 / 3.49 |
| small_7 | 579 / 579.0 / 0.0 | 534 / 534.0 / 0.0 | 534 / 534.0 / 0.0 | 534 / 534.0 / 0.0 |
| small_8 | 1286 / 1286.0 / 0.0 | 890 / 890.0 / 0.0 | 890 / 891.8 / 4.02 | 890 / 890.0 / 0.0 |
| small_9 | 1209 / 1209.0 / 0.0 | 1062 / 1062.0 / 0.0 | 1062 / 1062.0 / 0.0 | 1048 / 1051.6 / 7.44 |
| small_10 | 1028 / 1028.0 / 0.0 | 968 / 968.0 / 0.0 | 918 / 922.0 / 7.68 | 904 / 903.4 / 2.61 |
| small_11 | - / - / - | 47 / 47.0 / 0.0 | 45 / 44.2 / 1.79 | 47 / 46.0 / 2.0 |
| small_12 | 184 / 184.0 / 0.0 | 170 / 170.0 / 0.0 | 170 / 170.0 / 0.0 | 170 / 170.0 / 0.0 |
| small_13 | 216 / 216.0 / 0.0 | 216 / 216.0 / 0.0 | 138 / 138.0 / 0.0 | 146 / 142.6 / 9.07 |
| small_14 | 457 / 457.0 / 0.0 | 457 / 457.0 / 0.0 | 457 / 457.0 / 0.0 | 457 / 457.0 / 0.0 |
| small_15 | 3151 / 3151.0 / 0.0 | 2936 / 2936.0 / 0.0 | 329 / 1365.2 / 1448.19 | 2940 / 2941.4 / 3.51 |
| small_16 | 2058 / 2058.0 / 0.0 | 1971 / 1971.0 / 0.0 | 1980 / 1980.0 / 0.0 | 1978 / 1978.8 / 5.22 |
| small_17 | 772 / 772.0 / 0.0 | 709 / 709.0 / 0.0 | 719 / 716.8 / 3.49 | 713 / 712.2 / 2.17 |
| small_18 | 1905 / 1905.0 / 0.0 | 1877 / 1877.0 / 0.0 | 1871 / 1870.2 / 2.17 | 1877 / 1873.4 / 4.93 |
| small_19 | 525 / 525.0 / 0.0 | 509 / 509.0 / 0.0 | 506 / 506.0 / 0.0 | 502 / 502.4 / 1.82 |
| small_20 | 221 / 221.0 / 0.0 | 203 / 203.0 / 0.0 | 203 / 203.0 / 0.0 | 203 / 203.0 / 0.0 |
| small_21 | 1626 / 1626.0 / 0.0 | 1622 / 1622.0 / 0.0 | 173 / 180.8 / 14.13 | 1432 / 1009.0 / 606.99 |
| small_22 | 170 / 170.0 / 0.0 | 146 / 146.0 / 0.0 | 146 / 146.0 / 0.0 | 146 / 146.0 / 0.0 |
| small_23 | 1039 / 1039.0 / 0.0 | 1034 / 1034.0 / 0.0 | 273 / 322.4 / 186.72 | 762 / 714.4 / 109.82 |
| small_24 | 2681 / 2681.0 / 0.0 | 2605 / 2605.0 / 0.0 | 2609 / 2609.0 / 0.0 | 2556 / 2555.8 / 6.94 |
| small_25 | 588 / 588.0 / 0.0 | 564 / 564.0 / 0.0 | 569 / 569.0 / 0.0 | 571 / 570.0 / 2.45 |
| small_26 | 526 / 526.0 / 0.0 | 513 / 513.0 / 0.0 | 515 / 515.0 / 0.0 | 521 / 520.0 / 3.46 |
| small_27 | 629 / 629.0 / 0.0 | 455 / 455.0 / 0.0 | 457 / 457.0 / 0.0 | 456 / 455.8 / 2.28 |
| small_28 | 1271 / 1271.0 / 0.0 | 1257 / 1257.0 / 0.0 | 1052 / 1013.2 / 86.76 | 1254 / 1255.8 / 3.9 |
| small_29 | 1512 / 1512.0 / 0.0 | 735 / 735.0 / 0.0 | 741 / 741.0 / 4.06 | 730 / 730.0 / 0.0 |
| small_30 | 547 / 547.0 / 0.0 | 547 / 547.0 / 0.0 | 487 / 487.0 / 0.0 | 487 / 487.0 / 0.0 |
| small_31 | 132 / 132.0 / 0.0 | 124 / 124.0 / 0.0 | 124 / 124.0 / 0.0 | 124 / 124.0 / 0.0 |
| small_32 | 2181 / 2181.0 / 0.0 | 2155 / 2155.0 / 0.0 | 2155 / 2155.0 / 0.0 | 384 / 387.6 / 15.06 |
| small_33 | 2799 / 2799.0 / 0.0 | 2738 / 2738.0 / 0.0 | 2739 / 2739.0 / 0.0 | 1371 / 1644.6 / 609.59 |
| small_34 | - / - / - | 311 / 311.0 / 0.0 | 4184 / 4184.0 / 0.0 | 4178 / 4180.4 / 7.8 |
| small_35 | 370 / 370.0 / 0.0 | 342 / 342.0 / 0.0 | 349 / 349.0 / 0.0 | 344 / 345.2 / 3.56 |

Continued on next page

Table B.1: The objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | **Objective (median / mean / std)** | | | |
|---|---|---|---|---|
| | HC | TS | RTS | RTS-RN |
| small_36 | 278 / 278.0 / 0.0 | 197 / 197.0 / 0.0 | 197 / 197.0 / 0.0 | 197 / 197.0 / 0.0 |
| small_37 | 685 / 685.0 / 0.0 | 645 / 645.0 / 0.0 | 645 / 645.0 / 0.0 | 645 / 645.2 / 0.45 |
| small_38 | 1734 / 1734.0 / 0.0 | 915 / 915.0 / 0.0 | 851 / 851.0 / 0.0 | 777 / 941.4 / 339.05 |
| small_39 | 822 / 822.0 / 0.0 | 820 / 820.0 / 0.0 | 820 / 820.0 / 0.0 | 226 / 463.6 / 325.35 |
| small_40 | 2585 / 2585.0 / 0.0 | 2459 / 2459.0 / 0.0 | 366 / 783.6 / 938.85 | 2460 / 2460.2 / 2.39 |
| small_41 | 1278 / 1278.0 / 0.0 | 1255 / 1255.0 / 0.0 | 1261 / 1261.4 / 0.55 | 1259 / 1259.8 / 3.35 |
| small_42 | 1735 / 1735.0 / 0.0 | 288 / 288.0 / 0.0 | 358 / 358.0 / 0.0 | 280 / 278.0 / 5.83 |
| small_43 | 462 / 462.0 / 0.0 | 392 / 392.0 / 0.0 | 392 / 392.0 / 0.0 | 356 / 356.0 / 0.0 |
| small_44 | 965 / 965.0 / 0.0 | 872 / 872.0 / 0.0 | 886 / 886.0 / 0.0 | 878 / 883.4 / 14.71 |
| small_45 | - / - / - | - / - / - | 3776 / 3776.0 / 0.0 | 3766 / 3774.0 / 13.04 |
| small_46 | 1330 / 1330.0 / 0.0 | 1162 / 1162.0 / 0.0 | 1162 / 1162.0 / 0.0 | 1162 / 1162.0 / 0.0 |
| small_47 | 970 / 970.0 / 0.0 | 959 / 959.0 / 0.0 | 282 / 342.2 / 151.75 | 942 / 872.4 / 158.55 |
| small_48 | 336 / 336.0 / 0.0 | 336 / 336.0 / 0.0 | 336 / 336.0 / 0.0 | 336 / 336.0 / 0.0 |
| small_49 | 1833 / 1833.0 / 0.0 | 1814 / 1814.0 / 0.0 | 983 / 981.6 / 9.26 | 980 / 980.2 / 0.45 |
| small_50 | 4491 / 4491.0 / 0.0 | 4480 / 4480.0 / 0.0 | 210 / 208.2 / 7.53 | 132 / 134.6 / 6.99 |
| small_51 | 1254 / 1254.0 / 0.0 | 1254 / 1254.0 / 0.0 | 1119 / 1120.8 / 4.02 | 1119 / 1119.0 / 0.0 |
| small_52 | 773 / 773.0 / 0.0 | 744 / 744.0 / 0.0 | 740 / 740.0 / 0.0 | 742 / 745.4 / 12.48 |
| small_53 | 174 / 174.0 / 0.0 | 158 / 158.0 / 0.0 | 162 / 162.0 / 0.0 | 162 / 162.0 / 1.41 |
| small_54 | 2233 / 2233.0 / 0.0 | 2167 / 2167.0 / 0.0 | 2172 / 2172.0 / 0.0 | 2170 / 2171.8 / 13.46 |
| small_55 | 188 / 188.0 / 0.0 | 188 / 188.0 / 0.0 | 188 / 188.0 / 0.0 | 187 / 187.2 / 0.84 |
| small_56 | 1480 / 1480.0 / 0.0 | 1430 / 1430.0 / 0.0 | 342 / 342.0 / 12.25 | 312 / 312.0 / 0.0 |
| small_57 | 892 / 892.0 / 0.0 | 885 / 885.0 / 0.0 | 888 / 887.2 / 1.3 | 885 / 883.0 / 2.83 |
| small_58 | 434 / 434.0 / 0.0 | 393 / 393.0 / 0.0 | 393 / 393.0 / 0.0 | 393 / 393.8 / 3.63 |
| small_59 | 584 / 584.0 / 0.0 | 552 / 552.0 / 0.0 | 550 / 550.0 / 0.0 | 554 / 554.6 / 6.5 |
| small_60 | 1084 / 1084.0 / 0.0 | 1014 / 1014.0 / 0.0 | 1009 / 1009.0 / 0.0 | 1008 / 1008.0 / 3.94 |
| small_61 | 2480 / 2480.0 / 0.0 | 2464 / 2464.0 / 0.0 | 2406 / 1983.0 / 946.43 | 271 / 271.4 / 2.3 |
| small_62 | 1510 / 1510.0 / 0.0 | 1479 / 1479.0 / 0.0 | 1477 / 1477.6 / 1.34 | 1484 / 1229.6 / 573.33 |
| small_63 | 1466 / 1466.0 / 0.0 | 1452 / 1452.0 / 0.0 | 465 / 465.6 / 8.71 | 431 / 433.6 / 9.15 |
| small_64 | 272 / 272.0 / 0.0 | 270 / 270.0 / 0.0 | 170 / 170.6 / 6.23 | 259 / 259.8 / 1.92 |
| small_65 | 1233 / 1233.0 / 0.0 | 1154 / 1154.0 / 0.0 | 1155 / 1155.0 / 0.0 | 1161 / 1162.0 / 3.74 |
| small_66 | 154 / 154.0 / 0.0 | 145 / 145.0 / 0.0 | 146 / 146.0 / 0.0 | 143 / 142.0 / 5.15 |
| small_67 | 1883 / 1883.0 / 0.0 | 1815 / 1815.0 / 0.0 | 478 / 745.6 / 596.7 | 478 / 477.6 / 5.86 |
| small_68 | 350 / 350.0 / 0.0 | 284 / 284.0 / 0.0 | 220 / 220.0 / 0.0 | 214 / 216.4 / 6.54 |
| small_69 | 319 / 319.0 / 0.0 | 319 / 319.0 / 0.0 | 319 / 319.0 / 0.0 | 319 / 319.0 / 0.0 |
| small_70 | 267 / 267.0 / 0.0 | 254 / 254.0 / 0.0 | 264 / 263.2 / 1.79 | 249 / 248.6 / 2.88 |

Continued on next page

Table B.1: The objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | **Objective (median / mean / std)** | | | |
|---|---|---|---|---|
| | HC | TS | RTS | RTS-RN |
| small_71 | 3185 / 3185.0 / 0.0 | 3118 / 3119.6 / 2.19 | 3103 / 3104.2 / 18.89 | 3068 / 3071.8 / 10.83 |
| small_72 | 1595 / 1595.0 / 0.0 | 1568 / 1568.0 / 0.0 | 1559 / 1559.0 / 0.0 | 263 / 268.4 / 24.23 |
| small_73 | 3089 / 3089.0 / 0.0 | 2737 / 2737.0 / 0.0 | 2752 / 2752.0 / 0.0 | 2739 / 2738.8 / 13.16 |
| small_74 | 2641 / 2641.0 / 0.0 | 2627 / 2627.0 / 0.0 | 2630 / 2630.0 / 0.0 | 1407 / 1470.8 / 138.51 |
| small_75 | 246 / 246.0 / 0.0 | 242 / 242.0 / 0.0 | 242 / 242.0 / 0.0 | 242 / 242.0 / 0.0 |
| small_76 | 1482 / 1482.0 / 0.0 | 1429 / 1429.0 / 0.0 | 1426 / 1426.0 / 10.79 | 1452 / 1452.6 / 10.71 |
| small_77 | 1615 / 1615.0 / 0.0 | 1555 / 1555.0 / 0.0 | 1455 / 1455.0 / 0.0 | 1444 / 1434.8 / 17.8 |
| small_78 | 336 / 336.0 / 0.0 | 316 / 316.0 / 0.0 | 323 / 323.2 / 0.84 | 306 / 305.8 / 9.04 |
| small_79 | 319 / 319.0 / 0.0 | 312 / 312.0 / 0.0 | 137 / 205.4 / 96.41 | 307 / 273.8 / 76.48 |
| small_80 | 1412 / 1412.0 / 0.0 | 1364 / 1364.0 / 0.0 | 1389 / 1389.0 / 0.0 | 1367 / 1363.6 / 13.69 |
| small_81 | 203 / 203.0 / 0.0 | 202 / 202.0 / 0.0 | 203 / 203.0 / 0.0 | 202 / 203.0 / 1.41 |
| small_82 | 2604 / 2604.0 / 0.0 | 2503 / 2503.0 / 0.0 | 2509 / 2509.0 / 0.0 | 2514 / 2512.4 / 4.93 |
| small_83 | 690 / 690.0 / 0.0 | 521 / 521.0 / 0.0 | 541 / 536.6 / 9.84 | 510 / 510.8 / 1.79 |
| small_84 | 494 / 494.0 / 0.0 | 487 / 487.0 / 0.0 | 488 / 488.0 / 0.0 | 396 / 395.6 / 2.3 |
| small_85 | 895 / 895.0 / 0.0 | 770 / 770.0 / 0.0 | 767 / 767.0 / 0.0 | 764 / 763.2 / 4.55 |
| small_86 | 3625 / 3625.0 / 0.0 | 3625 / 3625.0 / 0.0 | 90 / 90.0 / 0.0 | 3552 / 3553.0 / 2.24 |
| small_87 | 4880 / 4880.0 / 0.0 | 4859 / 4859.2 / 0.45 | 4859 / 4859.0 / 0.0 | 3885 / 3988.4 / 412.73 |
| small_88 | 458 / 458.0 / 0.0 | 339 / 339.0 / 0.0 | 339 / 339.0 / 0.0 | 339 / 339.0 / 0.0 |
| small_89 | 182 / 182.0 / 0.0 | 178 / 178.0 / 0.0 | 178 / 177.6 / 0.55 | 178 / 178.0 / 0.0 |
| small_90 | 3473 / 3473.0 / 0.0 | 3446 / 3446.0 / 0.0 | 3448 / 3448.0 / 0.0 | 2074 / 2335.4 / 375.83 |
| small_91 | 2315 / 2315.0 / 0.0 | 2225 / 2225.0 / 0.0 | 2225 / 2225.0 / 0.0 | 2227 / 2227.0 / 3.81 |
| small_92 | 918 / 918.0 / 0.0 | 894 / 894.0 / 0.0 | 894 / 894.0 / 0.0 | 894 / 894.0 / 0.0 |
| small_93 | 464 / 464.0 / 0.0 | 431 / 431.0 / 0.0 | 427 / 427.0 / 0.0 | 420 / 289.0 / 181.21 |
| small_94 | 4253 / 4253.0 / 0.0 | 4055 / 4055.0 / 0.0 | 3974 / 3974.0 / 0.0 | 3983 / 3984.8 / 9.86 |
| small_95 | 3200 / 3200.0 / 0.0 | 1917 / 1917.0 / 0.0 | 1977 / 1974.2 / 3.83 | 1878 / 1555.4 / 449.53 |
| small_96 | 4470 / 4470.0 / 0.0 | 4452 / 4452.0 / 0.0 | 4436 / 4436.0 / 0.0 | 4374 / 4377.0 / 31.89 |
| small_97 | 424 / 424.0 / 0.0 | 86 / 86.0 / 0.0 | 90 / 91.6 / 6.69 | 86 / 86.0 / 0.0 |
| small_98 | 71 / 71.0 / 0.0 | 71 / 71.0 / 0.0 | 71 / 71.0 / 0.0 | 71 / 71.0 / 0.0 |
| small_99 | 146 / 146.0 / 0.0 | 128 / 128.0 / 0.0 | 130 / 130.0 / 1.0 | 124 / 123.6 / 0.89 |
| small_100 | 2850 / 2850.0 / 0.0 | 2850 / 2850.0 / 0.0 | 2629 / 2631.6 / 111.27 | 2850 / 2849.6 / 0.55 |
| medium_1 | 3125 / 3125.0 / 0.0 | 2817 / 2817.0 / 0.0 | 2712 / 2712.0 / 0.0 | 2646 / 2655.8 / 42.58 |
| medium_2 | 7650 / 7650.0 / 0.0 | 7543 / 7543.0 / 0.0 | 6971 / 6957.6 / 127.44 | 7529 / 7529.0 / 2.12 |
| medium_3 | 5342 / 5342.0 / 0.0 | 4634 / 4639.6 / 10.48 | 4369 / 4355.8 / 22.69 | 4725 / 4745.8 / 339.48 |
| medium_4 | 2420 / 2420.0 / 0.0 | 2285 / 2285.0 / 0.0 | 2280 / 2280.0 / 0.0 | 2692 / 2692.0 / 5.79 |
| medium_5 | 3880 / 3880.0 / 0.0 | 3839 / 3839.0 / 0.0 | 3406 / 3571.8 / 244.08 | 3824 / 3824.0 / 4.53 |

Continued on next page

Table B.1: The objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | **Objective (median / mean / std)** | | | |
|---|---|---|---|---|
| | HC | TS | RTS | RTS-RN |
| medium_6 | 4704 / 4704.0 / 0.0 | 4516 / 4516.0 / 0.0 | 4464 / 4462.4 / 3.58 | 4426 / 4430.6 / 26.16 |
| medium_7 | 7382 / 7382.0 / 0.0 | 7125 / 7125.0 / 0.0 | 7106 / 7106.0 / 0.0 | 7046 / 7019.0 / 57.3 |
| medium_8 | 1434 / 1434.0 / 0.0 | 1421 / 1421.0 / 0.0 | 1419 / 1419.0 / 0.0 | 1413 / 1417.8 / 17.56 |
| medium_9 | 1794 / 1794.0 / 0.0 | 1570 / 1570.0 / 0.0 | 1434 / 1434.0 / 0.0 | 1474 / 1467.6 / 43.6 |
| medium_10 | 6042 / 6042.0 / 0.0 | 5910 / 5910.0 / 0.0 | 5949 / 5949.2 / 8.07 | 5891 / 5887.0 / 13.17 |
| medium_11 | 1107 / 1107.0 / 0.0 | 881 / 881.0 / 0.0 | 875 / 876.0 / 1.73 | 859 / 853.6 / 22.81 |
| medium_12 | 2514 / 2514.0 / 0.0 | 2482 / 2482.0 / 0.0 | 2484 / 2484.0 / 0.0 | 2426 / 2427.8 / 22.88 |
| medium_13 | 2673 / 2673.0 / 0.0 | 2579 / 2579.0 / 0.0 | 2563 / 2563.0 / 0.0 | 2553 / 2551.6 / 5.86 |
| medium_14 | 3040 / 3040.0 / 0.0 | 2760 / 2759.0 / 2.24 | 2785 / 2785.0 / 0.0 | 2755 / 2761.0 / 8.22 |
| medium_15 | 1197 / 1197.0 / 0.0 | 1169 / 1169.0 / 0.0 | 1173 / 1173.0 / 0.0 | 1206 / 1200.4 / 17.08 |
| medium_16 | 4738 / 4738.0 / 0.0 | 4718 / 4718.0 / 0.0 | 3888 / 3891.2 / 308.52 | 4709 / 4709.6 / 3.91 |
| medium_17 | 440 / 440.0 / 0.0 | 426 / 426.0 / 0.0 | 418 / 417.8 / 0.45 | 427 / 425.0 / 3.24 |
| medium_18 | 2156 / 2156.0 / 0.0 | 2132 / 2132.0 / 0.0 | 2136 / 1960.8 / 391.76 | 2152 / 2164.0 / 39.5 |
| medium_19 | 4100 / 4100.0 / 0.0 | 3996 / 3996.0 / 0.0 | 4007 / 4007.0 / 0.0 | 3960 / 3961.0 / 10.56 |
| medium_20 | 2069 / 2069.0 / 0.0 | 2048 / 2048.0 / 0.0 | 2052 / 2052.0 / 0.0 | 2052 / 2051.6 / 3.05 |
| medium_21 | 999 / 999.0 / 0.0 | 897 / 897.0 / 0.0 | 873 / 873.0 / 0.0 | 832 / 823.0 / 21.05 |
| medium_22 | 3798 / 3798.0 / 0.0 | 3763 / 3763.0 / 0.0 | 3762 / 3762.0 / 0.0 | 3771 / 3765.2 / 20.96 |
| medium_23 | 2191 / 2191.0 / 0.0 | 2148 / 2148.0 / 0.0 | 2124 / 2124.0 / 0.0 | 2114 / 2112.8 / 15.45 |
| medium_24 | 4303 / 4303.0 / 0.0 | 4138 / 4138.0 / 0.0 | 4186 / 4186.0 / 0.0 | 4093 / 4101.8 / 16.77 |
| medium_25 | 3709 / 3709.0 / 0.0 | 2516 / 2516.0 / 0.0 | 2493 / 2515.0 / 44.06 | 2413 / 2422.4 / 27.58 |
| medium_26 | - / - / - | - / - / - | - / - / - | - / - / - |
| medium_27 | 2896 / 2896.0 / 0.0 | 2874 / 2874.0 / 0.0 | 2869 / 2869.0 / 0.0 | 2834 / 2837.6 / 21.34 |
| medium_28 | 2226 / 2226.0 / 0.0 | 2164 / 2164.0 / 0.0 | 2199 / 2199.4 / 0.89 | 2178 / 2179.8 / 8.79 |
| medium_29 | 8534 / 8534.0 / 0.0 | 8480 / 8481.6 / 2.19 | 8510 / 8506.8 / 7.16 | 8485 / 8485.2 / 2.86 |
| medium_30 | 3045 / 3045.0 / 0.0 | 3002 / 3002.0 / 0.0 | 3031 / 3031.0 / 0.0 | 2925 / 2903.6 / 34.95 |
| medium_31 | 3576 / 3576.0 / 0.0 | 3546 / 3546.0 / 0.0 | 3548 / 3548.0 / 0.0 | 3524 / 3524.2 / 6.18 |
| medium_32 | - / - / - | 484 / 484.0 / 0.0 | 484 / 484.0 / 0.0 | 484 / 484.4 / 0.89 |
| medium_33 | 1082 / 1082.0 / 0.0 | 942 / 942.0 / 0.0 | 917 / 917.0 / 0.0 | 912 / 907.0 / 24.79 |
| medium_34 | 1813 / 1813.0 / 0.0 | 1810 / 1810.0 / 0.0 | 1664 / 1663.2 / 105.37 | 1814 / 1813.8 / 1.48 |
| medium_35 | 2999 / 2999.0 / 0.0 | 2940 / 2940.0 / 0.0 | 2941 / 2941.0 / 0.0 | 2871 / 2881.2 / 30.25 |
| medium_36 | 9072 / 9072.0 / 0.0 | 8844 / 8844.0 / 0.0 | 8965 / 8965.0 / 0.0 | 8839 / 8844.0 / 42.93 |
| medium_37 | 3374 / 3374.0 / 0.0 | 3339 / 3339.0 / 0.0 | 3339 / 3339.0 / 0.0 | 3304 / 3304.6 / 13.52 |
| medium_38 | 3783 / 3783.0 / 0.0 | 3270 / 3270.0 / 0.0 | 3309 / 3305.2 / 16.07 | 3101 / 3112.2 / 60.35 |
| medium_39 | 5040 / 5040.0 / 0.0 | 4845 / 4845.0 / 0.0 | 4844 / 4848.2 / 6.26 | 4820 / 4819.2 / 4.32 |
| medium_40 | - / - / - | 1074 / 1074.0 / 0.0 | 1070 / 1070.0 / 0.0 | 1061 / 1090.4 / 62.09 |

Continued on next page

Table B.1: The objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | **Objective (median / mean / std)** | | | |
|---|---|---|---|---|
| | HC | TS | RTS | RTS-RN |
| medium_41 | 5625 / 5625.0 / 0.0 | 5546 / 5546.0 / 0.0 | 5551 / 5551.0 / 0.0 | 5526 / 5530.2 / 21.73 |
| medium_42 | 1002 / 1002.0 / 0.0 | 956 / 956.0 / 0.0 | 951 / 951.0 / 0.0 | 932 / 939.2 / 26.2 |
| medium_43 | 2932 / 2932.0 / 0.0 | 2780 / 2778.4 / 3.58 | 2673 / 2672.2 / 4.15 | 2659 / 2618.0 / 182.09 |
| medium_44 | 1232 / 1232.0 / 0.0 | 1177 / 1177.0 / 0.0 | 1204 / 1204.0 / 0.0 | 1171 / 1171.0 / 13.69 |
| medium_45 | 4936 / 4936.0 / 0.0 | 4824 / 4824.0 / 0.0 | 4822 / 4821.2 / 1.79 | 4663 / 4385.0 / 665.23 |
| medium_46 | 2824 / 2824.0 / 0.0 | 2500 / 2504.0 / 5.48 | 2450 / 2037.6 / 566.54 | 2450 / 2446.0 / 5.48 |
| medium_47 | 2147 / 2147.0 / 0.0 | 655 / 655.0 / 0.0 | 758 / 758.0 / 0.0 | 588 / 725.6 / 279.26 |
| medium_48 | 3560 / 3560.0 / 0.0 | 3344 / 3344.0 / 0.0 | 3341 / 3341.0 / 0.0 | 3315 / 3312.8 / 7.19 |
| medium_49 | 1494 / 1494.0 / 0.0 | 1328 / 1320.8 / 9.96 | 728 / 970.6 / 351.93 | 1420 / 1424.0 / 37.84 |
| medium_50 | 7393 / 7393.0 / 0.0 | 7278 / 7278.0 / 0.0 | 7270 / 7270.0 / 0.0 | 7255 / 7254.6 / 2.3 |
| medium_51 | 7079 / 7079.0 / 0.0 | 6542 / 6542.0 / 0.0 | 6549 / 6549.0 / 0.0 | 6443 / 6443.6 / 6.07 |
| medium_52 | 1001 / 1001.0 / 0.0 | 1001 / 1001.0 / 0.0 | 1001 / 1001.0 / 0.0 | 1001 / 1001.0 / 0.0 |
| medium_53 | 4052 / 4052.0 / 0.0 | 4010 / 4010.0 / 4.24 | 4010 / 4010.0 / 0.0 | 3978 / 3979.2 / 8.98 |
| medium_54 | 2679 / 2679.0 / 0.0 | 2676 / 2676.0 / 0.0 | 2201 / 2294.2 / 212.97 | 2668 / 2668.2 / 1.48 |
| medium_55 | 2779 / 2779.0 / 0.0 | 1248 / 1221.8 / 86.13 | 949 / 963.2 / 37.9 | 940 / 952.0 / 28.57 |
| medium_56 | 2807 / 2807.0 / 0.0 | 2730 / 2730.0 / 0.0 | 2744 / 2744.0 / 0.0 | 2736 / 2722.6 / 24.44 |
| medium_57 | 16072 / 16072.0 / 0.0 | 15820 / 15820.0 / 0.0 | 15845 / 15845.0 / 0.0 | 15814 / 15813.2 / 5.26 |
| medium_58 | 7689 / 7689.0 / 0.0 | 6522 / 6522.0 / 0.0 | 6575 / 6575.0 / 0.0 | 6468 / 6468.4 / 3.78 |
| medium_59 | 10272 / 10272.0 / 0.0 | 10171 / 10171.0 / 0.0 | 10105 / 10105.0 / 6.36 | 10094 / 10091.2 / 15.32 |
| medium_60 | 872 / 872.0 / 0.0 | 859 / 859.0 / 0.0 | 853 / 853.0 / 0.0 | 860 / 860.6 / 0.89 |
| medium_61 | 6032 / 6032.0 / 0.0 | 6024 / 6024.0 / 0.0 | 5330 / 5022.2 / 677.27 | 6024 / 6023.2 / 1.1 |
| medium_62 | 2747 / 2747.0 / 0.0 | 2724 / 2724.0 / 0.0 | 2723 / 2723.0 / 0.0 | 2720 / 2721.6 / 3.44 |
| medium_63 | 2997 / 2997.0 / 0.0 | 1072 / 1041.6 / 50.73 | 780 / 910.6 / 440.92 | 607 / 724.2 / 200.83 |
| medium_64 | 679 / 679.0 / 0.0 | 144 / 144.0 / 0.0 | 154 / 154.0 / 0.0 | 142 / 140.6 / 6.8 |
| medium_65 | 5578 / 5578.0 / 0.0 | 5548 / 5548.0 / 0.0 | 5500 / 5500.0 / 0.0 | 5506 / 5501.6 / 10.24 |
| medium_66 | 1977 / 1977.0 / 0.0 | 1924 / 1924.0 / 0.0 | 1936 / 1936.0 / 0.0 | 1913 / 1909.0 / 10.79 |
| medium_67 | 3022 / 3022.0 / 0.0 | 2926 / 2926.0 / 0.0 | 2932 / 2932.0 / 0.0 | 2886 / 2886.4 / 39.44 |
| medium_68 | 7038 / 7038.0 / 0.0 | 6728 / 6726.6 / 9.13 | 6755 / 6755.0 / 0.0 | 6667 / 6669.2 / 5.07 |
| medium_69 | 8452 / 8452.0 / 0.0 | 8368 / 8368.0 / 0.0 | 8337 / 8337.0 / 0.0 | 8317 / 8318.0 / 3.67 |
| medium_70 | 1964 / 1964.0 / 0.0 | 1877 / 1877.0 / 0.0 | 1875 / 1875.0 / 0.0 | 1893 / 1893.8 / 2.17 |
| medium_71 | 5662 / 5662.0 / 0.0 | 5633 / 5633.0 / 0.0 | 5629 / 5629.0 / 0.0 | 5597 / 5597.0 / 8.34 |
| medium_72 | 2489 / 2489.0 / 0.0 | 2320 / 2320.0 / 0.0 | 2165 / 2165.0 / 0.0 | 2064 / 2048.8 / 41.54 |
| medium_73 | 3835 / 3835.0 / 0.0 | 3706 / 3706.6 / 1.34 | 3684 / 3684.6 / 1.34 | 3658 / 3655.4 / 20.17 |
| medium_74 | 3984 / 3984.0 / 0.0 | 3939 / 3939.8 / 1.1 | 3930 / 3930.0 / 0.0 | 3801 / 3800.8 / 8.5 |
| medium_75 | 307 / 307.0 / 0.0 | 269 / 269.0 / 0.0 | 287 / 285.4 / 2.19 | 268 / 266.8 / 11.99 |

Continued on next page

Table B.1: The objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | **Objective (median / mean / std)** | | | |
|---|---|---|---|---|
| | HC | TS | RTS | RTS-RN |
| medium_76 | 1688 / 1688.0 / 0.0 | 1667 / 1667.0 / 0.0 | 1667 / 1667.0 / 0.0 | 1664 / 1664.0 / 0.0 |
| medium_77 | 5765 / 5765.0 / 0.0 | 5319 / 5319.0 / 0.0 | 5293 / 5293.0 / 0.0 | 5240 / 5236.8 / 9.2 |
| medium_78 | 2941 / 2941.0 / 0.0 | 2821 / 2821.0 / 0.0 | 2818 / 2818.0 / 0.0 | 2784 / 2776.2 / 21.57 |
| medium_79 | 3471 / 3471.0 / 0.0 | 3389 / 3389.0 / 0.0 | 3393 / 3393.0 / 0.0 | 3227 / 3249.2 / 41.75 |
| medium_80 | 13233 / 13233.0 / 0.0 | 12448 / 12453.8 / 13.54 | 12509 / 12509.0 / 0.0 | 12396 / 12389.8 / 21.11 |
| medium_81 | 4944 / 4944.0 / 0.0 | 4944 / 4944.0 / 0.0 | 3525 / 3637.0 / 494.01 | 4944 / 4944.0 / 0.0 |
| medium_82 | 3854 / 3854.0 / 0.0 | 3844 / 3844.0 / 0.0 | 2747 / 2900.6 / 421.15 | 3848 / 3847.4 / 3.91 |
| medium_83 | 5100 / 5100.0 / 0.0 | 4948 / 4948.0 / 0.0 | 5009 / 5005.0 / 6.52 | 4898 / 4855.6 / 65.6 |
| medium_84 | 1807 / 1807.0 / 0.0 | 1707 / 1707.0 / 0.0 | 1724 / 1724.0 / 0.0 | 1718 / 1713.4 / 27.49 |
| medium_85 | 8348 / 8348.0 / 0.0 | 7633 / 7631.6 / 3.13 | 7600 / 7600.0 / 0.0 | 7472 / 7479.6 / 24.31 |
| medium_86 | - / - / - | - / - / - | - / - / - | - / - / - |
| medium_87 | 2945 / 2945.0 / 0.0 | 2867 / 2867.0 / 0.0 | 2861 / 2861.0 / 0.0 | 2861 / 2861.0 / 0.0 |
| medium_88 | 2665 / 2665.0 / 0.0 | 2518 / 2518.0 / 0.0 | 2525 / 2522.2 / 6.26 | 2483 / 2483.0 / 0.0 |
| medium_89 | 3430 / 3430.0 / 0.0 | 3367 / 3367.0 / 0.0 | 3365 / 3365.4 / 0.89 | 3380 / 3377.8 / 14.91 |
| medium_90 | 6678 / 6678.0 / 0.0 | 6577 / 6576.0 / 4.0 | 6570 / 6570.0 / 0.0 | 6551 / 6523.6 / 59.61 |
| medium_91 | 1702 / 1702.0 / 0.0 | 1670 / 1670.0 / 0.0 | 1665 / 1665.0 / 0.0 | 1682 / 1675.0 / 15.07 |
| medium_92 | 4043 / 4043.0 / 0.0 | 3952 / 3952.0 / 0.0 | 3931 / 3927.0 / 15.17 | 3772 / 3767.2 / 30.58 |
| medium_93 | 452 / 452.0 / 0.0 | 398 / 398.0 / 0.0 | 395 / 395.0 / 0.0 | 369 / 374.2 / 14.2 |
| medium_94 | 4386 / 4386.0 / 0.0 | 4378 / 4378.0 / 0.71 | 4376 / 4376.0 / 0.0 | 4383 / 4382.4 / 1.34 |
| medium_95 | 2508 / 2508.0 / 0.0 | 2479 / 2479.0 / 0.0 | 2478 / 2158.0 / 438.69 | 2430 / 2431.6 / 15.29 |
| medium_96 | 979 / 979.0 / 0.0 | 975 / 975.0 / 0.0 | 975 / 975.0 / 0.0 | 975 / 975.0 / 0.0 |
| medium_97 | 5512 / 5512.0 / 0.0 | 5128 / 5128.6 / 1.34 | 5151 / 5151.0 / 0.0 | 5102 / 5102.6 / 11.52 |
| medium_98 | 8157 / 8157.0 / 0.0 | 8006 / 8006.0 / 0.0 | 8016 / 8016.0 / 0.0 | 7958 / 7199.8 / 1056.08 |
| medium_99 | 4872 / 4872.0 / 0.0 | 4731 / 4731.0 / 0.0 | 4698 / 4698.0 / 0.0 | 4630 / 4630.0 / 18.29 |
| medium_100 | 1239 / 1239.0 / 0.0 | 1110 / 1110.0 / 0.0 | 1127 / 1127.0 / 0.0 | 1118 / 1123.0 / 17.92 |
| large_1 | 7952 / 7952.0 / 0.0 | 7944 / 7944.0 / 0.0 | 7952 / 7952.0 / 0.0 | 7064 / 7068.8 / 7.16 |
| large_2 | 2237 / 2237.0 / 0.0 | 2139 / 2139.0 / 0.0 | 2132 / 2132.0 / 0.0 | 2118 / 2119.4 / 3.13 |
| large_3 | 6501 / 6501.0 / 0.0 | 6430 / 6429.8 / 4.15 | 6457 / 6457.0 / 0.0 | 6323 / 6332.8 / 25.24 |
| large_4 | 4478 / 4478.0 / 0.0 | 4451 / 4451.8 / 1.79 | 4459 / 4461.4 / 9.45 | 4314 / 4325.0 / 23.64 |
| large_5 | 18034 / 18034.0 / 0.0 | 17914 / 17914.0 / 0.0 | 17918 / 17918.0 / 0.0 | 17277 / 17548.0 / 413.59 |
| large_6 | 3387 / 3387.0 / 0.0 | 3380 / 3380.0 / 0.0 | 3387 / 3387.0 / 0.0 | 2982 / 2981.8 / 0.84 |
| large_7 | 14368 / 14368.0 / 0.0 | 14141 / 14141.0 / 0.0 | 14300 / 14286.0 / 23.74 | 14036 / 14037.4 / 14.62 |
| large_8 | 7402 / 7402.0 / 0.0 | 7396 / 7396.0 / 0.0 | 7362 / 7362.0 / 0.0 | 7350 / 7018.6 / 461.12 |
| large_9 | 7308 / 7308.0 / 0.0 | 6954 / 6960.8 / 9.34 | 7081 / 7081.0 / 0.0 | 6492 / 6505.8 / 52.67 |
| large_10 | - / - / - | - / - / - | - / - / - | - / - / - |

Continued on next page

Table B.1: The objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | **Objective (median / mean / std)** | | | |
|---|---|---|---|---|
| | HC | TS | RTS | RTS-RN |
| large_11 | 4945 / 4945.0 / 0.0 | 4891 / 4891.0 / 0.0 | 4916 / 4916.0 / 0.0 | 4723 / 4712.6 / 20.67 |
| large_12 | 19918 / 19918.0 / 0.0 | 19815 / 19813.8 / 2.68 | 19834 / 19829.8 / 6.02 | 19764 / 19759.2 / 20.32 |
| large_13 | 18521 / 18521.0 / 0.0 | 18341 / 18325.0 / 41.59 | 18371 / 18381.0 / 22.36 | 17745 / 17330.2 / 937.65 |
| large_14 | 4787 / 4787.0 / 0.0 | 4711 / 4710.2 / 3.35 | 4721 / 4721.0 / 0.0 | 4755 / 4527.0 / 501.55 |
| large_15 | 7446 / 7446.0 / 0.0 | 7246 / 7247.0 / 2.24 | 7234 / 7234.0 / 0.0 | 7308 / 7301.6 / 25.98 |
| large_16 | 4785 / 4785.0 / 0.0 | 4762 / 4762.0 / 0.0 | 4775 / 4775.0 / 0.0 | 4723 / 4723.8 / 9.2 |
| large_17 | 2076 / 2076.0 / 0.0 | 2072 / 2072.0 / 0.0 | 2070 / 2070.0 / 0.0 | 1985 / 1986.0 / 46.98 |
| large_18 | 4055 / 4055.0 / 0.0 | 4055 / 4044.8 / 44.15 | 3993 / 3999.0 / 14.28 | 3895 / 3882.8 / 111.19 |
| large_19 | 15498 / 15498.0 / 0.0 | 15497 / 15497.0 / 0.0 | 15498 / 15450.0 / 107.33 | 15472 / 15471.4 / 1.52 |
| large_20 | 5051 / 5051.0 / 0.0 | 5009 / 5009.8 / 1.1 | 4995 / 4991.8 / 4.38 | 4858 / 4855.2 / 9.98 |
| large_21 | 6282 / 6275.8 / 19.45 | 6271 / 6265.4 / 19.11 | 6259 / 6275.8 / 35.18 | 4841 / 5063.8 / 542.63 |
| large_22 | 1575 / 1575.0 / 0.0 | 1364 / 1369.4 / 12.07 | 1432 / 1431.2 / 1.79 | 1115 / 1185.0 / 221.1 |
| large_23 | 4896 / 4896.0 / 0.0 | 4865 / 4865.0 / 0.0 | 4871 / 4871.0 / 0.0 | 4740 / 4743.8 / 12.21 |
| large_24 | 2010 / 2010.0 / 0.0 | 1988 / 1988.0 / 0.0 | 1971 / 1971.0 / 0.0 | 1966 / 1977.0 / 27.36 |
| large_25 | 5144 / 5144.0 / 0.0 | 5036 / 5033.6 / 4.34 | 5036 / 5033.4 / 5.37 | 4987 / 4988.8 / 9.83 |
| large_26 | 4538 / 4538.0 / 0.0 | 4510 / 4510.0 / 0.0 | 4516 / 4516.4 / 1.67 | 4450 / 4444.0 / 15.17 |
| large_27 | 5646 / 5646.0 / 0.0 | 5646 / 5646.0 / 0.0 | 5646 / 5646.0 / 0.0 | 5421 / 5422.8 / 7.53 |
| large_28 | 1704 / 1704.0 / 0.0 | 1407 / 1407.0 / 0.0 | 1531 / 1531.0 / 0.0 | 1005 / 1018.2 / 67.67 |
| large_29 | 3436 / 3436.0 / 0.0 | 3363 / 3364.4 / 3.13 | 3380 / 3384.8 / 13.01 | 3222 / 3236.0 / 39.19 |
| large_30 | 4003 / 4003.0 / 0.0 | 3959 / 3959.0 / 0.0 | 3945 / 3946.6 / 3.58 | 3848 / 3768.8 / 207.33 |
| large_31 | 5513 / 5513.0 / 0.0 | 5489 / 5479.4 / 26.17 | 5497 / 5500.2 / 9.12 | 5210 / 5216.2 / 12.01 |
| large_32 | 3849 / 3849.0 / 0.0 | 3824 / 3824.0 / 0.0 | 3833 / 3833.0 / 0.0 | 2855 / 2752.2 / 306.05 |
| large_33 | - / - / - | 1002 / 1002.0 / 0.0 | 973 / 973.0 / 0.0 | 832 / 654.8 / 278.24 |
| large_34 | 4244 / 4244.0 / 0.0 | 4188 / 4188.0 / 0.0 | 4231 / 4231.0 / 0.0 | 4106 / 4099.4 / 22.58 |
| large_35 | 8779 / 8779.0 / 0.0 | 8483 / 8483.0 / 0.0 | 8470 / 8472.8 / 7.01 | 8165 / 8184.0 / 44.78 |
| large_36 | 1269 / 1269.0 / 0.0 | 1108 / 1108.0 / 0.0 | 1213 / 1211.8 / 1.64 | 1033 / 1041.8 / 19.92 |
| large_37 | 2528 / 2554.8 / 43.24 | 2524 / 2547.8 / 37.2 | 2518 / 2534.0 / 33.59 | 2405 / 2434.6 / 77.26 |
| large_38 | 4589 / 4589.0 / 0.0 | 4413 / 4412.8 / 0.45 | 4494 / 4494.4 / 0.89 | 4390 / 4392.0 / 12.51 |
| large_39 | 5067 / 5067.0 / 0.0 | 4913 / 4913.0 / 0.0 | 4942 / 4942.0 / 0.0 | 4898 / 4901.6 / 14.57 |
| large_40 | 1208 / 1208.0 / 0.0 | 1190 / 1189.2 / 1.1 | 1204 / 1201.6 / 5.37 | 1140 / 1159.8 / 42.9 |
| large_41 | 2555 / 2555.0 / 0.0 | 2377 / 2377.0 / 0.0 | 2381 / 2381.0 / 0.0 | 2367 / 2357.0 / 20.09 |
| large_42 | 4727 / 4727.0 / 0.0 | 4687 / 4686.2 / 1.1 | 4666 / 4666.0 / 0.0 | 4645 / 4658.6 / 30.71 |
| large_43 | 2008 / 2008.0 / 0.0 | 1948 / 1947.6 / 0.89 | 1969 / 1969.0 / 0.0 | 1960 / 1958.6 / 9.07 |
| large_44 | 3892 / 3892.0 / 0.0 | 3797 / 3797.0 / 0.0 | 3799 / 3799.0 / 0.0 | 3405 / 3342.4 / 132.85 |
| large_45 | 1474 / 1474.0 / 0.0 | 1440 / 1440.0 / 0.0 | 1434 / 1434.0 / 0.0 | 1390 / 1389.0 / 27.78 |

Continued on next page

Table B.1: The objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | **Objective (median / mean / std)** | | | |
|---|---|---|---|---|
| | HC | TS | RTS | RTS-RN |
| large_46 | 7026 / 7026.0 / 0.0 | 6793 / 6796.2 / 4.38 | 6717 / 6717.0 / 0.0 | 6703 / 6707.0 / 33.56 |
| large_47 | 7593 / 7593.0 / 0.0 | 7528 / 7528.4 / 0.55 | 7547 / 7548.8 / 4.02 | 7449 / 7462.2 / 27.71 |
| large_48 | 2957 / 2957.0 / 0.0 | 2768 / 2768.0 / 0.0 | 2791 / 2793.2 / 4.92 | 2684 / 2680.0 / 69.45 |
| large_49 | 3047 / 3047.0 / 0.0 | 3037 / 3037.0 / 0.0 | 3037 / 3037.0 / 0.0 | 2942 / 2962.6 / 61.47 |
| large_50 | 11625 / 11624.4 / 6.15 | 11623 / 11627.8 / 17.4 | 11618 / 11621.0 / 9.0 | 10201 / 10637.0 / 618.2 |
| large_51 | 8307 / 8307.0 / 0.0 | 8233 / 8233.0 / 0.0 | 8251 / 8251.0 / 0.0 | 7958 / 7960.6 / 6.77 |
| large_52 | 5050 / 5050.0 / 0.0 | 4794 / 4793.4 / 1.34 | 4835 / 4844.2 / 20.57 | 4664 / 4649.8 / 27.99 |
| large_53 | 2572 / 2572.0 / 0.0 | 2411 / 2411.0 / 0.0 | 2410 / 2410.0 / 0.0 | 2349 / 2267.4 / 170.37 |
| large_54 | 13992 / 13992.0 / 0.0 | 13964 / 13964.0 / 0.0 | 13971 / 13971.0 / 0.0 | 13776 / 13775.2 / 15.71 |
| large_55 | 6141 / 6141.0 / 0.0 | 5836 / 5838.4 / 5.37 | 5940 / 5940.0 / 0.0 | 5818 / 5816.6 / 8.47 |
| large_56 | - / - / - | 918 / 918.0 / 0.0 | 878 / 877.2 / 1.79 | 739 / 737.8 / 11.73 |
| large_57 | 9975 / 9975.0 / 0.0 | 9959 / 9959.0 / 0.0 | 9959 / 9959.0 / 0.0 | 9763 / 9766.4 / 10.14 |
| large_58 | 5981 / 5981.0 / 0.0 | 5966 / 5966.8 / 1.79 | 5962 / 5964.0 / 4.47 | 5840 / 5860.4 / 63.48 |
| large_59 | 3630 / 3630.0 / 0.0 | 3595 / 3593.8 / 2.68 | 3585 / 3585.0 / 0.0 | 3567 / 3566.0 / 20.54 |
| large_60 | - / - / - | - / - / - | - / - / - | 1789 / 1773.6 / 97.41 |
| large_61 | 5660 / 5660.0 / 0.0 | 5655 / 5655.0 / 0.0 | 5651 / 5651.0 / 0.0 | 5674 / 5402.8 / 612.59 |
| large_62 | 2473 / 2473.0 / 0.0 | 2441 / 2439.8 / 2.68 | 2461 / 2460.4 / 1.34 | 2385 / 2392.6 / 29.67 |
| large_63 | 2313 / 2313.0 / 0.0 | 2293 / 2293.0 / 0.0 | 2293 / 2293.0 / 0.0 | 2294 / 2293.6 / 0.55 |
| large_64 | - / - / - | 2229 / 2228.8 / 0.84 | 2246 / 2246.0 / 0.0 | 2243 / 2228.8 / 41.63 |
| large_65 | 4839 / 4839.0 / 0.0 | 4784 / 4784.0 / 0.0 | 4759 / 4759.0 / 0.0 | 4794 / 4792.4 / 42.66 |
| large_66 | 1295 / 1295.0 / 0.0 | 1295 / 1295.0 / 0.0 | 1295 / 1295.0 / 0.0 | 1273 / 1275.4 / 5.27 |
| large_67 | 12705 / 12705.0 / 0.0 | 12683 / 12683.0 / 0.0 | 12684 / 12684.0 / 0.0 | 12342 / 12478.4 / 190.91 |
| large_68 | 988 / 988.0 / 0.0 | 987 / 987.0 / 0.0 | 987 / 987.0 / 0.0 | 967 / 968.6 / 3.51 |
| large_69 | - / - / - | - / - / - | - / - / - | - / - / - |
| large_70 | 20780 / 20780.0 / 0.0 | 20768 / 20768.0 / 0.0 | 20764 / 20764.0 / 0.0 | 20762 / 20723.6 / 88.11 |
| large_71 | 14767 / 14767.0 / 0.0 | 14763 / 14763.0 / 0.0 | 14763 / 14763.0 / 0.0 | 14766 / 14765.2 / 4.97 |
| large_72 | 7438 / 7438.0 / 0.0 | 7333 / 7335.0 / 4.47 | 7353 / 7354.0 / 2.24 | 7279 / 7279.8 / 9.26 |
| large_73 | 4751 / 4751.0 / 0.0 | 4662 / 4662.0 / 0.0 | 4570 / 4572.6 / 15.29 | 4365 / 3775.8 / 834.04 |
| large_74 | 5007 / 5007.0 / 0.0 | 4915 / 4915.0 / 0.0 | 4878 / 4878.0 / 2.0 | 4752 / 4754.2 / 14.34 |
| large_75 | 1951 / 1951.0 / 0.0 | 1821 / 1821.0 / 0.0 | 1857 / 1857.0 / 0.0 | 1898 / 1894.8 / 14.06 |
| large_76 | 4006 / 4006.0 / 0.0 | 3937 / 3936.6 / 0.89 | 3978 / 3978.0 / 0.0 | 4050 / 4054.4 / 17.56 |
| large_77 | 3277 / 3277.0 / 0.0 | 3197 / 3197.0 / 0.0 | 3194 / 3194.0 / 0.0 | 2904 / 2918.4 / 53.8 |
| large_78 | 6015 / 6015.0 / 0.0 | 5985 / 5985.0 / 0.0 | 6005 / 6005.0 / 0.0 | 5783 / 5785.0 / 9.62 |
| large_79 | 4294 / 4290.8 / 11.19 | 4292 / 4291.6 / 9.94 | 4290 / 4290.8 / 12.13 | 3986 / 3991.0 / 36.98 |
| large_80 | 1889 / 1889.0 / 0.0 | 1884 / 1884.0 / 0.0 | 1884 / 1884.0 / 0.0 | 1872 / 1872.4 / 2.07 |

Continued on next page

Table B.1: The objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | **Objective (median / mean / std)** | | | |
|---|---|---|---|---|
| | HC | TS | RTS | RTS-RN |
| large_81 | 1842 / 1842.0 / 0.0 | 1800 / 1800.0 / 0.0 | 1829 / 1829.0 / 0.0 | 1301 / 1294.2 / 61.28 |
| large_82 | 16476 / 16476.0 / 0.0 | 16303 / 16303.0 / 0.0 | 16191 / 16192.0 / 9.27 | 14742 / 14884.6 / 438.45 |
| large_83 | 3205 / 3205.0 / 0.0 | 2744 / 2744.0 / 0.0 | 3039 / 3034.6 / 9.84 | 934 / 932.6 / 444.64 |
| large_84 | 3650 / 3657.6 / 19.86 | 3653 / 3659.4 / 17.98 | 3663 / 3660.4 / 18.89 | 1924 / 1934.0 / 151.75 |
| large_85 | 6182 / 6188.4 / 9.74 | 6182 / 6185.2 / 5.22 | 6200 / 6206.0 / 16.06 | 6103 / 6103.0 / 37.91 |
| large_86 | 5454 / 5454.0 / 0.0 | 5343 / 5343.0 / 0.0 | 5392 / 5390.0 / 4.47 | 5273 / 5256.8 / 32.7 |
| large_87 | 12511 / 12511.0 / 0.0 | 12393 / 12393.8 / 1.1 | 12419 / 12419.0 / 0.0 | 12044 / 12070.0 / 71.16 |
| large_88 | 16268 / 16268.0 / 0.0 | 16190 / 16190.0 / 0.0 | 16268 / 16268.0 / 0.0 | 15531 / 15374.4 / 392.85 |
| large_89 | 2788 / 2788.0 / 0.0 | 2735 / 2735.0 / 0.0 | 2755 / 2755.0 / 0.0 | 2511 / 2523.2 / 44.29 |
| large_90 | 2031 / 2031.0 / 0.0 | 1923 / 1924.0 / 2.24 | 1851 / 1851.0 / 0.0 | 1283 / 1400.0 / 178.96 |
| large_91 | 6028 / 6028.0 / 0.0 | 6000 / 6000.0 / 0.0 | 6009 / 6009.0 / 0.0 | 6039 / 6023.4 / 43.65 |
| large_92 | 8391 / 8391.0 / 0.0 | 8324 / 8324.0 / 0.0 | 8330 / 8329.2 / 1.1 | 6541 / 6546.8 / 15.66 |
| large_93 | 8139 / 8139.0 / 0.0 | 8082 / 8082.0 / 0.0 | 8091 / 8091.0 / 2.83 | 8017 / 8017.2 / 12.87 |
| large_94 | 3071 / 3071.0 / 0.0 | 2997 / 2996.4 / 1.34 | 3046 / 3046.8 / 2.17 | 1505 / 1525.8 / 69.68 |
| large_95 | 3876 / 3876.0 / 0.0 | 3415 / 3401.0 / 39.75 | 3515 / 3515.0 / 0.0 | 3209 / 3210.4 / 8.26 |
| large_96 | 5053 / 5053.0 / 0.0 | 4918 / 4910.8 / 9.96 | 4959 / 4960.8 / 3.03 | 4850 / 4859.8 / 33.71 |
| large_97 | 7229 / 7229.0 / 0.0 | 6798 / 6807.6 / 20.94 | 6991 / 6987.4 / 8.05 | 6235 / 6211.0 / 55.06 |
| large_98 | 1361 / 1361.0 / 0.0 | 1285 / 1285.0 / 0.0 | 1289 / 1289.0 / 0.0 | 1276 / 1277.0 / 9.43 |
| large_99 | 22336 / 22336.0 / 0.0 | 22307 / 22288.6 / 55.57 | 22234 / 22234.0 / 0.0 | 20841 / 20838.4 / 96.14 |
| large_100 | 4021 / 4021.0 / 0.0 | 3559 / 3559.2 / 0.45 | 3561 / 3561.0 / 0.0 | 3547 / 3632.0 / 202.05 |

Table B.2: The relative mean objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | HC | TS | RTS | RTS-RN |
|---|---|---|---|---|
| small_1 | 62.0059 | 61.7109 | 1.1357 | **1.0** |
| small_2 | 16.2133 | 16.136 | 1.0317 | **1.0** |
| small_3 | 1.0084 | 1.0024 | **1.0** | 1.0044 |
| small_4 | 4.0 | 3.9863 | 3.9781 | **1.0** |
| small_5 | 7.7497 | 7.6607 | 1.0178 | **1.0** |
| small_6 | 1.0452 | **1.0** | 1.0087 | 1.0002 |
| small_7 | 1.0843 | **1.0** | **1.0** | **1.0** |
| small_8 | 1.4449 | **1.0** | 1.002 | **1.0** |
| small_9 | 1.1497 | 1.0099 | 1.0099 | **1.0** |
| small_10 | 1.1379 | 1.0715 | 1.0206 | **1.0** |
| small_11 | - | 1.0633 | **1.0** | 1.0407 |
| small_12 | 1.0824 | **1.0** | **1.0** | **1.0** |
| small_13 | 1.5652 | 1.5652 | **1.0** | 1.0333 |
| small_14 | **1.0** | **1.0** | **1.0** | **1.0** |
| small_15 | 2.3081 | 2.1506 | **1.0** | 2.1546 |
| small_16 | 1.0441 | **1.0** | 1.0046 | 1.004 |
| small_17 | 1.0889 | **1.0** | 1.011 | 1.0045 |
| small_18 | 1.0186 | 1.0036 | **1.0** | 1.0017 |
| small_19 | 1.045 | 1.0131 | 1.0072 | **1.0** |
| small_20 | 1.0887 | **1.0** | **1.0** | **1.0** |
| small_21 | 8.9934 | 8.9712 | **1.0** | 5.5808 |
| small_22 | 1.1644 | **1.0** | **1.0** | **1.0** |
| small_23 | 3.2227 | 3.2072 | **1.0** | 2.2159 |
| small_24 | 1.049 | 1.0193 | 1.0208 | **1.0** |
| small_25 | 1.0426 | **1.0** | 1.0089 | 1.0106 |
| small_26 | 1.0253 | **1.0** | 1.0039 | 1.0136 |
| small_27 | 1.3824 | **1.0** | 1.0044 | 1.0018 |
| small_28 | 1.2544 | 1.2406 | **1.0** | 1.2394 |
| small_29 | 2.0712 | 1.0068 | 1.0151 | **1.0** |
| small_30 | 1.1232 | 1.1232 | **1.0** | **1.0** |
| small_31 | 1.0645 | **1.0** | **1.0** | **1.0** |
| small_32 | 5.6269 | 5.5599 | 5.5599 | **1.0** |
| small_33 | 1.7019 | 1.6648 | 1.6655 | **1.0** |
| small_34 | - | **1.0** | 13.4534 | 13.4418 |
| small_35 | 1.0819 | **1.0** | 1.0205 | 1.0094 |
| small_36 | 1.4112 | **1.0** | **1.0** | **1.0** |
| small_37 | 1.062 | **1.0** | **1.0** | 1.0003 |
| small_38 | 2.0376 | 1.0752 | **1.0** | 1.1062 |
| small_39 | 1.7731 | 1.7688 | 1.7688 | **1.0** |
| small_40 | 3.2989 | 3.1381 | **1.0** | 3.1396 |
| small_41 | 1.0183 | **1.0** | 1.0051 | 1.0038 |
| small_42 | 6.241 | 1.036 | 1.2878 | **1.0** |
| small_43 | 1.2978 | 1.1011 | 1.1011 | **1.0** |
| small_44 | 1.1067 | **1.0** | 1.0161 | 1.0131 |
| small_45 | - | - | 1.0005 | **1.0** |
| small_46 | 1.1446 | **1.0** | **1.0** | **1.0** |
| small_47 | 2.8346 | 2.8025 | **1.0** | 2.5494 |
| small_48 | **1.0** | **1.0** | **1.0** | **1.0** |
| small_49 | 1.87 | 1.8506 | 1.0014 | **1.0** |
| small_50 | 33.3655 | 33.2838 | 1.5468 | **1.0** |
| small_51 | 1.1206 | 1.1206 | 1.0016 | **1.0** |
| small_52 | 1.0446 | 1.0054 | **1.0** | 1.0073 |

Continued on next page

Table B.2: The relative mean objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | HC | TS | RTS | RTS-RN |
|---|---|---|---|---|
| small_53 | 1.1013 | **1.0** | 1.0253 | 1.0253 |
| small_54 | 1.0305 | **1.0** | 1.0023 | 1.0022 |
| small_55 | 1.0043 | 1.0043 | 1.0043 | **1.0** |
| small_56 | 4.7436 | 4.5833 | 1.0962 | **1.0** |
| small_57 | 1.0102 | 1.0023 | 1.0048 | **1.0** |
| small_58 | 1.1043 | **1.0** | **1.0** | 1.002 |
| small_59 | 1.0618 | 1.0036 | **1.0** | 1.0084 |
| small_60 | 1.0754 | 1.006 | 1.001 | **1.0** |
| small_61 | 9.1378 | 9.0789 | 7.3066 | **1.0** |
| small_62 | 1.228 | 1.2028 | 1.2017 | **1.0** |
| small_63 | 3.381 | 3.3487 | 1.0738 | **1.0** |
| small_64 | 1.5944 | 1.5826 | **1.0** | 1.5229 |
| small_65 | 1.0685 | **1.0** | 1.0009 | 1.0069 |
| small_66 | 1.0845 | 1.0211 | 1.0282 | **1.0** |
| small_67 | 3.9426 | 3.8003 | 1.5611 | **1.0** |
| small_68 | 1.6174 | 1.3124 | 1.0166 | **1.0** |
| small_69 | **1.0** | **1.0** | **1.0** | **1.0** |
| small_70 | 1.074 | 1.0217 | 1.0587 | **1.0** |
| small_71 | 1.0369 | 1.0156 | 1.0105 | **1.0** |
| small_72 | 5.9426 | 5.842 | 5.8085 | **1.0** |
| small_73 | 1.1286 | **1.0** | 1.0055 | 1.0007 |
| small_74 | 1.7956 | 1.7861 | 1.7881 | **1.0** |
| small_75 | 1.0165 | **1.0** | **1.0** | **1.0** |
| small_76 | 1.0393 | 1.0021 | **1.0** | 1.0187 |
| small_77 | 1.1256 | 1.0838 | 1.0141 | **1.0** |
| small_78 | 1.0988 | 1.0334 | 1.0569 | **1.0** |
| small_79 | 1.5531 | 1.519 | **1.0** | 1.333 |
| small_80 | 1.0355 | 1.0003 | 1.0186 | **1.0** |
| small_81 | 1.005 | **1.0** | 1.005 | 1.005 |
| small_82 | 1.0404 | **1.0** | 1.0024 | 1.0038 |
| small_83 | 1.3508 | 1.02 | 1.0505 | **1.0** |
| small_84 | 1.2487 | 1.231 | 1.2336 | **1.0** |
| small_85 | 1.1727 | 1.0089 | 1.005 | **1.0** |
| small_86 | 40.2778 | 40.2778 | **1.0** | 39.4778 |
| small_87 | 1.2235 | 1.2183 | 1.2183 | **1.0** |
| small_88 | 1.351 | **1.0** | **1.0** | **1.0** |
| small_89 | 1.0248 | 1.0023 | **1.0** | 1.0023 |
| small_90 | 1.4871 | 1.4756 | 1.4764 | **1.0** |
| small_91 | 1.0404 | **1.0** | **1.0** | 1.0009 |
| small_92 | 1.0268 | **1.0** | **1.0** | **1.0** |
| small_93 | 1.6055 | 1.4913 | 1.4775 | **1.0** |
| small_94 | 1.0702 | 1.0204 | **1.0** | 1.0027 |
| small_95 | 2.0573 | 1.2325 | 1.2693 | **1.0** |
| small_96 | 1.0212 | 1.0171 | 1.0135 | **1.0** |
| small_97 | 4.9302 | **1.0** | 1.0651 | **1.0** |
| small_98 | **1.0** | **1.0** | **1.0** | **1.0** |
| small_99 | 1.1812 | 1.0356 | 1.0518 | **1.0** |
| small_100 | 1.083 | 1.083 | **1.0** | 1.0828 |
| medium_1 | 1.1767 | 1.0607 | 1.0212 | **1.0** |
| medium_2 | 1.0995 | 1.0841 | **1.0** | 1.0821 |
| medium_3 | 1.2264 | 1.0652 | **1.0** | 1.0895 |
| medium_4 | 1.0614 | 1.0022 | **1.0** | 1.1807 |

Continued on next page

Table B.2: The relative mean objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | HC | TS | RTS | RTS-RN |
|---|---|---|---|---|
| medium_5 | 1.0863 | 1.0748 | **1.0** | 1.0706 |
| medium_6 | 1.0617 | 1.0193 | 1.0072 | **1.0** |
| medium_7 | 1.0517 | 1.0151 | 1.0124 | **1.0** |
| medium_8 | 1.0114 | 1.0023 | 1.0008 | **1.0** |
| medium_9 | 1.251 | 1.0948 | **1.0** | 1.0234 |
| medium_10 | 1.0263 | 1.0039 | 1.0106 | **1.0** |
| medium_11 | 1.2969 | 1.0321 | 1.0262 | **1.0** |
| medium_12 | 1.0355 | 1.0223 | 1.0231 | **1.0** |
| medium_13 | 1.0476 | 1.0107 | 1.0045 | **1.0** |
| medium_14 | 1.1018 | **1.0** | 1.0094 | 1.0007 |
| medium_15 | 1.024 | **1.0** | 1.0034 | 1.0269 |
| medium_16 | 1.2176 | 1.2125 | **1.0** | 1.2103 |
| medium_17 | 1.0531 | 1.0196 | **1.0** | 1.0172 |
| medium_18 | 1.0996 | 1.0873 | **1.0** | 1.1036 |
| medium_19 | 1.0351 | 1.0088 | 1.0116 | **1.0** |
| medium_20 | 1.0103 | **1.0** | 1.002 | 1.0018 |
| medium_21 | 1.2139 | 1.0899 | 1.0608 | **1.0** |
| medium_22 | 1.0096 | 1.0003 | **1.0** | 1.0009 |
| medium_23 | 1.037 | 1.0167 | 1.0053 | **1.0** |
| medium_24 | 1.0491 | 1.0088 | 1.0205 | **1.0** |
| medium_25 | 1.5311 | 1.0386 | 1.0382 | **1.0** |
| medium_26 | - | - | - | - |
| medium_27 | 1.0206 | 1.0128 | 1.0111 | **1.0** |
| medium_28 | 1.0287 | **1.0** | 1.0164 | 1.0073 |
| medium_29 | 1.0062 | **1.0** | 1.003 | 1.0004 |
| medium_30 | 1.0487 | 1.0339 | 1.0439 | **1.0** |
| medium_31 | 1.0147 | 1.0062 | 1.0068 | **1.0** |
| medium_32 | - | **1.0** | **1.0** | 1.0008 |
| medium_33 | 1.1929 | 1.0386 | 1.011 | **1.0** |
| medium_34 | 1.0901 | 1.0883 | **1.0** | 1.0905 |
| medium_35 | 1.0409 | 1.0204 | 1.0208 | **1.0** |
| medium_36 | 1.0258 | **1.0** | 1.0137 | **1.0** |
| medium_37 | 1.021 | 1.0104 | 1.0104 | **1.0** |
| medium_38 | 1.2155 | 1.0507 | 1.062 | **1.0** |
| medium_39 | 1.0458 | 1.0054 | 1.006 | **1.0** |
| medium_40 | - | 1.0037 | **1.0** | 1.0191 |
| medium_41 | 1.0171 | 1.0029 | 1.0038 | **1.0** |
| medium_42 | 1.0669 | 1.0179 | 1.0126 | **1.0** |
| medium_43 | 1.1199 | 1.0613 | 1.0207 | **1.0** |
| medium_44 | 1.0521 | 1.0051 | 1.0282 | **1.0** |
| medium_45 | 1.1257 | 1.1001 | 1.0995 | **1.0** |
| medium_46 | 1.3859 | 1.2289 | **1.0** | 1.2004 |
| medium_47 | 3.2779 | **1.0** | 1.1573 | 1.1078 |
| medium_48 | 1.0746 | 1.0094 | 1.0085 | **1.0** |
| medium_49 | 1.5393 | 1.3608 | **1.0** | 1.4671 |
| medium_50 | 1.0191 | 1.0032 | 1.0021 | **1.0** |
| medium_51 | 1.0986 | 1.0153 | 1.0164 | **1.0** |
| medium_52 | **1.0** | **1.0** | **1.0** | **1.0** |
| medium_53 | 1.0183 | 1.0077 | 1.0077 | **1.0** |
| medium_54 | 1.1677 | 1.1664 | **1.0** | 1.163 |
| medium_55 | 2.9191 | 1.2834 | 1.0118 | **1.0** |
| medium_56 | 1.031 | 1.0027 | 1.0079 | **1.0** |

Continued on next page

Table B.2: The relative mean objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | HC | TS | RTS | RTS-RN |
|---|---|---|---|---|
| medium_57 | 1.0164 | 1.0004 | 1.002 | **1.0** |
| medium_58 | 1.1887 | 1.0083 | 1.0165 | **1.0** |
| medium_59 | 1.0179 | 1.0079 | 1.0014 | **1.0** |
| medium_60 | 1.0223 | 1.007 | **1.0** | 1.0089 |
| medium_61 | 1.2011 | 1.1995 | **1.0** | 1.1993 |
| medium_62 | 1.0093 | 1.0009 | 1.0005 | **1.0** |
| medium_63 | 4.1384 | 1.4383 | 1.2574 | **1.0** |
| medium_64 | 4.8293 | 1.0242 | 1.0953 | **1.0** |
| medium_65 | 1.0142 | 1.0087 | **1.0** | 1.0003 |
| medium_66 | 1.0356 | 1.0079 | 1.0141 | **1.0** |
| medium_67 | 1.047 | 1.0137 | 1.0158 | **1.0** |
| medium_68 | 1.0553 | 1.0086 | 1.0129 | **1.0** |
| medium_69 | 1.0161 | 1.006 | 1.0023 | **1.0** |
| medium_70 | 1.0475 | 1.0011 | **1.0** | 1.01 |
| medium_71 | 1.0116 | 1.0064 | 1.0057 | **1.0** |
| medium_72 | 1.2149 | 1.1324 | 1.0567 | **1.0** |
| medium_73 | 1.0491 | 1.014 | 1.008 | **1.0** |
| medium_74 | 1.0482 | 1.0366 | 1.034 | **1.0** |
| medium_75 | 1.1507 | 1.0082 | 1.0697 | **1.0** |
| medium_76 | 1.0144 | 1.0018 | 1.0018 | **1.0** |
| medium_77 | 1.1009 | 1.0157 | 1.0107 | **1.0** |
| medium_78 | 1.0594 | 1.0161 | 1.0151 | **1.0** |
| medium_79 | 1.0683 | 1.043 | 1.0443 | **1.0** |
| medium_80 | 1.0681 | 1.0052 | 1.0096 | **1.0** |
| medium_81 | 1.3594 | 1.3594 | **1.0** | 1.3594 |
| medium_82 | 1.3287 | 1.3252 | **1.0** | 1.3264 |
| medium_83 | 1.0503 | 1.019 | 1.0308 | **1.0** |
| medium_84 | 1.0586 | **1.0** | 1.01 | 1.0037 |
| medium_85 | 1.1161 | 1.0203 | 1.0161 | **1.0** |
| medium_86 | - | - | - | - |
| medium_87 | 1.0294 | 1.0021 | **1.0** | **1.0** |
| medium_88 | 1.0733 | 1.0141 | 1.0158 | **1.0** |
| medium_89 | 1.0192 | 1.0005 | **1.0** | 1.0037 |
| medium_90 | 1.0237 | 1.008 | 1.0071 | **1.0** |
| medium_91 | 1.0222 | 1.003 | **1.0** | 1.006 |
| medium_92 | 1.0732 | 1.0491 | 1.0424 | **1.0** |
| medium_93 | 1.2079 | 1.0636 | 1.0556 | **1.0** |
| medium_94 | 1.0023 | 1.0005 | **1.0** | 1.0015 |
| medium_95 | 1.1622 | 1.1487 | **1.0** | 1.1268 |
| medium_96 | 1.0041 | **1.0** | **1.0** | **1.0** |
| medium_97 | 1.0802 | 1.0051 | 1.0095 | **1.0** |
| medium_98 | 1.1329 | 1.112 | 1.1134 | **1.0** |
| medium_99 | 1.0523 | 1.0218 | 1.0147 | **1.0** |
| medium_100 | 1.1162 | **1.0** | 1.0153 | 1.0117 |
| large_1 | 1.1249 | 1.1238 | 1.1249 | **1.0** |
| large_2 | 1.0555 | 1.0092 | 1.0059 | **1.0** |
| large_3 | 1.0266 | 1.0153 | 1.0196 | **1.0** |
| large_4 | 1.0354 | 1.0293 | 1.0315 | **1.0** |
| large_5 | 1.0277 | 1.0209 | 1.0211 | **1.0** |
| large_6 | 1.1359 | 1.1335 | 1.1359 | **1.0** |
| large_7 | 1.0236 | 1.0074 | 1.0177 | **1.0** |
| large_8 | 1.0546 | 1.0538 | 1.0489 | **1.0** |

Continued on next page

Table B.2: The relative mean objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | HC | TS | RTS | RTS-RN |
|---|---|---|---|---|
| large_9 | 1.1233 | 1.0699 | 1.0884 | **1.0** |
| large_10 | - | - | - | - |
| large_11 | 1.0493 | 1.0379 | 1.0432 | **1.0** |
| large_12 | 1.008 | 1.0028 | 1.0036 | **1.0** |
| large_13 | 1.0687 | 1.0574 | 1.0606 | **1.0** |
| large_14 | 1.0574 | 1.0405 | 1.0429 | **1.0** |
| large_15 | 1.0293 | 1.0018 | **1.0** | 1.0093 |
| large_16 | 1.013 | 1.0081 | 1.0108 | **1.0** |
| large_17 | 1.0453 | 1.0433 | 1.0423 | **1.0** |
| large_18 | 1.0443 | 1.0417 | 1.0299 | **1.0** |
| large_19 | 1.0031 | 1.003 | **1.0** | 1.0014 |
| large_20 | 1.0403 | 1.0318 | 1.0281 | **1.0** |
| large_21 | 1.2393 | 1.2373 | 1.2393 | **1.0** |
| large_22 | 1.3291 | 1.1556 | 1.2078 | **1.0** |
| large_23 | 1.0321 | 1.0255 | 1.0268 | **1.0** |
| large_24 | 1.0198 | 1.0086 | **1.0** | 1.003 |
| large_25 | 1.0311 | 1.009 | 1.0089 | **1.0** |
| large_26 | 1.0212 | 1.0149 | 1.0163 | **1.0** |
| large_27 | 1.0412 | 1.0412 | 1.0412 | **1.0** |
| large_28 | 1.6735 | 1.3819 | 1.5036 | **1.0** |
| large_29 | 1.0618 | 1.0397 | 1.046 | **1.0** |
| large_30 | 1.0621 | 1.0505 | 1.0472 | **1.0** |
| large_31 | 1.0569 | 1.0505 | 1.0544 | **1.0** |
| large_32 | 1.3985 | 1.3894 | 1.3927 | **1.0** |
| large_33 | - | 1.5302 | 1.4859 | **1.0** |
| large_34 | 1.0353 | 1.0216 | 1.0321 | **1.0** |
| large_35 | 1.0727 | 1.0365 | 1.0353 | **1.0** |
| large_36 | 1.2181 | 1.0635 | 1.1632 | **1.0** |
| large_37 | 1.0494 | 1.0465 | 1.0408 | **1.0** |
| large_38 | 1.0449 | 1.0047 | 1.0233 | **1.0** |
| large_39 | 1.0337 | 1.0023 | 1.0082 | **1.0** |
| large_40 | 1.0416 | 1.0253 | 1.036 | **1.0** |
| large_41 | 1.084 | 1.0085 | 1.0102 | **1.0** |
| large_42 | 1.0147 | 1.0059 | 1.0016 | **1.0** |
| large_43 | 1.031 | **1.0** | 1.011 | 1.0056 |
| large_44 | 1.1644 | 1.136 | 1.1366 | **1.0** |
| large_45 | 1.0612 | 1.0367 | 1.0324 | **1.0** |
| large_46 | 1.0476 | 1.0133 | 1.0015 | **1.0** |
| large_47 | 1.0175 | 1.0089 | 1.0116 | **1.0** |
| large_48 | 1.1034 | 1.0328 | 1.0422 | **1.0** |
| large_49 | 1.0285 | 1.0251 | 1.0251 | **1.0** |
| large_50 | 1.0928 | 1.0931 | 1.0925 | **1.0** |
| large_51 | 1.0435 | 1.0342 | 1.0365 | **1.0** |
| large_52 | 1.0861 | 1.0309 | 1.0418 | **1.0** |
| large_53 | 1.1343 | 1.0633 | 1.0629 | **1.0** |
| large_54 | 1.0157 | 1.0137 | 1.0142 | **1.0** |
| large_55 | 1.0558 | 1.0037 | 1.0212 | **1.0** |
| large_56 | - | 1.2442 | 1.1889 | **1.0** |
| large_57 | 1.0214 | 1.0197 | 1.0197 | **1.0** |
| large_58 | 1.0206 | 1.0182 | 1.0177 | **1.0** |
| large_59 | 1.0179 | 1.0078 | 1.0053 | **1.0** |
| large_60 | - | - | - | **1.0** |

Continued on next page

Table B.2: The relative mean objective values for the hill climber (HC), Tabu Search (TS), Reactive Tabu Search (RTS) and Reactive Tabu Search with restricted neighborhood (RTS-RN)

| | HC | TS | RTS | RTS-RN |
|---|---|---|---|---|
| large_61 | 1.0476 | 1.0467 | 1.0459 | **1.0** |
| large_62 | 1.0336 | 1.0197 | 1.0283 | **1.0** |
| large_63 | 1.0087 | **1.0** | **1.0** | 1.0003 |
| large_64 | - | **1.0** | 1.0077 | **1.0** |
| large_65 | 1.0168 | 1.0053 | **1.0** | 1.007 |
| large_66 | 1.0154 | 1.0154 | 1.0154 | **1.0** |
| large_67 | 1.0182 | 1.0164 | 1.0165 | **1.0** |
| large_68 | 1.02 | 1.019 | 1.019 | **1.0** |
| large_69 | - | - | - | - |
| large_70 | 1.0027 | 1.0021 | 1.0019 | **1.0** |
| large_71 | 1.0003 | **1.0** | **1.0** | 1.0001 |
| large_72 | 1.0217 | 1.0076 | 1.0102 | **1.0** |
| large_73 | 1.2583 | 1.2347 | 1.211 | **1.0** |
| large_74 | 1.0532 | 1.0338 | 1.026 | **1.0** |
| large_75 | 1.0714 | **1.0** | 1.0198 | 1.0405 |
| large_76 | 1.0176 | **1.0** | 1.0105 | 1.0299 |
| large_77 | 1.1229 | 1.0955 | 1.0944 | **1.0** |
| large_78 | 1.0398 | 1.0346 | 1.038 | **1.0** |
| large_79 | 1.0751 | 1.0753 | 1.0751 | **1.0** |
| large_80 | 1.0089 | 1.0062 | 1.0062 | **1.0** |
| large_81 | 1.4233 | 1.3908 | 1.4132 | **1.0** |
| large_82 | 1.1069 | 1.0953 | 1.0878 | **1.0** |
| large_83 | 3.4366 | 2.9423 | 3.2539 | **1.0** |
| large_84 | 1.8912 | 1.8921 | 1.8927 | **1.0** |
| large_85 | 1.014 | 1.0135 | 1.0169 | **1.0** |
| large_86 | 1.0375 | 1.0164 | 1.0253 | **1.0** |
| large_87 | 1.0365 | 1.0268 | 1.0289 | **1.0** |
| large_88 | 1.0581 | 1.053 | 1.0581 | **1.0** |
| large_89 | 1.1049 | 1.0839 | 1.0919 | **1.0** |
| large_90 | 1.4507 | 1.3743 | 1.3221 | **1.0** |
| large_91 | 1.0047 | **1.0** | 1.0015 | 1.0039 |
| large_92 | 1.2817 | 1.2715 | 1.2723 | **1.0** |
| large_93 | 1.0152 | 1.0081 | 1.0092 | **1.0** |
| large_94 | 2.0127 | 1.9638 | 1.9969 | **1.0** |
| large_95 | 1.2073 | 1.0594 | 1.0949 | **1.0** |
| large_96 | 1.0398 | 1.0105 | 1.0208 | **1.0** |
| large_97 | 1.1639 | 1.0961 | 1.125 | **1.0** |
| large_98 | 1.0658 | 1.0063 | 1.0094 | **1.0** |
| large_99 | 1.0719 | 1.0696 | 1.067 | **1.0** |
| large_100 | 1.1297 | **1.0** | 1.0005 | 1.0205 |

# Overview of Generative AI Tools Used

The tool Grammarly [1] was used throughout the thesis to correct grammar and spelling mistakes.

[1] https://www.grammarly.com/

# List of Figures

# List of Tables

# List of Algorithms

# Acronyms

**BACP** Balanced Academic Curriculum Problem. 9

**BPP** Bin Packing Problem. 9–11

**BTPP** Balanced Task Planning Problem. 5, 6, 9–13, 15–17, 19–21, 23, 26, 59, 93, 97

**ETDP** Employee Task Distribution Problem. 37, 38, 42, 45, 46, 50

**PLP** Production Leveling Problem. 9

**RTS** Reactive Tabu Search. 22, 24, 25, 97

**TS** Tabu Search. 19

# Bibliography

[AHA09] Mahsa Armaghan, Abolfazl T. Haghighat, and Mohammadreza Armaghan. QoS multicast routing algorithms based on Tabu Search with Elite candidate list. In *2009 International Conference on Application of Information and Communication Technologies*, pages 1–5, Baku, Azerbaijan, October 2009. IEEE.

[BT94] Roberto Battiti and Giampietro Tecchiolli. The Reactive Tabu Search. *ORSA Journal on Computing*, 6(2):126–140, May 1994.

[CDGGS12] Marco Chiarandini, Luca Di Gaspero, Stefano Gualandi, and Andrea Schaerf. The balanced academic curriculum problem revisited. *Journal of Heuristics*, 18(1):119–148, February 2012.

[CKPT17] Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

[CM01] Carlos Castro and Sebastian Manzano. Variable and Value Ordering When Solving Balanced Academic Curriculum Problems, October 2001.

[CSS$^+$] G. Chu, P.J. Stuckey, A. Schutt, T. Ehlers, G. Gange, and K. Francis. Chuffed, a lazy clause generation solver. https://github.com/chuffed/chuffed.

[DIM16] Maxence Delorme, Manuel Iori, and Silvano Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20, 2016.

[GL97] Fred Glover and Manuel Laguna. *Tabu Search*. Springer US, Boston, MA, 1997.

[Glo86] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[Gur24] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. https://www.gurobi.com, 2024.

[Har82] Juris Hartmanis. Computers and Intractability: A Guide to the Theory of NP-Completeness (Michael R. Garey and David S. Johnson). *Siam Review*, 24:90–91, 1982.

[HGQ+12] Simon Hamel, Jonathan Gaudreault, Claude-Guy Quimper, Mathieu Bouchard, and Philippe Marier. Human-machine interaction for real-time linear optimization. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 673–680, October 2012.

[HLM+24a] Matthias Horn, Marie-Louise Lackner, Christoph Mrkvicka, Nysret Musliu, Jakob Preininger, and Felix Winter. Benchmark instances, models and technical appendix for the Employee Task Distribution Problem. https://owncloud.tuwien.ac.at/index.php/s/JltgU8OiozNaTKf, November 2024.

[HLM+24b] Matthias Horn, Marie-Louise Lackner, Christoph Mrkvicka, Nysret Musliu, Jakob Preininger, and Felix Winter. Solving the Employee Task Distribution Problem with Multiple Objectives. In *Proceedings of the 14th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2024*, pages 36–51, 2024.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, May 1983.

[LDPC+16] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[LMMV14] Andrea Lodi, Silvano Martello, Michele Monaci, and Daniele Vigo. Two-Dimensional Bin Packing Problems. In Vangelis Th. Paschos, editor, *Paradigms of Combinatorial Optimization*, pages 107–129. Wiley, 1 edition, July 2014.

[Lou03] R. Lougee-Heimer. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, January 2003.

[MCP24] MCP GmbH. MCP Workforce Management. https://www.algorithm-factory.com, December 2024.

[ME21] Chanaleä Munien and Absalom E. Ezugwu. Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications. *Journal of Intelligent Systems*, 30(1):636–663, April 2021.

[MKF$^+$15] David Meignan, Sigrid Knust, Jean-Marc Frayret, Gilles Pesant, and Nicolas Gaud. A Review and Taxonomy of Interactive Optimization Methods in Operations Research. *ACM Transactions on Interactive Intelligent Systems*, 5(3):17:1–17:43, September 2015.

[MSZ$^+$07] Jean-Noël Monette, Pierre Schaus, Stéphane Zampelli, Yves Deville, and Pierre Dupont. A CP approach to the balanced academic curriculum problem. In *Seventh International Workshop on Symmetry and Constraint Satisfaction Problems*, volume 7, 2007.

[NSB$^+$07] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. MiniZinc: Towards a Standard CP Modelling Language. In Christian Bessière, editor, *Principles and Practice of Constraint Programming – CP 2007*, volume 4741, pages 529–543. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[PD24] Laurent Perron and Frédéric Didier. CP-SAT. Google, May 2024.

[PMSB14] Paola Pellegrini, Franco Mascia, Thomas Stützle, and Mauro Birattari. On the sensitivity of reactive tabu search to its meta-parameters. *Soft Computing*, 18(11):2177–2190, November 2014.

[Vas19] Johannes Vass. *Exact and Metaheuristic Approaches for the Production Leveling Problem.* Thesis, 2019.

[VLM$^+$22] Johannes Vass, Marie-Louise Lackner, Christoph Mrkvicka, Nysret Musliu, and Felix Winter. Exact and meta-heuristic approaches for the production leveling problem. *Journal of Scheduling*, 25(3):339–370, June 2022.