Models and Matheuristics for Large-Scale Combinatorial Optimization Problems

INAUGURALDISSERTATION

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

Mario Gnägi

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

Bern, Mai 2020

Die Fakultät hat diese Arbeit am 17. September 2020 auf Antrag der beiden Gutachter Prof. Dr. Dolores Romero Morales und Prof. Dr. Norbert Trautmann als Dissertation angenommen, ohne damit zu den darin ausgesprochenen Auffassungen Stellung nehmen zu wollen.

Contents

Introduction

Combinatorial optimization deals with efficiently determining an optimal (or at least a good) decision among a finite set of alternatives. In business administration, such combinatorial optimization problems arise in, e.g., portfolio selection, project management, data analysis, and logistics. These optimization problems have in common that the set of alternatives becomes very large as the problem size increases, and therefore an exhaustive search of all alternatives may require a prohibitively long computation time. Moreover, due to their combinatorial nature no closed-form solutions to these problems exist.

In practice, a common approach to tackle combinatorial optimization problems is to formulate them as mathematical models and to solve them using a mathematical programming solver (cf., e.g., Bixby et al. 1999; Achterberg et al. 2020). For small-scale problem instances, the mathematical models comprise a manageable number of variables and constraints such that mathematical programming solvers are able to devise optimal solutions within a reasonable computation time. For large-scale problem instances, the number of variables and constraints becomes very large which extends the computation time required to find an optimal solution considerably. Therefore, despite the continuously improving performance of mathematical programming solvers and computing hardware, the availability of mathematical models that are efficient in terms of the number of variables and constraints used is of crucial importance. Another frequently used approach to address combinatorial optimization problems are matheuristics. Matheuristics decompose the considered optimization problem into subproblems, which are then formulated as mathematical models and solved with the help of a mathematical programming solver. Matheuristics are particularly suitable for situations where it is required to find a good, but not necessarily an optimal solution within a short computation time, since the speed of the solution process can be controlled by choosing an appropriate size of the subproblems.

This thesis consists of three papers on large-scale combinatorial optimization problems. We consider a portfolio optimization problem in finance, a scheduling problem in project management, and a clustering problem in data analysis. For these problems, we present novel mathematical models that require a relatively small number of variables and

constraints, and we develop matheuristics that are based on novel problem-decomposition strategies. In extensive computational experiments, the proposed models and matheuristics performed favorably compared to state-of-the-art models and solution approaches from the literature.

In the first paper, we consider the problem of determining a portfolio for an enhanced index-tracking fund. Enhanced index-tracking funds aim to replicate the returns of a particular financial stock-market index as closely as possible while outperforming that index by a small positive excess return. Additionally, we consider various real-life constraints that may be imposed by investors, stock exchanges, or investment guidelines. Since enhanced index-tracking funds are particularly attractive to investors if the index comprises a large number of stocks and thus is well diversified, it is of particular interest to tackle large-scale problem instances. For this problem, we present two matheuristics that consist of a novel construction matheuristic, and two different improvement matheuristics that are based on the concepts of local branching (cf. Fischetti and Lodi 2003) and iterated greedy heuristics (cf., e.g., Ruiz and Stützle 2007). Moreover, both matheuristics are based on a novel mathematical model for which we provide insights that allow to remove numerous redundant variables and constraints. We tested both matheuristics in a computational experiment on problem instances that are based on large stock-market indices with up to 9,427 constituents. It turns out that our matheuristics yield better portfolios than benchmark approaches in terms of out-of-sample risk-return characteristics.

In the second paper, we consider the problem of scheduling a set of precedence-related project activities, each of which requiring some time and scarce resources during their execution. For each activity, alternative execution modes are given, which differ in the duration and the resource requirements of the activity. Sought is a start time and an execution mode for each activity, such that all precedence relationships are respected, the required amount of each resource does not exceed its prescribed capacity, and the project makespan is minimized. For this problem, we present two novel mathematical models, in which the number of variables remains constant when the range of the activities' durations and thus also the planning horizon is increased. Moreover, we enhance the performance of the proposed mathematical models by eliminating some symmetric solutions from the search space and by adding some redundant sequencing constraints for activities that cannot be processed in parallel. In a computational experiment based on instances consisting of activities with durations ranging from one up to 260 time units, the proposed models consistently outperformed all reference models from the literature.

In the third paper, we consider the problem of grouping similar objects into clusters, where the similarity between a pair of objects is determined by a distance measure

based on some features of the objects. In addition, we consider constraints that impose a maximum capacity for the clusters, since the size of the clusters is often restricted in practical clustering applications. Furthermore, practical clustering applications are often characterized by a very large number of objects to be clustered. For this reason, we present a matheuristic based on novel problem-decomposition strategies that are specifically designed for large-scale problem instances. The proposed matheuristic comprises two phases. In the first phase, we decompose the considered problem into a series of generalized assignment problems, and in the second phase, we decompose the problem into subproblems that comprise groups of clusters only. In a computational experiment, we tested the proposed matheuristic on problem instances with up to 498,378 objects. The proposed matheuristic consistently outperformed the state-of-the-art approach on medium- and large-scale instances, while matching the performance for small-scale instances.

Although we considered three specific optimization problems in this thesis, the proposed models and matheuristics can be adapted to related optimization problems with only minor modifications. Examples for such related optimization problems are the UCITS-constrained index-tracking problem (cf, e.g., Strub and Trautmann 2019), which consists of determining the portfolio of an investment fund that must comply with regulatory restrictions imposed by the European Union, the multi-site resource-constrained project scheduling problem (cf., e.g., Laurent et al. 2017), which comprises the scheduling of a set of project activities that can be executed at alternative sites, or constrained clustering problems with must-link and cannot-link constraints (cf., e.g., González-Almagro et al. 2020).

Bibliography

Achterberg, T., Bixby, R. E., Gu, Z., Rothberg, E., Weninger, D., 2020. Presolve reductions in mixed integer programming. INFORMS Journal on Computing 32 (2), 473–506.

Bixby, E. R., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R., 1999. MIP: Theory and practice - closing the gap. In: Powell, M., Scholtes, S. (Eds.), IFIP Conference on System Modeling and Optimization. Cambridge, UK, pp. 19–49.

Fischetti, M., Lodi, A., 2003. Local branching. Mathematical Programming 98 (1–3), 23–47.

González-Almagro, G., Luengo, J., Cano, J.-R., García, S., 2020. DILS: constrained clustering through dual iterative local search. Computers & Operations Research, 104979.

Laurent, A., Deroussi, L., Grangeon, N., Norre, S., 2017. A new extension of the RCPSP in a multi-site context: Mathematical model and metaheuristics. Computers & Industrial Engineering 112, 634–644.

Ruiz, R., Stützle, T., 2007. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 177, 2033–2049.

Strub, O., Trautmann, N., 2019. A two-stage approach to the UCITS-constrained index-tracking problem. Computers & Operations Research 103, 167–183.

Paper I

Tracking and outperforming large stock-market indices[1]

Mario Gnägi Oliver Strub

Department of Business Administration
University of Bern

Contents

[1]Published in Omega 90, 101999 (DOI:10.1016/j.omega.2018.11.008)

Abstract

Enhanced index-tracking funds aim to track the returns of a given financial benchmark index as closely as possible while outperforming that index by a small positive excess return. These funds are attractive to investors, especially when the index is large and thus well diversified. We consider the problem of determining a portfolio for an enhanced index-tracking fund that is benchmarked against a large stock-market index subject to real-life constraints that may be imposed by investors, stock exchanges, or investment guidelines. Existing approaches to enhanced index tracking exhibit one of the following shortcomings: they may not exploit information about the weights of the stocks in the index, they may neglect real-life constraints such as the minimum trading values imposed by stock exchanges, or they may not devise good feasible portfolios within a reasonable computational time when the index is large. To overcome these shortcomings, we present two matheuristic approaches based on a novel mixed-integer quadratic programming formulation. We tested both matheuristics on a novel set of problem instances based on large stock-market indices with up to more than 9,000 *constituents. Our computational results indicate that within a limited computational time, both matheuristics yield better feasible portfolios than benchmark approaches in terms of the objective function value and out-of-sample risk-return characteristics.*

1.1 Introduction

A stock-market index reflects the overall development of the stocks that constitute that index. Examples of such indices include the Standard & Poor's 500 index, the EURO STOXX 50 index, and the Thomson Reuters Global index, which reflect the development of national, regional, and global stock markets, respectively. Stock-market indices serve as benchmarks for evaluating the performance of professional managers of both active and passive investment funds. A passive fund, also known as an index-tracking fund, aims to replicate the return of an index, whereas an active fund aims to achieve an excess return over its benchmark index. Passive funds tend to be less risky and incur lower management costs than active funds (cf. Beasley et al. 2003). However, active funds have a higher potential return. Recently, a new type of investment fund has emerged, so-called enhanced index-tracking funds, which are based on the idea of combining the advantages

of both active and passive funds by aiming at a small target excess return with minimum additional risk relative to the index, i.e., a minimum tracking error (cf. Filippi et al. 2016). Note that we regard index-tracking funds as a special type of enhanced index-tracking funds with a target excess return of zero. Enhanced index-tracking funds are attractive to investors, especially when such a fund is benchmarked against an index that has a large number of constituents and thus is well diversified.

We consider the enhanced index-tracking problem (EITP) faced by the portfolio manager of an enhanced index-tracking fund that is benchmarked against a large stock-market index. In the EITP, the portfolio manager is given the current composition of the index and the current composition of the portfolio, which can consist of stocks from the index and cash. The portfolio manager can receive cash deposits and cash withdrawal requests. The available investment budget consists of the net cash flow from deposits and withdrawals plus the value of the current portfolio. Furthermore, the portfolio manager is given the following data from the past, i.e., the in-sample period: the values of the index, the prices of the stocks that currently constitute the index, and the interest rates on cash. The portfolio manager needs to decide how to revise (rebalance) the current portfolio such that the rebalanced portfolio will exhibit a small tracking error and achieve a given target excess return in the future, i.e., the out-of-sample period. Because future outcomes are not known in advance, the portfolio manager aims to minimize the expected tracking error subject to a constraint that prescribes some minimum expected excess return. When rebalancing the portfolio, the manager must consider a budget constraint that ensures that the investment in the stocks plus the total transaction costs spent for rebalancing do not exceed the investment budget. Furthermore, the portfolio manager must also consider various real-life constraints that may be imposed by investment guidelines, the investors, or stock exchanges. Specifically, we consider the following real-life constraints, which are common both in the literature and in practice (cf., e.g., Filippi et al. 2016; Guastaroba and Speranza 2012; Strub and Baumann 2018). The number of stocks included in the portfolio, i.e., the portfolio cardinality, must not exceed a given upper bound because investing in all constituents of a large index would be impractical due to the consequent prohibitive management costs. The trading value of each traded stock and the weight of each stock in the portfolio must be within given ranges. The total proportional and fixed transaction costs spent for rebalancing must not exceed a given fraction of the investment budget. Finally, the short selling of stocks is prohibited, and it is assumed that fractional units of stocks can be traded. Note that the EITP also includes the construction of a new portfolio as a special case when the portfolio before rebalancing consists only of cash.

In the literature, various mathematical programming formulations have been proposed

for the problem of determining an enhanced index-tracking portfolio. These formulations differ with respect to the real-life constraints considered, the way the expected tracking error is attempted to be minimized, and whether and how the expected excess return is integrated. With respect to the real-life constraints, some authors have determined enhanced index-tracking portfolios without considering real-life constraints (cf., e.g., Roll 1992), whereas others have considered all real-life constraints as defined in the EITP (cf., e.g., Strub and Baumann 2018). With respect to the expected tracking error, the earliest studies attempted to minimize the tracking error variance (TEV), which is a quadratic function of the covariances between the returns of the stocks, the weights of the stocks in the portfolio, and the weights of the stocks in the index (cf., e.g., Roll 1992). Minimizing the TEV corresponds to minimizing the estimated variance of the return differences between the portfolio and the index in the out-of-sample period (cf. Mutunge and Haugland 2018). By contrast, later studies attempted to minimize the expected tracking error by using as the objective function a dissimilarity function that captured the deviation between the historical developments of the portfolio and the index. One of the most widely used dissimilarity functions is the mean-absolute deviation (MAD) between the historical values of the portfolio and the index (cf., e.g., Filippi et al. 2016; Guastaroba and Speranza 2012; Konno and Wijayanayake 2001). In the most recent study, the goal of minimizing the TEV was revisited (cf. Mutunge and Haugland 2018). With respect to the expected excess return, some studies have focused on the problem of determining the portfolio for an index-tracking fund without considering the expected excess return (cf., e.g., Strub and Baumann 2018). In other studies, the expected excess return has been considered by using a bi-objective approach with the maximization of the expected excess return as a second competing objective (cf., e.g., Filippi et al. 2016), by introducing into the objective function a second term that captures the expected excess return (cf., e.g., Beasley et al. 2003), or by introducing a constraint that prescribes a minimum expected excess return (cf., e.g., Roll 1992). From an optimization point of view, these various means of integrating the expected excess return are very similar because all functions used for the expected excess return are linear. Various exact approaches, such as mixed-integer programming, and metaheuristic approaches, such as population-based heuristics or local-search heuristics, have all been proposed as solution approaches for the problem of determining an enhanced index-tracking portfolio.

We have identified four gaps in the literature on enhanced index tracking. Gap 1: it remains an open question whether it is preferable in terms of the out-of-sample tracking error to use the TEV as the objective function, which, together with the real-life constraints, constitutes a cardinality-constrained quadratic optimization problem that is

known to be very challenging to solve (cf. Bertsimas and Shioda 2009; Wu et al. 2017), or whether it is preferable to use a dissimilarity function such as the MAD, which can be formulated as a linear objective function and thus is less challenging to optimize. Gap 2: the EITP as defined above has not been previously considered because the problems studied in the literature may neglect the minimum expected excess return or some of the real-life constraints. Hence, the EITP as defined above has not been formulated as a mathematical program. Moreover, the existing mathematical programming formulations for problems related to the EITP that consider transaction costs allow the implicit holding of cash because the budget constraint is modeled as an inequality or because the modeled transaction costs correspond to merely an upper bound on the true transaction costs. Consequently, these cash holdings are not considered in the formulation of the expected tracking error and the expected excess return. Gap 3: the existing solution approaches for the related problems studied in the literature may not be appropriate for the EITP when the TEV is used as the objective function. The existing exact approaches would require the solution of a series of quadratic programming relaxations, which may become computationally very expensive when large indices are considered. The existing metaheuristic approaches would require adaptation to the real-life constraints of the EITP, which may reduce their effectiveness because they are tailored for other specific problems that are less constrained. Gap 4: there are no available instances of the EITP based on large stock-market indices; the existing instances of related problems either are based on small indices or do not provide information about the index composition.

The main contribution of this paper is to address the open question corresponding to gap 1 by providing novel theoretical arguments and novel experimental results. The theoretical arguments indicate that minimizing the TEV instead of a dissimilarity function may lead to superior out-of-sample tracking errors, especially when the index is large, because dissimilarity functions may not exploit the known index composition. To be able to provide experimental results, we first had to address the gaps 2 to 4. To address gap 2, we present a novel mixed-integer quadratic programming (MIQP) formulation and a novel mixed-integer linear programming (MILP) formulation of the EITP. In the MIQP formulation, we use the TEV as the objective function. In the MILP formulation, we use the MAD as the objective function. The novelties in these formulations are a formulation of the considered real-life constraints in which cash holdings are explicitly considered and insights that allow to remove redundant variables and constraints. To address gap 3, we present a construction matheuristic and two improvement matheuristics based on the proposed MIQP formulation that are able to determine good feasible portfolios, i.e., portfolios that satisfy all considered constraints, within a reasonable computational time

for the EITP when the TEV is used as the objective function, especially for instances based on large indices. The construction heuristic, which can be used to find an initial feasible portfolio quickly, is based on a novel idea of linearizing the TEV by using the identity matrix as a simplified covariance matrix and by considering absolute instead of squared deviations in the terms of the resulting function. The first improvement heuristic is based on the concept of local branching, which has been successfully applied to various combinatorial optimization problems (cf. Fischetti and Lodi 2003). In local branching, starting from the initial feasible solution, the solution space to be searched is iteratively defined with an upper bound on the number of binary variables whose values flip. The novelty of this improvement heuristic is that we consider a subset of promising stocks that differs in each iteration to reduce the required computational time. The second improvement heuristic is based on the concept of iterated greedy heuristics (cf., e.g., Ruiz and Stützle 2007). In iterated greedy heuristics, a current feasible solution is iteratively deconstructed and subsequently reconstructed in a greedy manner to form a new feasible solution. The novelties of this improvement heuristic are that we also consider a different subset of promising stocks in each iteration and that, in contrast to existing iterated greedy heuristics (cf., e.g., Strub and Trautmann 2016), we apply mixed-integer quadratic programming for the reconstruction. The proposed matheuristics are particularly suitable for the EITP because they are simple to implement and because they combine the flexibility of mathematical programming to easily incorporate complex constraints such as the considered real-life constraints with the ability of heuristics to find good feasible solutions quickly. Hence, the proposed matheuristics exhibit the properties of accuracy, speed, simplicity, and flexibility, which are the four essential attributes of good heuristics according to Cordeau et al. (2002). Finally, to address gap 4, we generated a set of novel instances of the EITP based on nine large regional and global real-world stock-market indices maintained by Thomson Reuters. The largest of these indices has more than 9,000 constituents. In a computational experiment based on these instances, we tested two heuristic solution approaches that are based on the two proposed improvement matheuristics initialized with the proposed construction matheuristic and two exact solution approaches that are based on the MIQP and the MILP formulation along with a commercial mixed-integer programming solver. This computational experiment yielded the following three main findings. 1) An exact solution approach may be appropriate for the EITP when the MAD is used as the objective function, but may not be appropriate when the TEV is minimized, which indicates the potential improvements that may be achieved by applying heuristics to the EITP when the TEV is used as the objective function. 2) The proposed matheuristics are indeed able to achieve substantial improvements

in terms of the TEV compared to an exact solution approach within a limited computational time. 3) Minimizing the TEV instead of the MAD leads to superior portfolios in terms of the out-of-sample tracking error.

The remainder of this paper is organized as follows. In Section 1.2, we review the existing solution approaches in the literature for problems that are related to the EITP. In Section 1.3, we present the MIQP and the MILP formulation and provide the arguments to address gap 1 theoretically. In Section 1.4, we present the construction matheuristic and the two improvement matheuristics. In Section 1.5, we report the computational results to address gap 1 experimentally. In Section 1.6, we offer some concluding remarks and an outlook on future research.

1.2 Related literature

Various papers in the literature have studied problems that are related to the EITP. Table 1.1 lists, for each of these papers with a ✓-symbol, whether it considers the real-life constraints of the EITP mentioned above and whether the objective is index tracking (IT) or enhanced index tracking (EIT). We categorize the papers into two groups based on whether the objective function used is non-linear or linear. In the following, we describe the proposed solution approaches of both groups.

Table 1.1: Problems related to the EITP considered in the literature.

	Paper	Real-life constraints				Objective	
		Cardinality	Min./max. weights	Transaction costs	Min./max. trades	IT	EIT
Non-linear objective function	Jorion (2003)					✓	✓
	Roll (1992)					✓	✓
	Rudd (1980)		✓			✓	✓
	Konno and Wijayanayake (2001)			✓		✓	
	Chiam et al. (2013)			✓		✓	
	Gaivoronski et al. (2005)	✓		✓		✓	
	Takeda et al. (2013)	✓				✓	
	Kwiatkowski (1992)	✓				✓	
	Maringer and Oyewumi (2007)	✓				✓	
	Mutunge and Haugland (2018)	✓				✓	
	Sant'Anna et al. (2017a)	✓				✓	✓
	Sant'Anna et al. (2017b)	✓				✓	
	Andriosopoulos and Nomikos (2014)	✓	✓			✓	✓
	Jansen and Van Dijk (2002)	✓	✓			✓	
	Scozzari et al. (2013)	✓	✓			✓	
	Krink et al. (2009)	✓	✓			✓	
	Beasley et al. (2003)	✓	✓	✓		✓	
Linear objective function	Rudolf et al. (1999)					✓	
	Bruni et al. (2015)	✓	✓			✓	✓
	Guastaroba et al. (2016)	✓	✓			✓	✓
	Guastaroba and Speranza (2012)	✓	✓	✓		✓	✓
	Filippi et al. (2016)	✓	✓	✓		✓	✓
	Strub and Baumann (2018)	✓	✓	✓	✓	✓	

The first group of problems consists of those that involve the optimization of a non-linear objective function. In some papers, only indices with a small number of constituents are considered, such that exact approaches are applicable (cf. Gaivoronski et al. 2005; Jansen and Van Dijk 2002; Rudd 1980). In other papers, the real-life constraints are neglected, which allows closed-form solutions to be devised (cf. Jorion 2003; Roll 1992). In the remaining papers, metaheuristics such as evolutionary algorithms (cf. Andriosopoulos and Nomikos 2014; Chiam et al. 2013; Krink et al. 2009; Maringer and Oyewumi 2007; Sant'Anna et al. 2017a,b; Scozzari et al. 2013) or local-search heuristics (cf. Kwiatkowski 1992; Mutunge and Haugland 2018; Takeda et al. 2013) are proposed. The majority of the papers in this first group neglect most of the real-life constraints of the EITP. An exception is the paper by Beasley et al. (2003), in which the goal is to optimize the trade-off between a non-linear dissimilarity function and the expected excess return subject to a cardinality constraint, minimum and maximum weights for the stocks included in the portfolio, and a budget for proportional transaction costs. An evolutionary algorithm is presented that uses cross-over and mutation operators to combine and modify, respectively, individuals that represent feasible and infeasible solutions. The presented algorithm includes a customized procedure for determining portfolio weights, a repair operator, and a penalty term in the objective function to handle infeasible solutions.

The second group of problems consists of those that involve the optimization of a linear objective function. For these problems, exact approaches such as linear programming and MILP approaches are able to devise good feasible solutions within a reasonable computational time, even when real-life constraints and large indices are considered (cf. Bruni et al. 2015; Filippi et al. 2016; Guastaroba et al. 2016; Guastaroba and Speranza 2012; Rudolf et al. 1999; Strub and Baumann 2018). Among all these problems, those studied in the following papers are most similar to the EITP in terms of the real-life constraints considered. Strub and Baumann (2018) introduce a MILP formulation for determining the portfolio for an index-tracking fund in which a linear dissimilarity function is minimized subject to all real-life constraints of the EITP. Guastaroba and Speranza (2012) minimize the MAD between the historical values of the portfolio and the index, which is modeled as a linear dissimilarity function, subject to a budget for fixed and proportional transaction costs, minimum and maximum portfolio weights, and a cardinality constraint. They also present a heuristic called Kernel Search, which is a matheuristic that can easily handle various real-life constraints. In this heuristic, the information from the solution to the linear programming relaxation is exploited to construct different sub-problems that can be solved quickly. They also show that their heuristic can be applied for enhanced index tracking by tracking an artificial index that represents the index return plus the target

excess return. Filippi et al. (2016) aim to maximize a linear excess-return function and minimize the same linear dissimilarity function subject to the same real-life constraints as those of Guastaroba and Speranza (2012). They modify the Kernel Search heuristic such that it can be applied to the considered problem. In the MILP formulation presented by Strub and Baumann (2018), implicit cash holdings can occur because the budget constraint is modeled as an inequality, which is necessary because the total transaction costs spent for rebalancing plus the value of the portfolio may not exactly match the investment budget. In the MILP formulations proposed by Guastaroba and Speranza (2012) and Filippi et al. (2016), implicit cash holdings can occur because the modeled transaction costs correspond merely to an upper bound on the true transaction costs. A drawback of these implicit cash holdings is that they are not considered in the calculation of the historical portfolio values and thus are also ignored in the dissimilarity and excess return functions.

The existing solution approaches presented in the literature may not be appropriate for the EITP when the TEV is used as the objective function. The existing exact approaches and the Kernel Search heuristic would first require the solution of the continuous relaxation of the MIQP formulation of the EITP, which is a quadratic program that becomes computationally very expensive to solve when large indices are considered. The existing metaheuristics would require adaptation to the real-life constraints of the EITP, which may reduce their effectiveness because they are tailored for other specific problems that do not include all of the real-life constraints of the EITP. A further drawback of metaheuristics is that they may investigate many infeasible solutions and thus be ineffective.

1.3 Mixed-integer linear and quadratic programming formulations

In this section, we present the novel MIQP formulation and the novel MILP formulation of the EITP. In Subsection 1.3.1, we first present the objective functions and the constraint on the expected excess return that are used in the two mixed-integer programming (MIP) formulations. In Subsection 1.3.2, we present new arguments that using the TEV instead of a dissimilarity function as the objective function may lead to superior portfolios in terms of the out-of-sample tracking error. In Subsection 1.3.3, we introduce the formulation of the real-life constraints. In Subsection 1.3.4, we provide insights that allow to remove redundant variables and constraints from the formulation of the real-life constraints, and we present the complete MIP formulations without the removed variables and constraints.

Table 1.2 shows the nomenclature used in the MIP formulations. The set of available

assets consists of the set of index constituents $U = \{1, \ldots, n\}$ and an asset $n + 1$ that represents the explicitly modeled cash holdings. Note that in Table 1.2, the decision variables are defined only for a set of considered stocks I, with I being a subset of the set of index constituents U and a superset of the set I_s that contains the stocks that must always be included in the portfolio after rebalancing, i.e., $I_s \subseteq I \subseteq U$. Thereby, the set I_s must always contain the stocks included in the portfolio before rebalancing that cannot be sold off completely due to the minimum and maximum trading values. Then, let T be the point in time at which the EITP must be solved, P_{it} be the prices of the stocks $i \in U$ at the point in time $t \in \{1, \ldots, T\}$, and $P_{n+1,t}$ be the values of the asset that represents cash calculated as $P_{n+1,t} = P_{n+1,T} \exp(-\sum_{s=t+1}^{T} i_s)$ for $t \in \{1, \ldots, T-1\}$, with $P_{n+1,T} = 100$ and i_t for $t \in \{2, \ldots, T\}$ corresponding to the continuously compounded interest rate on the cash holdings. Furthermore, let Y_i be the number of units of the assets $i \in U \cup \{n+1\}$ in the portfolio before rebalancing, κ be the net cash flow from deposits and withdrawals, and $C = \kappa + \sum_{i \in U \cup \{n+1\}} Y_i P_{iT}$ be the investment budget. Then, the stocks that cannot be sold off completely are those that have a value in the portfolio before rebalancing of $P_{iT}Y_i$ that is greater than the maximum trading value of $\eta_i C$ or greater than zero but smaller than the minimum trading value of $\zeta_i C$, i.e., $I_s \supseteq \{i \in U : P_{iT}Y_i > \eta_i C \vee 0 < P_{iT}Y_i < \zeta_i C\}$. We define the MIP formulations in this general form based on the sets I and I_s because this simplifies the notation for the MIP formulations without the removed redundant variables and constraints presented in Subsection 1.3.4 and because we can then use the MIQP formulation with only minor modifications for the heuristic solution approaches presented in Section 1.4.

1.3.1 Objective functions and the constraint on the expected excess return

The two competing objectives in enhanced index tracking are the minimization of the expected tracking error and the maximization of the expected excess return. In this subsection, we present the functions used to model these objectives in the proposed MIP formulations. In the MIQP and the MILP formulation, we use the TEV and the MAD, respectively, for the expected tracking error. In both formulations, we use the function presented by Roll (1992) for the expected excess return. We adjust all functions to account for the set of considered stocks I and the explicitly modeled cash holdings.

We define $X_i \geq 0$ to be the main decision variables that correspond to the number of units of the assets $i \in I \cup \{n + 1\}$ in the portfolio after rebalancing. Then, the TEV, which is used in the MIQP formulation, is a function of the covariances σ_{ij} between the

Table 1.2: Nomenclature for the MIP formulations.

Sets and parameters:	
T	Point in time at which the EITP must be solved (today)
n	Number of stocks in the index
U	Set of index constituents ($U = \{1, \ldots, n\}$)
I	Set of considered stocks ($I \subseteq U$)
I_s	Set of stocks that must be included in the portfolio after rebalancing ($I_s \subseteq I$)
k	Maximum portfolio cardinality
κ	Net cash flow from deposits and withdrawals
i_t	Continuously compounded interest rate on cash for the period starting at $t-1$ and ending at t, $t \in \{2, \ldots, T\}$
I_t/P_{it}	Historical value/price of index/asset $i \in U \cup \{n+1\}$ at $t \in \{1, \ldots, T\}$
Y_i	Number of units of asset $i \in U \cup \{n+1\}$ in the portfolio before rebalancing
C	Investment budget
ζ_i/η_i	Minimum/maximum trading value of stock $i \in U$ if traded, expressed as a percentage of C
ε_i/δ_i	Minimum/maximum weight of stock $i \in U$ if included in the portfolio after rebalancing
c_i^f	Fixed transaction cost for trading stock $i \in U$
c_i^b/c_i^s	Proportional transaction cost for buying/selling stock $i \in U$ as a percentage of the trading value
γ	Maximum total transaction costs, expressed as a percentage of C
w_i^I	Weight of asset $i \in U \cup \{n+1\}$ in the index, with $w_{n+1}^I = 0$
$\overline{r}_i$	Expected return of asset $i \in U \cup \{n+1\}$
α	Prescribed minimum expected excess return
σ_{ij}	Covariance between the discrete returns of asset $i \in U \cup \{n+1\}$ and asset $j \in U \cup \{n+1\}$
Continuous non-negative decision variables:	
X_i	Number of units of asset $i \in I \cup \{n+1\}$ in the portfolio after rebalancing
G_i	Total transaction costs associated with stock $i \in I$
v_i^b/v_i^s	Value bought/sold of stock $i \in I$
u_t/d_t	Absolute upside/downside deviation between the values of the portfolio and the index at $t \in \{1, \ldots, T\}$
Binary decision variables:	
z_i	$= 1$, if $X_i > 0$; $= 0$, otherwise ($i \in I$)
z_i^b	$= 1$, if $X_i > Y_i$; $= 0$, otherwise ($i \in I$)
z_i^s	$= 1$, if $X_i < Y_i$; $= 0$, otherwise ($i \in I$)

returns of assets $i \in U \cup \{n+1\}$ and $j \in U \cup \{n+1\}$, the weights $\frac{P_{iT}X_i}{C}$ of the assets $i \in U \cup \{n+1\}$ in the portfolio, and the weights w_i^I of the assets $i \in U \cup \{n+1\}$ in the index, with $w_{n+1}^I = 0$. Any stock that is not included in I will have a portfolio weight of zero. Thus, the following function represents the TEV:

$$\sum_{i,j \in I \cup \{n+1\}} \sigma_{ij} \left(\frac{P_{iT}X_i}{C} - w_i^I \right) \left(\frac{P_{jT}X_j}{C} - w_j^I \right) - 2 \sum_{i \in I \cup \{n+1\}} \sum_{j \in U \setminus I} \sigma_{ij} \left(\frac{P_{iT}X_i}{C} w_j^I - w_i^I w_j^I \right) + \sum_{i,j \in U \setminus I} \sigma_{ij} w_i^I w_j^I \tag{1.1}$$

Based on the expected returns $\overline{r}_i$ of the assets $i \in U \cup \{n+1\}$, the expected excess return is calculated as the difference between the expected return of the portfolio and the expected return of the index:

$$\sum_{i \in I \cup \{n+1\}} \frac{P_{iT}X_i}{C} \overline{r}_i - \sum_{i \in U \cup \{n+1\}} w_i^I \overline{r}_i \tag{1.2}$$

In the MIQP formulation, we minimize the TEV subject to a constraint that prescribes a minimum expected excess return of α, as follows:

$$\begin{cases} \text{Min. } (1.1) & (1.3) \\ \text{s.t.} \quad \displaystyle\sum_{i \in I \cup \{n+1\}} \frac{P_{iT}X_i}{C} \overline{r}_i - \sum_{i \in U \cup \{n+1\}} w_i^I \overline{r}_i \geq \alpha & (1.4) \end{cases}$$

In the MILP formulation, the dissimilarity function captures the MAD over all in-sample time points $t \in \{1, \ldots, T\}$ between the values of the index I_t, scaled to the investment budget C at time point T, and the values of the portfolio $\sum_{i \in I \cup \{n+1\}} P_{it}X_i$. With the introduction of the non-negative decision variables u_t and d_t for $t \in \{1, \ldots, T\}$, the MAD can be minimized subject to the constraint on the expected excess return as follows:

$$\begin{cases} \text{Min. } \dfrac{1}{T} \displaystyle\sum_{t \in \{1,\ldots,T\}} (u_t + d_t) & (1.5) \\ \text{s.t.} \quad u_t - d_t = \displaystyle\sum_{i \in I \cup \{n+1\}} P_{it}X_i - I_t \frac{C}{I_T} \qquad (t \in \{1, \ldots, T\}) & (1.6) \\ \displaystyle\sum_{i \in I \cup \{n+1\}} \frac{P_{iT}X_i}{C} \overline{r}_i - \sum_{i \in U \cup \{n+1\}} w_i^I \overline{r}_i \geq \alpha & (1.4) \end{cases}$$

1.3.2 TEV: a comparison with dissimilarity functions

In this subsection, we compare the minimization of the TEV with the minimization of a dissimilarity function such as the MAD in terms of the out-of-sample tracking error. For this purpose, we consider all index constituents, i.e., $I = U$, and we consider only a budget constraint that ensures that the entire investment budget is invested in the assets. We assume that the number of available assets is much larger than the number of in-sample time points, i.e., $|U \cup \{n+1\}| \gg T$, and that the matrix consisting of the in-sample prices of each stock, where each stock corresponds to a column, has full row rank. Both assumptions are usually satisfied when the index is large. We further assume that the matrix of the covariances is positive definite. This assumption is satisfied when an appropriate estimator is used for the covariances, such as that of Ledoit and Wolf (2004b), but may be violated when the sample covariance is used as an estimator (cf. Ledoit and Wolf 2004a).

When the TEV is to be minimized with a positive-definite matrix of covariances, the only solution with zero TEV is the portfolio that has the same composition as the index. By contrast, when a dissimilarity function is used, i.e., when the known index composition is ignored, infinitely many different portfolios can exist that achieve a dissimilarity of zero with respect to the index. To see this, note that finding a portfolio with zero dissimilarity is equivalent to solving a system of T linear equations with $n+1$ unknowns, where these T equations state that the portfolio value at each time point $t \in \{1, \ldots, T\}$ must match the scaled index value at that time point. Note that the equation for time point T also ensures that the budget constraint is satisfied because of the scaling of the index values. Under the assumption of a full row rank matrix of stock prices, infinitely many solutions to this linear system exist, which means that infinitely many portfolios with zero dissimilarity exist.

Based on the arguments above, one drawback of minimizing a dissimilarity function is that, in contrast to the case of minimizing the TEV, many different portfolios can exist that each have an objective function value of zero but a composition that strongly differs from that of the index. These portfolios may have very high out-of-sample tracking errors. Hence, for our computational experiment reported in Section 1.5, we expect that the compositions of the portfolios obtained when minimizing the MAD will differ more strongly from the composition of the index than the compositions of the portfolios obtained when minimizing the TEV. Consequently, we also expect that, over all considered problem instances, the average and the worst-case tracking error for the out-of-sample period will be worse when the MAD is minimized instead of the TEV.

1.3.3 Real-life constraints

Next, we model the real-life constraints. The constraints expressed in (1.7) assign at least the absolute value bought or sold of each stock $i \in I$ to the non-negative decision variable v_i^b or v_i^s, respectively. These decision variables are used to model the transaction costs and the minimum and maximum trading values.

$$v_i^b - v_i^s = P_{iT}(X_i - Y_i) \qquad (i \in I) \tag{1.7}$$

The purpose of constraints (1.8) and (1.9) is twofold. First, the binary variables z_i^b and z_i^s are assigned a value of one if the variables v_i^b and v_i^s, respectively, take a positive value and a value of zero otherwise. Second, the constraints prescribe minimum and maximum values of $\zeta_i C$ and $\eta_i C$, respectively, for v_i^b and v_i^s.

$$\zeta_i C z_i^b \leq v_i^b \leq \eta_i C z_i^b \qquad (i \in I) \tag{1.8}$$

$$\zeta_i C z_i^s \leq v_i^s \leq \eta_i C z_i^s \qquad (i \in I) \tag{1.9}$$

The constraints defined in (1.10) ensure that for each stock $i \in I$, at most one of the binary variables z_i^b and z_i^s can be set to one.

$$z_i^b + z_i^s \leq 1 \qquad (i \in I) \tag{1.10}$$

Together, constraints (1.8), (1.9), and (1.10) ensure that for each stock $i \in I$, either v_i^b or v_i^s must be set to zero. Because it is not possible for both variables v_i^b and v_i^s to take positive values simultaneously for a given stock $i \in I$, the constraints defined in (1.7) assign the actual values bought or sold of each stock $i \in I$ to the variables v_i^b or v_i^s, respectively. These actual values are necessary to model the minimum and maximum trading values $\zeta_i C$ and $\eta_i C$ using constraints (1.8) and (1.9).

Let c_i^f, c_i^b, and c_i^s be the parameters that determine the fixed transaction costs for trading the stocks $i \in U$, the proportional transaction costs for buying units of the stocks $i \in U$, and the proportional transaction costs for selling units of the stocks $i \in U$, respectively. Then, based on the variables v_i^b, v_i^s, z_i^b, and z_i^s, the transaction costs G_i for each stock $i \in I$ are calculated using the constraints defined in (1.11). Note that the variables G_i take values equal to the actual transaction costs associated with each stock $i \in I$, because we ensure that the variables v_i^b and v_i^s take the actual values bought and sold of each stock, and that at most one of the binary variables z_i^b and z_i^s can be set to

one if stock $i \in I$ is traded, whereas both variables z_i^b and z_i^s are set to zero otherwise.

$$G_i = c_i^b v_i^b + c_i^s v_i^s + c_i^f (z_i^b + z_i^s) \qquad (i \in I) \tag{1.11}$$

The budget constraint (1.12) states that the available investment budget C must be either held in cash, invested in the stocks that constitute the index, or spent for transaction costs. Note the possibility that some stocks were included in the portfolio before rebalancing but are not included in the set of considered stocks I. Hence, the shares of these stocks must be sold, incurring total transaction costs of $\sum_{i \in U \setminus I : Y_i > 0} \left(c_i^s Y_i P_{iT} + c_i^f \right)$. Since the variables G_i take values equal to the actual transaction costs associated with each stock $i \in I$, constraint (1.12) ensures that the variable X_{n+1} corresponds exactly to the part of the investment budget that is not invested in stocks or spent for transaction costs. Hence, we can explicitly account for these cash holdings when formulating the TEV, the MAD, and the expected excess return.

$$\sum_{i \in I \cup \{n+1\}} P_{iT} X_i + \sum_{i \in I} G_i + \sum_{i \in U \setminus I : Y_i > 0} \left(c_i^s Y_i P_{iT} + c_i^f \right) = C \tag{1.12}$$

Constraint (1.13) prescribes a budget of γC for the total transaction costs.

$$\sum_{i \in I} G_i + \sum_{i \in U \setminus I : Y_i > 0} \left(c_i^s Y_i P_{iT} + c_i^f \right) \leq \gamma C \tag{1.13}$$

The constraints (1.14) ensure that each binary variable z_i takes a value of one if stock $i \in I$ is included in the portfolio after rebalancing and a value of zero otherwise. Furthermore, these constraints define minimum and maximum values of ε_i and δ_i, respectively, for the weight of each stock $i \in I$ in the portfolio.

$$\varepsilon_i z_i \leq \frac{P_{iT} X_i}{C} \leq \delta_i z_i \qquad (i \in I) \tag{1.14}$$

Based on the binary variables z_i, the cardinality constraint (1.15) is formulated as follows.

$$\sum_{i \in I} z_i \leq k \tag{1.15}$$

The domains of the decision variables are specified by (1.16) and (1.17).

$$X_i \geq 0 \qquad (i \in I \cup \{n+1\}) \tag{1.16}$$

$$z_i^b, z_i^s, z_i \in \{0, 1\};\ v_i^b, v_i^s, G_i \geq 0 \qquad (i \in I) \tag{1.17}$$

1.3.4 Removing redundant variables and constraints from the mixed-integer programming formulations

From the formulation of the real-life constraints presented in Subsection 1.3.3, we can remove some redundant variables and constraints based on the following three insights. First, we note that some stocks that are included in the portfolio before rebalancing must always be included in the portfolio after rebalancing due to the specified minimum and maximum trading values. As mentioned above, these stocks are included in the set $I_s \supseteq \{i \in U : P_{iT}Y_i > \eta_i C \vee 0 < P_{iT}Y_i < \zeta_i C\}$. Second, Filippi et al. (2016) note that if a stock is not included in the portfolio before rebalancing and is traded, then this stock will always be included in the portfolio after rebalancing. Third, Strub and Baumann (2018) note that stocks that are not included in the portfolio before rebalancing cannot be sold because short selling is not allowed. We combine all three insights to obtain the following restrictions on stocks based on their values in the portfolio before rebalancing. For each stock i that is not included in the portfolio before rebalancing, i.e., $Y_i = 0$, selling stock i is not possible, and trading stock i means that it will be included in the portfolio after rebalancing. For each stock i that has a value in the portfolio before rebalancing that is positive but smaller than the minimum trading value, i.e., $0 < P_{iT}Y_i < \zeta_i C$, selling stock i is not possible, and thus, stock i must be included in the portfolio after rebalancing. For each stock i that has a value in the portfolio before rebalancing that is larger than the maximum trading value, i.e., $\eta_i C < P_{iT}Y_i$, selling all units of stock i is not possible, and thus, stock i must be included in the portfolio after rebalancing.

Based on the restrictions above, we can eliminate certain variables and constraints. For each stock i that cannot be sold, the binary variable z_i^s and the continuous variable v_i^s must both be zero and thus can be removed. Additionally, the continuous variable v_i^b can be replaced with $P_{iT}(X_i - Y_i)$. Furthermore, for each stock i that is not included in the portfolio before rebalancing, i.e., $Y_i = 0$, we can replace the binary variable z_i^b with the binary variable z_i because selling is not possible and buying stock i means that it will be included in the portfolio after rebalancing. For each stock i that must be included in the portfolio after rebalancing, we can set the binary variable z_i equal to one.

The novel MIQP formulation (M-Q) and the novel MILP formulation (M-L) below include the real-life constraints (without redundant variables and constraints) along with the constraint on the expected excess return and their corresponding objective functions.

$$
\text{(M-Q)}\quad
\begin{cases}
\text{Min.} & (1.1) \\
\text{s.t.} & (1.4), (1.12), (1.13) \\
& v_i^b - v_i^s = P_{iT}(X_i - Y_i) \qquad (i \in I : P_{iT}Y_i \geq \zeta_i C) & (1.18)\\
& \zeta_i C \leq X_i P_{iT} \leq \eta_i C \qquad (i \in I_s : Y_i = 0) & (1.19)\\
& \zeta_i C z_i \leq X_i P_{iT} \leq \eta_i C z_i \qquad (i \in I \setminus I_s : Y_i = 0) & (1.20)\\
& \zeta_i C z_i^b \leq (X_i - Y_i) P_{iT} \leq \eta_i C z_i^b \qquad (i \in I : 0 < P_{iT}Y_i < \zeta_i C) & (1.21)\\
& \zeta_i C z_i^b \leq v_i^b \leq \eta_i C z_i^b \qquad (i \in I : P_{iT}Y_i \geq \zeta_i C) & (1.22)\\
& \zeta_i C z_i^s \leq v_i^s \leq \eta_i C z_i^s \qquad (i \in I : P_{iT}Y_i \geq \zeta_i C) & (1.23)\\
& z_i^b + z_i^s \leq 1 \qquad (i \in I : P_{iT}Y_i \geq \zeta_i C) & (1.24)\\
& G_i = c_i^b P_{iT} X_i + c_i^f \qquad (i \in I_s : Y_i = 0) & (1.25)\\
& G_i = c_i^b P_{iT} X_i + c_i^f z_i \qquad (i \in I \setminus I_s : Y_i = 0) & (1.26)\\
& G_i = c_i^b P_{iT}(X_i - Y_i) + c_i^f z_i^b \qquad (i \in I : 0 < P_{iT}Y_i < \zeta_i C) & (1.27)\\
& G_i = c_i^b v_i^b + c_i^s v_i^s + c_i^f (z_i^b + z_i^s) \qquad (i \in I : P_{iT}Y_i \geq \zeta_i C) & (1.28)\\
& \varepsilon_i \leq \dfrac{P_{iT}X_i}{C} \leq \delta_i \qquad (i \in I_s) & (1.29)\\
& \varepsilon_i z_i \leq \dfrac{P_{iT}X_i}{C} \leq \delta_i z_i \qquad (i \in I \setminus I_s) & (1.30)\\
& \displaystyle\sum_{i \in I \setminus I_s} z_i \leq k - |I_s| & (1.31)\\
& X_i \geq 0 \qquad (i \in I \cup \{n+1\}) & (1.32)\\
& G_i \geq 0 \qquad (i \in I) & (1.33)\\
& z_i \in \{0,1\} \qquad (i \in I \setminus I_s) & (1.34)\\
& v_i^b, v_i^s \geq 0,\ z_i^s \in \{0,1\} \qquad (i \in I : P_{iT}Y_i \geq \zeta_i C) & (1.35)\\
& z_i^b \in \{0,1\} \qquad (i \in I : Y_i > 0) & (1.36)
\end{cases}
$$

$$
\text{(M-L)}\quad
\begin{cases}
\text{Min.} & (1.5) \\
\text{s.t.} & (1.6), (1.4), (1.12), (1.13), (1.18), (1.19), \\
& (1.20), (1.21), (1.22), (1.23), (1.24), (1.25), \\
& (1.26), (1.27), (1.28), (1.29), (1.30), (1.31), \\
& (1.32), (1.33), (1.34), (1.35), (1.36) \\
& u_t, d_t \geq 0 \qquad (t \in \{1, \ldots, T\}) & (1.37)
\end{cases}
$$

Table 1.3: Additional nomenclature for the heuristic solution approaches.

Parameters:	
q	Parameter that defines the number of considered stocks
Δ	Number of stocks that can be added and removed
$\overline{\nu}$	Maximum number of iterations without improvement
d	Maximum number of stocks to be removed
Continuous non-negative decision variables:	
ξ_i	Absolute deviation between the portfolio weight and index weight of asset $i \in I \cup \{n+1\}$

1.4 Heuristic solution approaches

In this section, we present the matheuristics for the EITP when the TEV is used as the objective function. In Subsection 1.4.1, we present the construction heuristic for determining an initial feasible portfolio. In Subsections 1.4.2 and 1.4.3, we present the two improvement heuristics based on local branching and the concept of iterated greedy heuristics, respectively, for improving a given initial feasible portfolio. Table 1.3 defines the additional nomenclature used in formulating these matheuristics.

1.4.1 Construction heuristic

Constructing a feasible portfolio is not straightforward. It is possible that no feasible portfolio exists, e.g., when the prescribed minimum excess return is set too high or when a current portfolio must be rebalanced so heavily to ensure the prescribed minimum and maximum weights that the prescribed budget for transaction costs is too low. Even if a feasible portfolio exists, when selecting the stocks that should be included in the portfolio by applying, e.g., a random or greedy algorithm, it might not be possible to find weights for these selected stocks such that the portfolio is feasible with respect to all constraints.

Therefore, we propose a MILP-based construction heuristic that is able to find a feasible portfolio quickly if one exists, and can also prove the nonexistence of a feasible portfolio. To simplify the search for a feasible portfolio for the MIQP formulation (M-Q), we use the identity matrix as a simplified covariance matrix. Thus, the objective function (1.1) reduces to the following terms:

$$\sum_{i \in I \cup \{n+1\}} \left(\frac{P_{iT} X_i}{C} - w_i^I \right)^2 + \sum_{i \in U \setminus I} (w_i^I)^2 \tag{1.38}$$

Furthermore, we consider the sum of the absolute deviations (instead of the squared

deviations) between the weights of the assets in the portfolio and the weights of the assets in the index, and we ignore the second sum because it is constant. The resulting objective function can be optimized subject to the constraints of the EITP using the following MILP formulation:

$$
\text{(M-C)} \begin{cases}
\text{Min.} & \displaystyle\sum_{i \in I \cup \{n+1\}} \xi_i & & (1.39) \\
\text{s.t.} & \xi_i \geq \dfrac{P_{iT} X_i}{C} - w_i^I & (i \in I \cup \{n+1\}) & (1.40) \\
& \xi_i \geq w_i^I - \dfrac{P_{iT} X_i}{C} & (i \in I \cup \{n+1\}) & (1.41) \\
& (1.4), (1.12), (1.13), (1.18), (1.19), (1.20), & & \\
& (1.21), (1.22), (1.23), (1.24), (1.25), (1.26), & & \\
& (1.27), (1.28), (1.29), (1.30), (1.31), (1.32), & & \\
& (1.33), (1.34), (1.35), (1.36) & & \\
& \xi_i \geq 0 & (i \in I \cup \{n+1\}) & (1.42)
\end{cases}
$$

To further simplify the search for a feasible portfolio, we consider only a limited set of promising stocks I. If no feasible portfolio can be found based on a given set I, we increase the cardinality of the set I. This preselection is crucial for finding good feasible portfolios for the MILP formulation (M-C). In general, the set I should contain the stocks with the highest weights in the index to allow a small objective function value to be achieved. Moreover, the set I should contain the stocks that are in the current portfolio, because not including these stocks in the set I would mean that we were required to sell all units of these stocks, which would incur high transaction costs.

Algorithm 1.1 describes the construction heuristic. First, we initialize ν, and we include in the set I all stocks that are held in the portfolio before rebalancing and all stocks that are in the set I_s. Then, we gradually expand the set I by including the k stocks that have the highest weights in the index and are not yet included in the set I. Thereafter, based on the expanded set I, the MILP formulation (M-C) is solved. This process is repeated until a feasible portfolio is found or until $I = U$. If no feasible portfolio is found with $I = U$, this proves that no feasible portfolio exists.

1.4.2 Local branching heuristic

Local branching refers to a local-search framework for MIP formulations that is based on so-called local-branching cuts. Given a feasible solution, these local-branching cuts itera-

Algorithm 1.1 Construction heuristic

```
1: procedure ConstructionHeuristic()
2:     ν ← 0; I ← {i ∈ U : Y_i > 0} ∪ I_s;
3:     while true do
4:         while I ≠ U and |I| < |{i ∈ U : Y_i > 0} ∪ I_s| + k(1 + ν) do
5:             I ← I ∪ {a}, where a ∈ argmax_{i∈U\I} w_i^I;
6:         end while
7:         Solve (M-C);
8:         if feasible portfolio found then
9:             return set of stocks included in the feasible portfolio;
10:        else if I = U then
11:            return no feasible portfolio exists;
12:        end if
13:        ν ← ν + 1;
14:    end while
15: end procedure
```

tively define the neighborhood to be searched by placing an upper bound on the number of binary variables whose values can be flipped, either from one to zero or from zero to one. Based on this framework, we extend the MIQP formulation (M-Q) by incorporating constraints (1.43) and (1.44). Starting from the best feasible portfolio found so far, which includes the stocks in the set I^*, these constraints restrict the search space to all feasible portfolios that can be reached by adding at most Δ stocks to the portfolio and by removing at most Δ stocks from the portfolio. When removing stocks, we ensure that no stocks from the set I_s are removed because these stocks must be included in the portfolio after rebalancing. This results in the MIQP formulation (M-LBH) shown below.

$$
\text{(M-LBH)} \begin{cases}
\text{Min.} & (1.1) \\
\text{s.t.} & (1.4), (1.12), (1.13), (1.18), (1.19), (1.20), \\
 & (1.21), (1.22), (1.23), (1.24), (1.25), (1.26), \\
 & (1.27), (1.28), (1.29), (1.30), (1.31), (1.32), \\
 & (1.33), (1.34), (1.35), (1.36) \\
 & \displaystyle\sum_{i \in I^* \setminus I_s} (1 - z_i) \leq \Delta & (1.43) \\
 & \displaystyle\sum_{i \in I \setminus I^*} z_i \leq \Delta & (1.44)
\end{cases}
$$

The local branching framework requires the solution of a series of quadratic programs,

which is computationally expensive for large indices when all available stocks are considered in each iteration, i.e., when $I = U$, even when the search space is restricted by local-branching cuts. Therefore, we propose a novel approach in which the search space is further restricted by considering only a limited set of promising stocks I. To prevent the exclusion of high-quality solutions from the search space due to a poor preselection of the stocks to be included in the set I, we use a randomly selected set I in each iteration. Because we consider stocks with higher index weights to be more promising for obtaining low objective function values, we define the probability that a stock will be included in the set I in each iteration to be proportional to its weight in the index.

Algorithm 1.2 describes the local branching heuristic. First, we initialize ν and Δ. Then, we include in the set I the stocks from the set I^*, which contains the stocks that are included in the best feasible portfolio found so far. Subsequently, we iteratively expand the set I until it contains $k + q$ stocks. In a given iteration of this process, each stock $i \in U \setminus I$ has a probability $\frac{w_i^I}{\sum_{j \in U \setminus I} w_j^I}$ of being selected for inclusion in the set I. Thereafter, the MIQP formulation (M-LBH) is solved. If a better feasible portfolio is found, we update the set I^* to contain the selected stocks in this new best feasible portfolio, and we reset the number of iterations elapsed without finding a better feasible portfolio ν to zero; otherwise, we increase ν by one. If ν reaches $\overline{\nu}$, i.e., the maximum number of iterations without a better feasible portfolio having been found, we increase Δ by one to enlarge the search space. As soon as a new best feasible portfolio is found, we reset Δ to one. This process is repeated until a given termination criterion is satisfied. Finally, the best feasible portfolio found so far is returned.

1.4.3 Iterated greedy heuristic

In an iterated greedy heuristic, two phases are performed repeatedly: deconstruction and reconstruction. During the deconstruction phase, we remove several randomly selected stocks from the current best feasible portfolio. During the subsequent reconstruction phase, we add stocks back into the deconstructed portfolio in a greedy manner to obtain a new feasible portfolio.

In contrast to existing iterated greedy heuristics, we restrict the search space by considering only a limited set of promising stocks I, as in the local branching heuristic, and we repeatedly solve an MIQP formulation during the reconstruction phase to add the myopic best stock to the deconstructed portfolio, which allows all constraints in the MIQP formulation (M-Q) to be easily considered. Specifically, we solve the MIQP formulation (M-IGH) below that corresponds to the MIQP formulation (M-Q) without the cardinality constraint (1.31), but with the additional constraint (1.45). This additional constraint

Algorithm 1.2 Local branching heuristic

procedure LocalBranchingHeuristic(I^*, q, $\overline{\nu}$)
  $\nu \leftarrow 0$; $\Delta \leftarrow 1$;
  while termination criterion not satisfied **do**
    $I \leftarrow I^*$;
    while $I \neq U$ **and** $|I| < k + q$ **do**
      $a \leftarrow$ select stock from set $U \setminus I$ with probability $\frac{w_i^I}{\sum_{j \in U \setminus I} w_j^I}$ of the selection of stock $i \in U \setminus I$;
      $I \leftarrow I \cup \{a\}$;
    end while
    Solve (M-LBH) to obtain a feasible portfolio by adding at most Δ stocks and by removing at most Δ stocks;
    if new best feasible portfolio found **then**
      $I^* \leftarrow$ set of selected stocks in the new best feasible portfolio;
      $\nu \leftarrow 0$; $\Delta \leftarrow 1$;
    else
      $\nu \leftarrow \nu + 1$;
      if $\nu = \overline{\nu}$ **then**
        $\nu \leftarrow 0$; $\Delta \leftarrow \Delta + 1$;
      end if
    end if
  end while
  return best feasible portfolio found;
end procedure

prescribes that at most one stock from the set I that is not included in the set I_s can be added to the portfolio. During the execution of the iterated greedy heuristic, we modify the set I_s such that it contains the stocks that must be included in the reconstructed portfolio, i.e., the stocks that are included in the current best feasible portfolio and were not removed during the most recent deconstruction phase.

$$\text{(M-IGH)} \begin{cases} \text{Min.} & (1.1) \\ \text{s.t.} & (1.4), (1.12), (1.13), (1.18), (1.19), (1.20), \\ & (1.21), (1.22), (1.23), (1.24), (1.25), (1.26), \\ & (1.27), (1.28), (1.29), (1.30), (1.32), (1.33), \\ & (1.34), (1.35), (1.36) \\ & \displaystyle\sum_{i \in I \setminus I_s} z_i \leq 1 \end{cases} \tag{1.45}$$

Algorithm 1.3 describes the iterated greedy heuristic. First, we store the stocks that

must be included in the portfolio after rebalancing in the set I'_s because the algorithm modifies the set I_s during its execution. Then, we enter the main loop, which consists of the deconstruction, reconstruction, and acceptance phases. Before beginning the deconstruction phase, we include in the set I the stocks that are included in the best feasible portfolio found so far. Then, during the deconstruction phase, we first remove p randomly selected stocks from the set I. When removing stocks, we must ensure that no stocks from the set I'_s are removed because these stocks must be included in the portfolio after rebalancing. After having removed p stocks from the set I, we define the set of stocks that must be included in the reconstructed portfolio, i.e., the set I_s, as the set of stocks currently in the set I. Then, the reconstruction phase begins. During the reconstruction phase, we expand the set I by adding stocks based on probabilities that depend on the weights of the stocks in the index, as is done in the local branching heuristic. As soon as the set I consists of $k + q$ stocks, we iteratively solve (M-IGH) to add to the portfolio at most one new stock in each iteration from the set $I \setminus I_s$. If a feasible portfolio with one new stock from the set $I \setminus I_s$ can be found, then the newly selected stock is added to the set I_s. This process is repeated until either k stocks are included in the portfolio, no new stock is added to the portfolio, or it is found that no feasible portfolio for (M-IGH) exists. After the reconstruction phase, we check whether a new best feasible portfolio has been found. If this is the case, we update the set I^* to contain the selected stocks in the new best feasible portfolio. The deconstruction, reconstruction, and acceptance phases are repeated until a specified termination criterion is met. Finally, we reset the set I_s and return the best feasible portfolio found so far.

1.5 Computational results

In this section, we address the open question whether it is preferable in terms of the out-of-sample tracking error to use the TEV or the MAD as the objective function by providing computational results. For this purpose, we test the performance of two exact solution approaches and two heuristic solution approaches for three scenarios that differ in terms of the composition of the portfolio before rebalancing. The two exact solution approaches are based on the MIQP formulation (M-Q) and the MILP formulation (M-L); we refer to these MIP approaches as M-Q and M-L, respectively. In these MIP approaches, we consider the entire set of index constituents, i.e., $I = U$. For the two heuristic solution approaches, the construction heuristic (cf. Algorithm 1.1) is used to determine an initial feasible portfolio, and the two improvement heuristics (cf. Algorithms 1.2 and 1.3) are used to improve this initial feasible portfolio; we refer to these approaches as LBH and

Algorithm 1.3 Iterated greedy heuristic

procedure IteratedGreedyHeuristic(I^*, q, d)
  $I'_s \leftarrow I_s$;
  while termination criterion not satisfied **do**
    $I \leftarrow I^*$;
    $p \leftarrow$ random integer from set $\{1, \ldots, d\}$; ▷ Deconstruction
    for $i \leftarrow 1$ **to** p **do**
      Remove a randomly selected element from I that is not included in I'_s;
    end for
    $I_s \leftarrow I$;
    while $I \neq U$ **and** $|I| < k + q$ **do** ▷ Reconstruction
      $a \leftarrow$ select stock from set $U \setminus I$ with probability $\frac{w_i^I}{\sum_{j \in U \setminus I} w_j^I}$ of the selection of stock $i \in U \setminus I$;
      $I \leftarrow I \cup \{a\}$;
    end while
    while $|I_s| < k$ **do**
      Solve (M-IGH) to add at most one new stock to the portfolio;
      if a feasible portfolio with one new stock has been found **then**
        $I_s \leftarrow$ set of selected stocks in the feasible portfolio;
      else
        break;
      end if
    end while
    if new best feasible portfolio found **then** ▷ Acceptance
      $I^* \leftarrow$ set of selected stocks in the new best feasible portfolio;
    end if
  end while
  $I_s \leftarrow I'_s$
  return best feasible portfolio found;
end procedure

IGH, respectively.

This section is organized as follows. In Subsections 1.5.1 and 1.5.2, we explain the design of our experiment and describe the novel problem instances, respectively. In Subsection 1.5.3, we investigate the index-tracking capabilities of the portfolios given in the three scenarios before rebalancing to provide a reference against which to assess the index-tracking capabilities of the portfolios after rebalancing. In Subsection 1.5.4, we show that in contrast to M-L, M-Q leads to feasible portfolios within a limited computational time that may be improved substantially in terms of the objective function value, which indicates the potential of applying LBH and IGH. In Subsection 1.5.5, we show that LBH and IGH are indeed able to achieve substantial improvements in terms of the objective

function value compared to M-Q. In Subsection 1.5.6, we provide insights that minimizing the TEV may be superior to minimizing the MAD in terms of the out-of-sample tracking error. In Subsection 1.5.7, we offer further insights that this superior out-of-sample performance may be attributed to the different compositions of the rebalanced portfolios.

1.5.1 Experimental design

We use an experimental design similar to those of Guastaroba and Speranza (2012) and Filippi et al. (2016). We assume that the manager of an investment fund rebalances a portfolio at the end of an in-sample period that consists of 104 weeks, i.e., $T = 104$. The portfolio is then left unchanged for the entirety of an out-of-sample period that consists of 52 weeks. We also assume that the fixed transaction cost for trading is 12 for all stocks (i.e., $c_i^f = 12$ for all $i \in U$), that the proportional transaction costs for buying and selling are each 1% of the trading value for all stocks (i.e., $c_i^b = c_i^s = 0.01$ for all $i \in U$), and that the budget available for transaction costs is 1.5% of the investment budget (i.e., $\gamma = 0.015$).

Furthermore, we define three scenarios, I, II, and III, which differ in terms of the composition of the investment fund's portfolio before rebalancing. In scenarios I and II, a portfolio of stocks already exists. In scenario III, a new portfolio must be constructed from cash. In scenarios I and II, the portfolios before rebalancing consist of the k stocks with the highest and lowest weights in the index, respectively. The weight of each stock in the portfolio before rebalancing is set such that it is proportional to the weight of that stock in the index and such that the sum of the weights of all stocks in the portfolio is equal to one. The portfolio before rebalancing in scenario I is a portfolio with a rather good index-tracking capability, whereas the portfolio before rebalancing in scenario II is a portfolio with a rather poor index-tracking capability. This claim is supported by the results presented in Subsection 1.5.3.

The values of the remaining parameters deviate from the values used by Guastaroba and Speranza (2012) and Filippi et al. (2016). The reason for these deviations is that we are considering much larger indices, and thus, we wish to allow the portfolio to have a larger cardinality. Specifically, we use two different values for the maximum portfolio cardinality, namely, $k = 100$ and $k = 200$. Because of the larger portfolio cardinality, we must also allow the portfolio weights to take smaller values. For the minimum and maximum portfolio weights, we adopt values of 0.2% and 20%, respectively, for all stocks (i.e., $\varepsilon_i = 0.002$ and $\delta_i = 0.2$ for all $i \in U$). For the parameters that define the minimum and maximum trading values, we use the same values as for the parameters ε_i and δ_i for all stocks (i.e., $\zeta_i = 0.002$ and $\eta_i = 0.2$ for all $i \in U$). For scenarios I and II, we assume a

value for each portfolio of 10,000,000 and a net change in cash of $\kappa = 0$. For scenario III, we assume a portfolio value of 0 and a κ of 10,000,000; this is the same as assuming that the portfolio before rebalancing consists of cash only and that the net change in cash is zero (i.e., $Y_{n+1} = \frac{10{,}000{,}000}{P_{n+1,T}}$, $Y_i = 0$ for all $i \in U$, and $\kappa = 0$). Hence, in all three scenarios, the investment budget C is 10,000,000. Since the prescribed minimum expected excess return α and the interest rate i_t on cash were not used by Guastaroba and Speranza (2012) and Filippi et al. (2016), we define new values. Specifically, we use three α values of 0, 0.0001914, and 0.0005686, which correspond to annualized α values of 0%, 1%, and 2%, respectively, and a value of zero for the interest rate on cash at all time points (i.e., $i_t = 0$ for all $t \in \{2, \ldots, T\}$). The values of the parameters n, I_t, P_{it}, and w_i^I depend on the considered index.

We use the estimator of Ledoit and Wolf (2004b) to estimate the covariances σ_{ij} based on the discrete in-sample returns of the assets $i, j \in U \cup \{n+1\}$. As Ledoit and Wolf (2004a) note, using this estimator ensures that the matrix of the covariances is positive definite, and thus, the TEV is a convex quadratic function of the weights of the stocks in the portfolio, which allows commercial MIP solvers such as CPLEX and Gurobi to be applied. The expected returns of the assets $i \in U \cup \{n+1\}$ are estimated as follows: $\overline{r}_i = \frac{1}{T-1} \sum_{t \in \{2,\ldots,T\}} \frac{P_{it} - P_{i,t-1}}{P_{i,t-1}}$.

After preliminary experiments, we adopt the following values for the input parameters of the heuristic solution approaches for all problem instances. For LBH, we set the number of stocks considered to $k+50$ and the maximum number of iterations without improvement to ten (i.e., $q = 50$ and $\overline{\nu} = 10$). For IGH, we adopt the same number of stocks considered as for LBH and a value of two as the maximum number of stocks to be removed from the best feasible portfolio found so far (i.e., $q = 50$ and $d = 2$).

We evaluate the in-sample performance of the tested solution approaches by using the following performance measures:

- OFV: objective function value of the best feasible portfolio found within the prescribed computational time limit. Note that the OFV is scaled by a factor of 100,000.
- MIP gap [%]: relative deviation between the OFV and the best lower bound (LB) provided by the solver within the prescribed computational time limit, calculated as: $(\text{OFV} - \text{LB})/\text{OFV}$.

To evaluate the out-of-sample performance of the tested solution approaches we report the following performance measures. Thereby, we let T' be the end of the out-of-sample period, i.e., $T' = 156$:

- TE_{TEV} [%]: annualized standard deviation of the differences between the portfolio returns and the index returns during the out-of-sample period, calculated as:

$$\sqrt{\frac{52}{T'-T}\sum_{t\in\{T+1,\ldots,T'\}}\left(r_t^D-\overline{r}^D\right)^2} \tag{1.46}$$

where $r_t^D=\left(\frac{\sum_{i\in U\cup\{n+1\}}P_{it}X_i}{\sum_{i\in U\cup\{n+1\}}P_{i,t-1}X_i}-1\right)-\left(\frac{I_t}{I_{t-1}}-1\right)$ for $t\in\{T+1,\ldots,T'\}$ and $\overline{r}^D=\frac{1}{T'-T}\sum_{t\in\{T+1,\ldots,T'\}}r_t^D$. Note that the TE_{TEV} is the out-of-sample tracking-error measure that corresponds to the TEV that is optimized in-sample by LBH, IGH, and M-Q.

- TE_{MAD}: annualized average absolute difference between the portfolio values and the index values during the out-of-sample period, calculated as:

$$\frac{52}{T'-T}\sum_{t\in\{T+1,\ldots,T'\}}\left|\sum_{i\in U\cup\{n+1\}}P_{it}X_i-I_t\right|\frac{100}{C} \tag{1.47}$$

Note that the TE_{MAD} is the out-of-sample tracking-error measure that corresponds to the MAD that is optimized in-sample by M-L. Also note that the TE_{MAD} is scaled to an investment budget of 100.

- ER [%]: annualized difference between the cumulated portfolio return and the cumulated index return during the out-of-sample period, calculated as:

$$\left(\frac{\sum_{i\in U\cup\{n+1\}}P_{iT'}X_i}{\sum_{i\in U\cup\{n+1\}}P_{iT}X_i}\right)^{\frac{52}{T'-T}}-\left(\frac{I_{T'}}{I_T}\right)^{\frac{52}{T'-T}} \tag{1.48}$$

All calculations were performed on an HP Z820 workstation with two 3.1 GHz Intel Xeon CPUs and 128 GB of RAM. As the termination criterion, we prescribed a computational time limit of 60 seconds for the heuristic solution approaches and a fairly longer computational time limit of 300 seconds for the exact solution approaches. We implemented the two exact solution approaches and the two heuristic solution approaches in C, and we used Gurobi 7.5 as the solver. We used the default solver settings, except for the MILP formulation (M-C) that is solved in Algorithm 1.1, in which we stopped the Gurobi solver as soon as the MIP gap reached a value of 10% or lower.

1.5.2 Novel problem instances

To the best of our knowledge, there is no set of instances available in the literature for the EITP. In the existing sets of problem instances (cf., e.g., Beasley et al. 2003; Canakgoz and Beasley 2008; Guastaroba et al. 2009; Strub and Baumann 2018), the weights of the stocks in the index at the time of rebalancing, i.e., at the end of week T, are not provided. Furthermore, no available set of problem instances contains very large regional and global stock-market indices. The largest existing problem instance that corresponds to the Russell 3000 index consists of fewer than 2,500 US stocks. Hence, we here provide a set of novel instances of the EITP. These instances are based on real-world data from nine different stock-market indices maintained by Thomson Reuters (TR) for two different time periods. Table 1.4 lists the names of the indices, the number of stocks n in the indices, the considered time periods, and the applicable values for k. Combining the nine different indices, the two periods, the applicable values for k, the three considered scenarios, and the three consider α values, we obtain a set of 297 instances in total. We used DATASTREAM to download 156 weekly values of each index and 156 weekly closing prices of the constituents of each index during the corresponding time period. We consider the constituents of the index at the end of the in-sample period, i.e., at the end of week 104. As done by Beasley et al. (2003), Canakgoz and Beasley (2008), and Strub and Baumann (2018), we disregard the constituents for which the price data for the considered 156 weeks are incomplete; thus, the number of stocks n in the index can differ between the two different time periods. We also provide the weight of each constituent in the index at the end of the in-sample period. For all indices, the sum of the original weights of the stocks with complete price data is at least 95% for both periods. The weights of these index constituents are then scaled for each index such that their sum is equal to one.

Figure 1.1 shows the evolution of the value of the TR Global index over the two overlapping periods from August 2012 to July 2015 (period 1) and from August 2013 to July 2016 (period 2). In the figure, both periods are split into three equal parts consisting of 52 weeks each. The first two parts of each period correspond to the in-sample period, and the third part corresponds to the out-of-sample period. As Figure 1.1 shows, the Black Monday market crash, a drastic downward revision of the growth expectations for China's economy, and the UK referendum regarding the European Union all led to high market volatility during the time frame marked in red. These events did not affect the out-of-sample period of period 1, but strongly impacted the out-of-sample period of period 2. Hence, the differentiation between periods 1 and 2 enables investigation of the performance of portfolios during out-of-sample periods characterized by both low and high market volatility.

Table 1.4: Problem instances.

	Index	n	Time period	k
Period 1	TR Africa	168	08/2012–07/2015	100
	TR Latin America	194	08/2012–07/2015	100
	TR Europe	1,310	08/2012–07/2015	100, 200
	TR United States	1,592	08/2012–07/2015	100, 200
	TR North America	1,866	08/2012–07/2015	100, 200
	TR Global Emerging Markets	2,912	08/2012–07/2015	100, 200
	TR Asia Pacific	5,018	08/2012–07/2015	100, 200
	TR Global Developed Markets	5,965	08/2012–07/2015	100, 200
	TR Global	8,877	08/2012–07/2015	100, 200
Period 2	TR Africa	168	08/2013–07/2016	100
	TR Latin America	246	08/2013–07/2016	100, 200
	TR Europe	1,504	08/2013–07/2016	100, 200
	TR United States	2,222	08/2013–07/2016	100, 200
	TR Global Emerging Markets	2,532	08/2013–07/2016	100, 200
	TR North America	2,620	08/2013–07/2016	100, 200
	TR Asia Pacific	4,663	08/2013–07/2016	100, 200
	TR Global Developed Markets	6,896	08/2013–07/2016	100, 200
	TR Global	9,427	08/2013–07/2016	100, 200

1.5.3 Portfolios without rebalancing: in-sample and out-of-sample performance analysis

In this subsection, we investigate the index-tracking capabilities of the portfolios given in the three scenarios before rebalancing to provide a reference against which to assess the index-tracking capabilities of the portfolios after rebalancing. For this purpose, we present in-sample and out-of-sample results for scenarios I, II, and III under the assumption that the given portfolio is not rebalanced by the investment fund's manager at the end of the in-sample period and thus remains unchanged for the out-of-sample period, i.e., $X_i = Y_i$ for all $i \in U \cup \{n+1\}$. For scenario III, this means that the portfolio after rebalancing still consists of cash only, i.e., $X_{n+1} = Y_{n+1} = \frac{C}{P_{n+1,T}}$ and $X_i = Y_i = 0$ for each stock $i \in U$. Note that these portfolios are not necessarily feasible portfolios; for example, the constraint regarding the prescribed minimum excess return or the constraints regarding the minimum and maximum portfolio weights might be violated.

Table 1.5 summarizes the in-sample and out-of-sample results for these portfolios for period 1 and period 2. Column one indicates the considered scenario and column two shows the number of considered instances for period 1. Columns three, four, and five show the average OFV, specifically the TEV according to (1.1), the average $\mathrm{TE}_{\mathrm{TEV}}$, and the

Figure 1.1: Considered time periods.

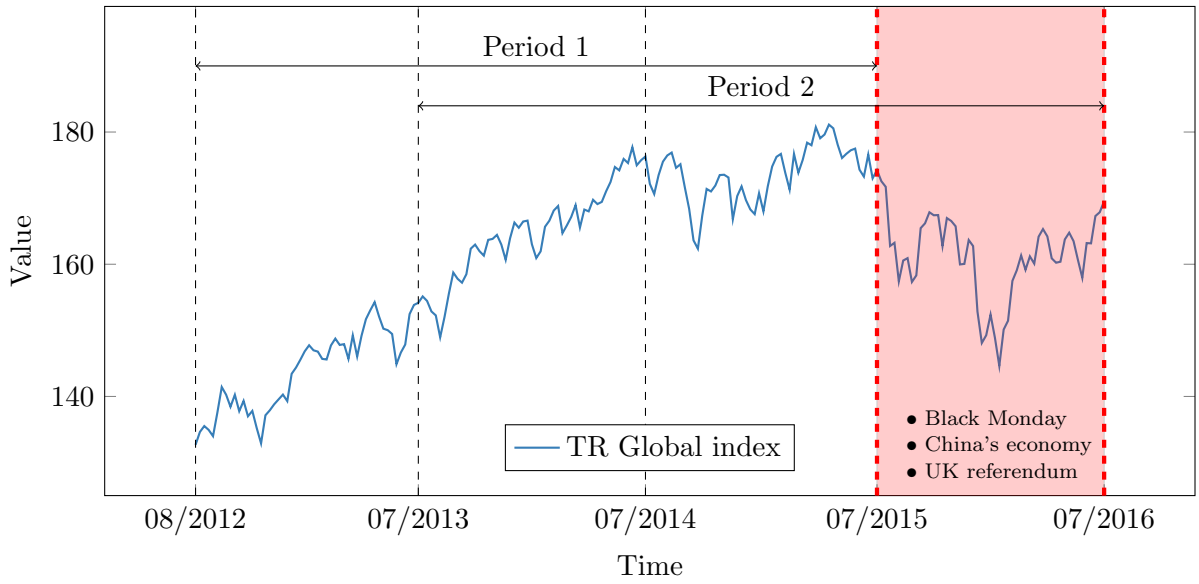


Table 1.5: In-sample and out-of-sample results for portfolios without rebalancing.

	Period 1				Period 2			
	# INST	OFV	TE_{TEV}	TE_{MAD}	# INST	OFV	TE_{TEV}	TE_{MAD}
Scenario I	16	0.71	2.86	37.09	17	0.75	3.39	58.83
Scenario II	16	19.20	13.13	314.42	17	15.61	13.37	245.51
Scenario III	16	124.62	13.37	244.88	17	137.00	19.49	457.20

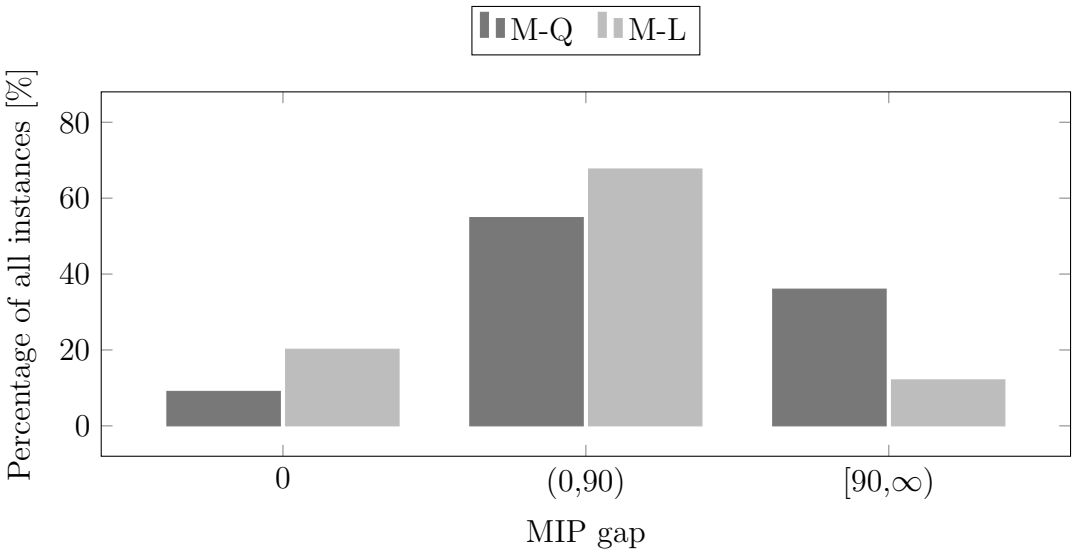
average TE_{MAD}, respectively, for period 1. Columns six to nine report the corresponding results for period 2.

From Table 1.5, we can gain the following insights. The portfolios of scenario I clearly outperform those of scenarios II and III in terms of the objective function value and both out-of-sample tracking-error measures, regardless of the considered period. Therefore, portfolios that consist of stocks with high weights in the index tend to have a better index-tracking capability than portfolios that consist of stocks with low index weights.

1.5.4 MIP gaps: M-Q in comparison with M-L

In this subsection, we show that in contrast to M-L, M-Q leads to feasible portfolios within a limited computational time that may be improved substantially in terms of the objective function value. For this purpose, we investigate the in-sample performance in terms of the MIP gap of the portfolios obtained with M-Q and M-L within the prescribed

Figure 1.2: In-sample results for rebalanced portfolios in terms of the MIP gap.



computational time limit of 300 seconds.

Figure 1.2 shows the distribution of the in-sample results in terms of the MIP gap for M-Q and M-L. M-L determines provably optimal portfolios for about 20% of all problem instances, which is twice as much as M-Q does. Furthermore, M-L determines feasible portfolios with a MIP gap larger than 90% for only about 10% of all problem instances. In contrast, the MIP gap of the feasible portfolios determined with M-Q are larger than 90% for more than one third of all problem instances. These results suggest that an exact solution approach may be appropriate to the EITP when the MAD is used as the objective function, but may not be appropriate when the TEV is minimized. Hence, improvements in terms of the objective function value might be achieved by applying heuristics to the EITP when the TEV is used as the objective function, which is supported by the results provided in the following.

1.5.5 In-sample performance analysis: LBH and IGH in comparison with M-Q

In this subsection, we show that LBH and IGH achieve substantial improvements in terms of the objective function value compared to M-Q within a limited computational time. For this purpose, we present in-sample results for the portfolios of scenarios I, II, and III after rebalancing using the three considered solution approaches.

Table 1.6 summarizes the in-sample results in terms of the OFV, i.e., the TEV. Columns one to four indicate the considered scenario, the maximum portfolio cardinal-

ity k, the annualized prescribed minimum expected excess return α, and the number of considered instances, respectively. Columns five, seven, and nine show the average OFV for the solution approaches M-Q, LBH, and IGH, respectively; the best average OFV in each row is shown in bold. Columns six, eight, and ten show the numbers of instances for which M-Q, LBH, and IGH, respectively, are not able to find a feasible portfolio within the prescribed computational time limit. Note that we exclude all instances for which not all three considered solution approaches devise at least a feasible portfolio from the calculation of the average OFV.

From Table 1.6, we can gain the following insights:

- Within the prescribed computational time limit of 60 seconds, LBH and IGH are able to find much better portfolios in terms of the objective function value than M-Q is within 300 seconds.
- LBH and IGH are able to find feasible portfolios for all considered problem instances within the prescribed computational time limit, whereas this is not the case for M-Q. This finding demonstrates that constructing a feasible portfolio for the considered problem is not straightforward.
- Compared with the portfolios before rebalancing (cf. Table 1.5), the portfolios found by using LBH and IGH have considerably lower objective function values.
- IGH tends to find better portfolios in terms of the objective function value than LBH does for scenarios I and II. The opposite is true for scenario III, in which the investment fund has a portfolio before rebalancing that consists only of cash. Hence, LBH should be applied to construct a portfolio from cash, and IGH should be applied to rebalance an existing stock portfolio.

1.5.6 Out-of-sample performance analysis

In this subsection, we provide insights that minimizing the TEV may be superior to minimizing the MAD in terms of the out-of-sample tracking error. In Subsection 1.5.6.1, considering only the instances that could be solved to proven optimality by M-Q and M-L, we show that the portfolios obtained with M-Q are superior to those obtained with M-L, interestingly in terms of both the $\mathrm{TE}_{\mathrm{TEV}}$ and the $\mathrm{TE}_{\mathrm{MAD}}$. In Subsection 1.5.6.2, we show that the feasible portfolios determined with LBH and IGH are superior to the feasible portfolios determined with M-L also in terms of both the $\mathrm{TE}_{\mathrm{TEV}}$ and the $\mathrm{TE}_{\mathrm{MAD}}$.

Table 1.6: In-sample results for rebalanced portfolios in terms of the OFV.

				M-Q (300s)		LBH (60s)		IGH (60s)	
		α p.a.	# INST	OFV	# NFP	OFV	# NFP	OFV	# NFP
Scenario I	$k = 100$	0%	18	13.22	0	0.55	0	**0.48**	0
		1%	18	13.05	0	0.57	0	**0.49**	0
		2%	18	16.33	1	0.57	0	**0.49**	0
	$k = 200$	0%	15	17.72	0	**0.24**	0	0.26	0
		1%	15	11.01	0	**0.25**	0	0.26	0
		2%	15	16.64	0	**0.27**	0	**0.27**	0
Scenario II	$k = 100$	0%	18	16.77	0	13.98	0	**13.25**	0
		1%	18	17.33	0	14.12	0	**12.71**	0
		2%	18	18.44	0	**12.31**	0	12.92	0
	$k = 200$	0%	15	15.77	1	5.10	0	**4.98**	0
		1%	15	17.07	1	4.31	0	**4.27**	0
		2%	15	14.30	1	4.24	0	**4.06**	0
Scenario III	$k = 100$	0%	18	55.73	0	**0.42**	0	0.43	0
		1%	18	55.96	0	**0.42**	0	0.43	0
		2%	18	56.21	0	**0.43**	0	0.44	0
	$k = 200$	0%	15	66.75	0	**0.17**	0	**0.17**	0
		1%	15	67.02	0	**0.17**	0	0.18	0
		2%	15	67.31	0	**0.17**	0	0.18	0

1.5.6.1 M-Q in comparison with M-L

We compare the out-of-sample performance of the portfolios of scenarios I, II, and III after rebalancing using M-Q and M-L for the instances that could be solved to proven optimality by both solution approaches within the prescribed computational time limit.

Table 1.7 summarizes the out-of-sample tracking errors for these portfolios. Column one indicates the considered scenario and column two shows the number of considered instances for each scenario. Columns three and four show the average $\mathrm{TE}_{\mathrm{TEV}}$ for the approaches M-Q and M-L, respectively. Columns five and six report the average $\mathrm{TE}_{\mathrm{MAD}}$ for the same approaches.

From Table 1.7, we can gain the insight that M-Q leads to lower out-of-sample tracking errors in terms of the $\mathrm{TE}_{\mathrm{TEV}}$ but also in terms of the $\mathrm{TE}_{\mathrm{MAD}}$, even though the portfolios determined with M-L have the lowest possible in-sample MAD.

Figure 1.3 shows the out-of-sample excess returns against the out-of-sample tracking errors of the rebalanced portfolios obtained using the solution approaches M-Q and M-L for the instances that could be solved to proven optimality by both solution approaches

Table 1.7: Out-of-sample tracking errors for instances solved to proven optimality by both M-Q and M-L.

		TE_{TEV}		TE_{MAD}	
	# INST	M-Q (300s)	M-L (300s)	M-Q (300s)	M-L (300s)
Scenario I	9	**3.92**	6.54	**27.06**	71.48
Scenario II	9	**4.16**	7.99	**91.61**	105.42
Scenario III	9	**4.00**	6.50	**53.93**	65.48
	Average	**4.03**	7.01	**57.54**	80.80

within the prescribed computational time limit. This figure indicates that M-Q is superior to M-L with respect to both objectives in enhanced index tracking, i.e., obtaining a low tracking error and achieving a small target excess return in the out-of-sample period.

These findings suggest that minimizing the TEV leads to better enhanced index-tracking portfolios than minimizing the MAD.

1.5.6.2 LBH and IGH in comparison with M-L

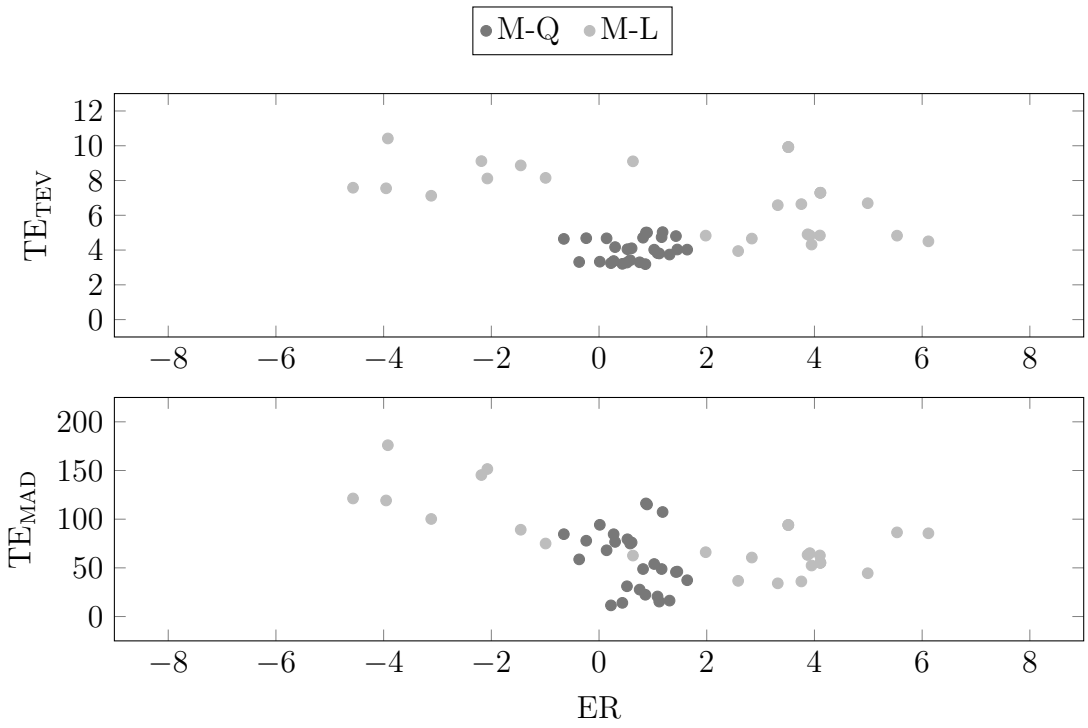
We compare the out-of-sample tracking errors of the portfolios of scenarios I, II, and III after rebalancing using LBH, IGH, and M-L. Furthermore, we provide insights on the impact of the prescribed computational time limit on the out-of-sample tracking errors of the portfolios obtained with M-L. Finally, we provide out-of-sample risk-return characteristics of the rebalanced portfolios.

Table 1.8 summarizes the out-of-sample results for periods 1 and 2 individually. Column one indicates the considered period. The contents of columns two to five are the same as those of columns one to four of Table 1.6. Columns six to eight show the average TE_{TEV} for the considered instances for the solution approaches M-L, LBH, and IGH, respectively. Columns nine to eleven present the average TE_{MAD} for the same instances and the same solution approaches. The best average TE_{TEV} and the best average TE_{MAD} in each row are shown in bold. Note that all considered solution approaches found at least a feasible solution for all instances.

From Table 1.8, we can gain the following insights:

- Regardless of the period considered, the average TE_{TEV} for the portfolios obtained with LBH and IGH within 60 seconds are much lower than those for the portfolios obtained with M-L within 300 seconds. Interestingly, LBH and IGH also devise portfolios with a much lower average TE_{MAD}, even though M-L explicitly aims at minimizing the MAD during the in-sample period.

Figure 1.3: Out-of-sample risk-return characteristics for instances solved to proven optimality by M-Q and M-L.



- LBH and IGH lead to considerably lower worst-case TE_{TEV} and TE_{MAD} than M-L does, regardless of the period considered. These empirical findings support the arguments presented in Subsection 1.3.2.

- For all considered solution approaches, the average TE_{TEV} is higher for period 2 than for period 1 because the out-of-sample period of period 2 exhibits higher market volatility than that of period 1.

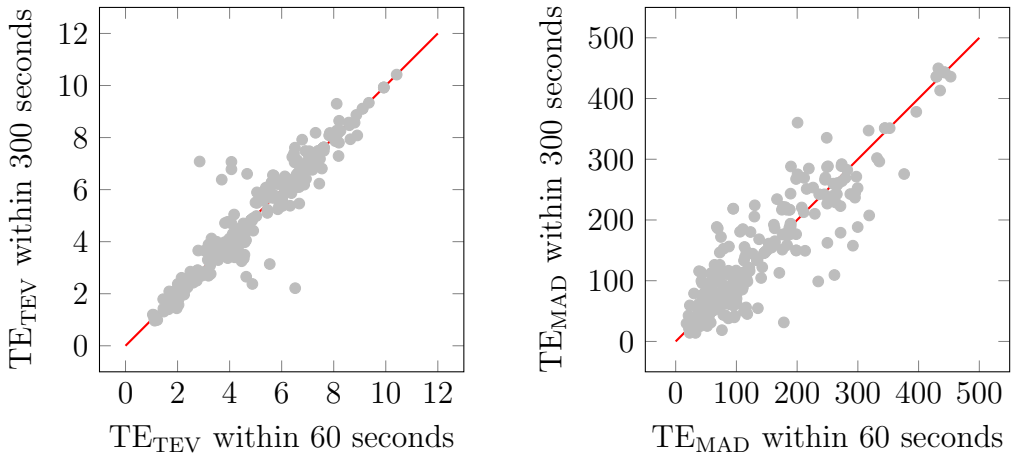
- For scenario II, in which the index-tracking capability of the portfolio before rebalancing is rather poor, the average TE_{TEV} are higher than those for the other scenarios. This is consistent with the higher objective function values, i.e., the higher TEV, for scenario II (cf. Table 1.6).

Figure 1.4 shows the out-of-sample tracking errors of the rebalanced portfolios obtained with the exact solution approach M-L within 60 and 300 seconds. For the portfolios that are below the red line, the longer computational time limit leads to a performance improvement in terms of the TE_{TEV} or the TE_{MAD}. By contrast, the corresponding out-of-sample tracking-error measure deteriorates for the portfolios that are above the red line. Figure 1.4 shows that the out-of-sample tracking errors of the rebalanced portfolios

Table 1.8: Out-of-sample tracking errors for rebalanced portfolios.

					TE_{TEV}			TE_{MAD}		
			α p.a.	# INST	M-L (300s)	LBH (60s)	IGH (60s)	M-L (300s)	LBH (60s)	IGH (60s)
Period 1	Scenario I	$k = 100$	0%	9	3.81	2.89	**2.77**	74.06	**36.99**	44.10
			1%	9	3.77	2.90	**2.79**	88.49	**38.63**	41.50
			2%	9	4.10	2.91	**2.80**	70.49	**39.98**	49.12
		$k = 200$	0%	7	2.40	**2.01**	2.03	72.63	**19.87**	21.29
			1%	7	2.33	2.07	**2.06**	75.18	**21.75**	24.63
			2%	7	2.25	2.12	**2.02**	68.61	**19.60**	27.54
	Scenario II	$k = 100$	0%	9	6.38	4.44	**4.05**	195.97	**123.75**	167.05
			1%	9	6.70	4.32	**4.12**	200.17	109.18	**106.90**
			2%	9	6.75	**4.17**	4.29	211.49	148.26	**128.11**
		$k = 200$	0%	7	5.94	3.86	**3.82**	196.66	129.62	**126.75**
			1%	7	5.94	3.79	**3.63**	164.26	128.18	**106.23**
			2%	7	5.90	3.79	**3.77**	198.45	137.32	**131.99**
	Scenario III	$k = 100$	0%	9	5.55	**2.72**	2.75	139.05	61.91	**60.22**
			1%	9	5.63	**2.67**	2.73	158.43	61.75	**54.25**
			2%	9	5.72	2.76	**2.73**	150.18	**51.97**	54.91
		$k = 200$	0%	7	4.77	**1.93**	1.94	153.18	57.14	**56.20**
			1%	7	5.09	**1.92**	**1.92**	172.92	**45.32**	51.31
			2%	7	5.17	**1.91**	1.94	169.58	60.90	**58.09**
				Average	4.96	3.00	**2.94**	142.33	**72.15**	73.50
				Worst case	10.42	5.80	**5.76**	360.26	325.82	**288.16**
Period 2	Scenario I	$k = 100$	0%	9	3.43	3.11	**2.81**	55.23	**47.06**	48.91
			1%	9	3.49	3.16	**2.89**	57.81	**48.44**	61.07
			2%	9	3.43	3.15	**2.98**	84.97	**51.51**	52.45
		$k = 200$	0%	8	3.40	**2.30**	2.31	51.92	41.71	**38.36**
			1%	8	3.03	2.33	**2.27**	50.91	43.52	**41.97**
			2%	8	2.91	2.35	**2.31**	64.69	41.96	**40.40**
	Scenario II	$k = 100$	0%	9	6.08	5.23	**4.69**	155.74	90.58	**85.23**
			1%	9	5.83	**5.11**	5.14	179.66	94.28	**80.88**
			2%	9	6.04	**4.85**	5.12	187.62	**83.59**	96.51
		$k = 200$	0%	8	7.13	4.14	**4.10**	219.27	80.06	**67.09**
			1%	8	7.12	**4.18**	4.23	213.18	77.50	**67.84**
			2%	8	7.21	4.18	**4.13**	219.26	78.55	**77.00**
	Scenario III	$k = 100$	0%	9	5.63	**2.79**	2.87	90.81	72.54	**64.17**
			1%	9	5.71	2.84	**2.82**	115.90	**66.57**	70.31
			2%	9	5.81	**2.83**	2.87	118.59	**66.38**	75.49
		$k = 200$	0%	8	5.77	2.27	**2.21**	110.04	69.82	**65.64**
			1%	8	5.67	**2.20**	2.23	118.48	74.46	**73.53**
			2%	8	5.79	**2.25**	**2.25**	127.51	**68.33**	71.38
				Average	5.18	3.31	**3.25**	123.00	66.64	**65.76**
				Worst case	9.92	**7.19**	7.78	449.74	168.53	**168.45**

Figure 1.4: Out-of-sample tracking errors for rebalanced portfolios obtained with M-L within 60 and 300 seconds.



obtained within 60 seconds are similar to those of the portfolios devised within 300 seconds; for many portfolios obtained within 300 seconds the TE_{TEV} and the TE_{MAD} are even worse. This suggests that the results provided in Table 1.8 for M-L would not change considerably if a computational time limit of more than 300 seconds was imposed.

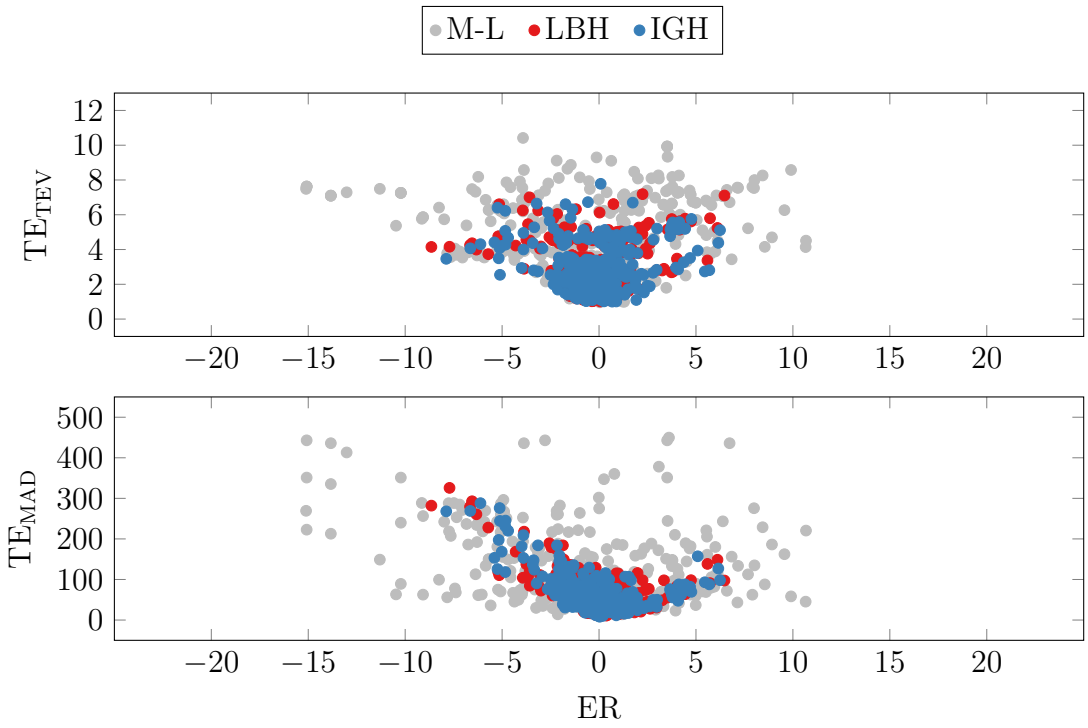
Figure 1.5 shows the out-of-sample excess returns against the out-of-sample tracking errors of the rebalanced portfolios obtained using the solution approaches M-L, LBH, and IGH. LBH and IGH devise portfolios with a performance that is similar to M-L in terms of the out-of-sample excess returns. Furthermore, the out-of-sample excess returns of the portfolios obtained with LBH and IGH are much less volatile. Finally, Figure 1.5 shows that a portfolio that achieves the prescribed minimum expected excess return during the in-sample period is not necessarily guaranteed to achieve an excess return during the out-of-sample period as there are many portfolios with a negative out-of-sample excess return.

1.5.7 Portfolio compositional characteristics: LBH and IGH in comparison with M-L

In this subsection, we offer further insights with respect to the compositions of the rebalanced portfolios. For this purpose, we report various compositional characteristics of portfolios that have been rebalanced using the solution approaches LBH, IGH, and M-L.

Figure 1.6 shows the following compositional characteristics for the portfolios of scenarios I, II, and III after rebalancing: the active share, i.e., the sum of the absolute differences between the weights of the assets in the portfolio and the weights of the assets

Figure 1.5: Out-of-sample risk-return characteristics for rebalanced portfolios.



in the index ($\frac{1}{2}\sum_{i\in U\cup\{n+1\}}\left|\frac{P_{iT}X_i}{C}-w_i^I\right|$); the portfolio cardinality, i.e., the number of different stocks that are selected after rebalancing ($|\{i\in I: X_i>0\}|$); the transaction costs, i.e., the sum of the fixed and proportional transaction costs relative to the transaction cost budget ($\frac{1}{\gamma C}(\sum_{i\in I}G_i+\sum_{i\in U\setminus I:Y_i>0}(c_i^sY_iP_{iT}+c_i^f))$); and the weight of the cash asset ($\frac{P_{n+1,T}X_{n+1}}{C}$). We present the compositional characteristics for all instances with $k=200$ and $\alpha=0\%$ sorted in non-decreasing order of n. The compositional characteristics for the instances with $k=100$ and $\alpha>0\%$ are not shown because they are similar to those presented in Figure 1.6. For reference, we also present the compositional characteristics in the case that no rebalancing is performed.

From Figure 1.6, we can gain the following insights:

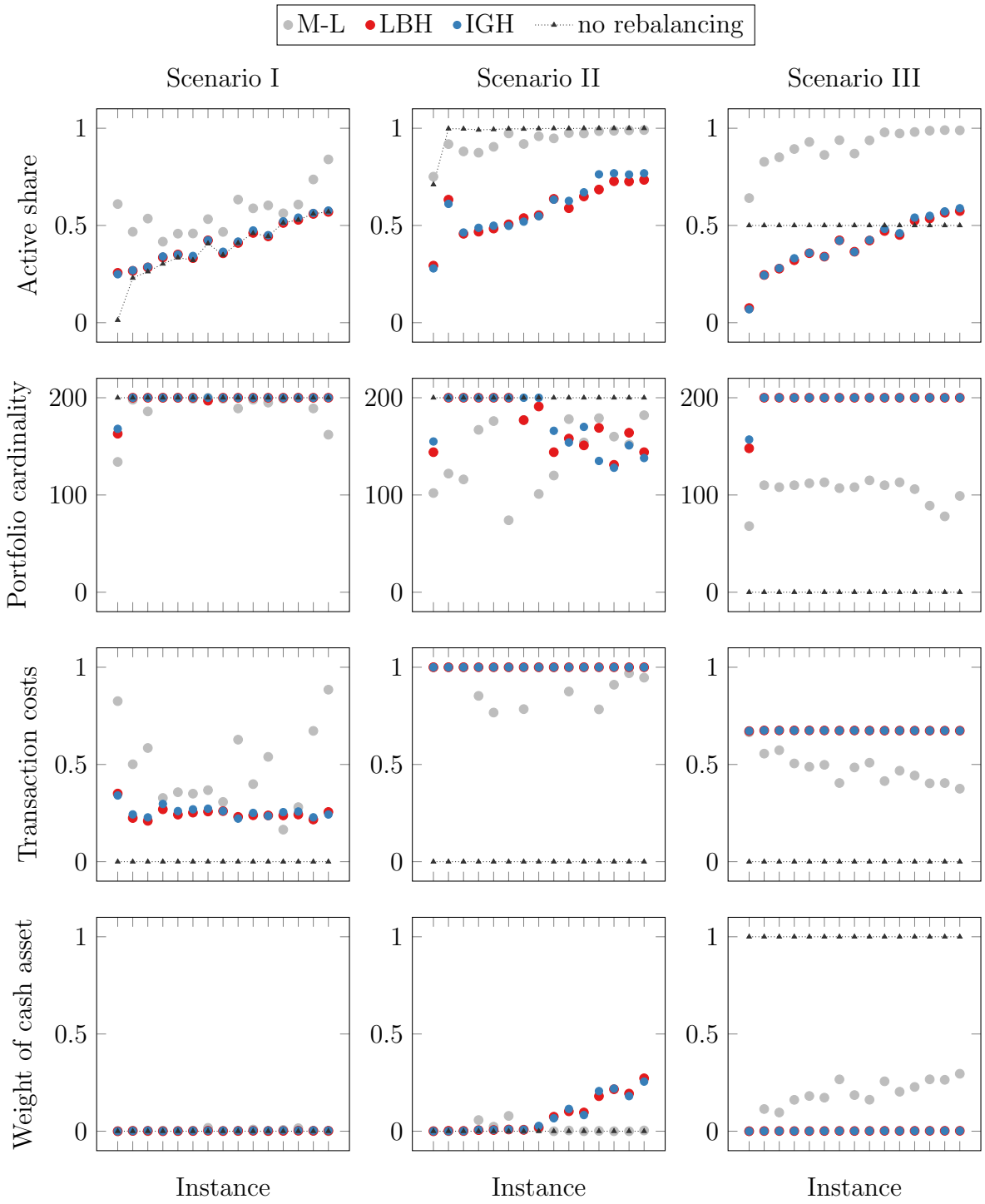
- LBH and IGH lead to portfolios with lower active shares and higher cardinalities than M-L does, regardless of the considered scenario. Thus, the compositions of the portfolios that have been rebalanced using LBH and IGH are more similar to the composition of the index. This finding is consistent with the arguments presented in Subsection 1.3.2 and offers a possible explanation for the superior performance of LBH and IGH in terms of both out-of-sample tracking-error measures, the $\mathrm{TE}_{\mathrm{TEV}}$ and the $\mathrm{TE}_{\mathrm{MAD}}$, reported in Table 1.8.

- In scenario I, LBH and IGH incur lower transaction costs than M-L does because the portfolio before rebalancing is already invested in the k stocks with the highest index weights. In scenarios II and III, the transaction costs for LBH and IGH are higher than those for M-L. To rebalance the portfolios with a poor index-tracking capability of scenario II, LBH and IGH completely exhaust the transaction cost budget for all problem instances.

- For scenarios I and III, LBH and IGH lead to portfolios in which the weight of the cash asset is almost zero for all instances. By contrast, M-L leads to portfolios that hold a substantial amount of cash after rebalancing for scenario III. For scenario II, in which the index-tracking capacity of the portfolio before rebalancing is rather poor, LBH and IGH also lead to portfolios that contain a considerable proportion of cash. This might be because the transaction cost budget is not sufficiently large to sell all currently held stocks with low index weights and exchange them for stocks with high index weights. In this case, it is more beneficial in terms of the objective function value to sell the stocks with low index weights and maintain the revenue in cash.

From the results provided in this section, the six main findings are as follows. 1) In contrast to M-L, M-Q leads to feasible portfolios within a limited computational time that may be improved substantially in terms of the objective function value, which indicates the possible improvements that might be achieved by applying LBH and IGH. 2) LBH and IGH are indeed able to determine considerably better feasible portfolios in terms of the objective function value within 60 seconds than M-Q is within 300 seconds. 3) LBH should be used to construct a portfolio from cash. 4) IGH should be used to rebalance an existing stock portfolio. 5) In terms of both out-of-sample tracking-error measures, the $\mathrm{TE}_{\mathrm{TEV}}$ and the $\mathrm{TE}_{\mathrm{MAD}}$, considering only the instances that could be solved to proven optimality by both M-Q and M-L, the portfolios obtained with M-Q are superior to those obtained with M-L. 6) In terms of both out-of-sample tracking-error measures, the $\mathrm{TE}_{\mathrm{TEV}}$ and the $\mathrm{TE}_{\mathrm{MAD}}$, the feasible portfolios determined by running LBH and IGH for 60 seconds are superior to the feasible portfolios determined with M-L within 300 seconds. These findings suggest that minimizing the TEV is superior to minimizing a dissimilarity function such as the MAD in terms of the out-of-sample tracking error.

Figure 1.6: Compositional characteristics of rebalanced portfolios.



1.6 Conclusions

In this paper, we consider the problem of determining the portfolio for an enhanced index-tracking fund. For this problem, we propose novel mixed-integer linear and quadratic programming formulations and novel matheuristics to minimize the tracking error variance or the mean-absolute deviation between the historical values of the portfolio and the index. The results of a computational experiment using the proposed matheuristics and exact solution approaches based on the mixed-integer linear and quadratic programming formulations suggest that it is superior in terms of the out-of-sample tracking error to minimize the tracking error variance instead of the mean-absolute deviation.

In future research, it would be interesting to investigate whether the out-of-sample tracking error achieved here could be further improved by using more sophisticated estimators for the covariance matrix. Periodic portfolio rebalancing could be applied to further improve the out-of-sample performance of the presented solution approaches, especially in market environments with high volatility. Moreover, a promising direction for future research is to combine the proposed matheuristics based on the findings regarding their individual strengths to further improve the out-of-sample tracking error.

Bibliography

Andriosopoulos, K., Nomikos, N., 2014. Performance replication of the Spot Energy Index with optimal equity portfolio selection: Evidence from the UK, US and Brazilian markets. European Journal of Operational Research 234 (2), 571–582.

Beasley, J. E., Meade, N., Chang, T.-J., 2003. An evolutionary heuristic for the index tracking problem. European Journal of Operational Research 148 (3), 621–643.

Bertsimas, D., Shioda, R., 2009. Algorithm for cardinality-constrained quadratic optimization. Computational Optimization and Applications 43 (1), 1–22.

Bruni, R., Cesarone, F., Scozzari, A., Tardella, F., 2015. A linear risk-return model for enhanced indexation in portfolio optimization. OR Spectrum 37 (3), 735–759.

Canakgoz, N. A., Beasley, J. E., 2008. Mixed-integer programming approaches for index tracking and enhanced indexation. European Journal of Operational Research 196 (1), 384–399.

Chiam, S. C., Tan, K. C., Al Mamun, A., 2013. Dynamic index tracking via multi-objective evolutionary algorithm. Applied Soft Computing 13 (7), 3392–3408.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., Semet, F., 2002. A guide to vehicle routing heuristics. Journal of the Operational Research society 53 (5), 512–522.

Filippi, C., Guastaroba, G., Speranza, M., 2016. A heuristic framework for the bi-objective enhanced index tracking problem. Omega 65, 122–137.

Fischetti, M., Lodi, A., 2003. Local branching. Mathematical Programming 98 (1–3), 23–47.

Gaivoronski, A. A., Krylov, S., Van der Wijst, N., 2005. Optimal portfolio selection and dynamic benchmark tracking. European Journal of Operational Research 163 (1), 115–131.

Guastaroba, G., Mansini, R., Ogryczak, W., Speranza, M. G., 2016. Linear programming models based on Omega ratio for the enhanced index tracking problem. European Journal of Operational Research 251 (3), 938–956.

Guastaroba, G., Mansini, R., Speranza, M. G., 2009. On the effectiveness of scenario generation techniques in single-period portfolio optimization. European Journal of Operational Research 192 (2), 500–511.

Guastaroba, G., Speranza, M. G., 2012. Kernel Search: An application to the index tracking problem. European Journal of Operational Research 217 (1), 54–68.

Jansen, R., Van Dijk, R., 2002. Optimal benchmark tracking with small portfolios. The Journal of Portfolio Management 28 (2), 33–39.

Jorion, P., 2003. Portfolio optimization with tracking-error constraints. Financial Analysts Journal 59 (5), 70–82.

Konno, H., Wijayanayake, A., 2001. Minimal cost index tracking under nonlinear transaction costs and minimal transaction unit constraints. International Journal of Theoretical and Applied Finance 4 (6), 939–957.

Krink, T., Mittnik, S., Paterlini, S., 2009. Differential evolution and combinatorial search for constrained index-tracking. Annals of Operations Research 172 (1), 153–176.

Kwiatkowski, J. W., 1992. Algorithms for index tracking. IMA Journal of Management Mathematics 4 (3), 279–299.

Ledoit, O., Wolf, M., 2004a. Honey, I shrunk the sample covariance matrix. The Journal of Portfolio Management 30 (4), 110–119.

Ledoit, O., Wolf, M., 2004b. A well-conditioned estimator for large-dimensional covariance matrices. Journal of Multivariate Analysis 88 (2), 365–411.

Maringer, D., Oyewumi, O., 2007. Index tracking with constrained portfolios. Intelligent Systems in Accounting, Finance and Management 15 (1-2), 57–71.

Mutunge, P., Haugland, D., 2018. Minimizing the tracking error of cardinality constrained portfolios. Computers & Operations Research 90, 33–41.

Roll, R., 1992. A mean/variance analysis of tracking error. The Journal of Portfolio Management 18 (4), 13–22.

Rudd, A., 1980. Optimal selection of passive portfolios. Financial Management, 57–66.

Rudolf, M., Wolter, H.-J., Zimmermann, H., 1999. A linear model for tracking error minimization. Journal of Banking & Finance 23 (1), 85–103.

Ruiz, R., Stützle, T., 2007. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research 177, 2033–2049.

Sant'Anna, L. R., Filomena, T. P., Caldeira, J. F., 2017a. Index tracking and enhanced indexing using cointegration and correlation with endogenous portfolio selection. The Quarterly Review of Economics and Finance 65, 146–157.

Sant'Anna, L. R., Filomena, T. P., Guedes, P. C., Borenstein, D., 2017b. Index tracking with controlled number of assets using a hybrid heuristic combining genetic algorithm and non-linear programming. Annals of Operations Research 258 (2), 849–867.

Scozzari, A., Tardella, F., Paterlini, S., Krink, T., 2013. Exact and heuristic approaches for the index tracking problem with UCITS constraints. Annals of Operations Research 205 (1), 235–250.

Strub, O., Baumann, P., 2018. Optimal construction and rebalancing of index-tracking portfolios. European Journal of Operational Research 264 (1), 370–387.

Strub, O., Trautmann, N., 2016. An iterated greedy heuristic for the 1/N portfolio tracking problem. In: Vitoriano, B., Parlier, G., de Werra, D. (Eds.), Proceedings of the 5th International Conference on Operations Research and Enterprise Systems. Rome, pp. 424–431.

Takeda, A., Niranjan, M., Gotoh, J., Kawahara, Y., 2013. Simultaneous pursuit of out-of-sample performance and sparsity in index tracking portfolios. Computational Management Science 10 (1), 21–49.

Wu, D., Kwon, R. H., Costa, G., 2017. A constrained cluster-based approach for tracking the S&P 500 index. International Journal of Production Economics 193, 222–243.

Paper II

Two continuous-time assignment-based models for the multi-mode resource-constrained project scheduling problem [2]

Mario Gnägi Tom Rihm Adrian Zimmermann Norbert Trautmann

Department of Business Administration
University of Bern

Contents

[2]Published in Computers & Industrial Engineering 129, 346–353 (DOI:10.1016/j.cie.2019.01.033)

Abstract

In the multi-mode resource-constrained project scheduling problem, a set of precedence-related project activities and, for each activity, a set of alternative execution modes are given. Each activity requires some time and some scarce resources during execution; these requirements depend on the selected execution mode. Sought is a project schedule, i.e, a start time and an execution mode for each activity, such that the project makespan is minimized. In the literature, beside a large variety of specific solution approaches, several Mixed-Integer Linear Programming (MILP) models have been proposed for this problem. We present two novel MILP models that are based on mode-selection, resource-assignment and sequencing variables; we enhance the performance of the models by eliminating some symmetric solutions from the search space and by adding some redundant sequencing constraints for pairs and for triples of activities that cannot be processed in parallel. In a comparison with reference models from the literature, it turned out that the advantages of the novel models are a simple structure, an enhanced flexibility, and a superior performance when the range of the activities' durations is relatively large.

2.1 Introduction

A project consists of a set of activities that are interrelated by precedence relationships and require time and scarce resources for their execution (cf., e.g., Brucker et al. 1999). Often there is a trade-off between the duration and the resource requirements of the project activities; this trade-off can be represented by alternative execution modes. Determining the start times and execution modes for the activities and allocating the scarce resources over time to the execution of the activities such that the project makespan is minimized represents a challenging combinatorial optimization problem.

We consider the multi-mode resource-constrained project scheduling problem (MR-CPSP), which can be described as follows (cf., e.g., Mika et al. 2015). Given are a set of project activities that require time and scarce resources for their execution, and a set of completion-start precedence relationships among the project activities. Three different types of resources are distinguished (cf. Słowiński 1980): renewable, non-renewable and doubly constrained resources. Renewable resources, e.g., manpower, are limited over each time period and are renewed from one time period to the next. For non-renewable

resources, e.g., raw materials, the total usage over the entire project duration is limited. Doubly constrained resources represent a combination of a renewable and a non-renewable resource (cf. Talbot 1982). Furthermore, for each activity, a set of alternative execution modes is given, which differ in the duration and the resource requirements of the activity. Sought is a project schedule, i.e., a start time and an execution mode for each activity, such that all precedence relationships are respected, the total required quantity of each renewable and each non-renewable resource does not exceed its prescribed capacity at any point in time, and the project makespan is minimized. The MRCPSP is a generalization of the widely studied single-mode resource-constrained project scheduling problem (RCPSP) and has been applied to the scheduling of, e.g., table-tennis leagues (cf. Knust 2010), construction projects (cf. Xu and Zeng 2015), and automotive R&D projects (cf. Bartels and Zimmermann 2015).

In the literature, in addition to many problem-specific heuristic and exact solution approaches, several Mixed-Integer Linear Programming (MILP) models have been proposed for the MRCPSP (cf., e.g., Mika et al. 2015 for an overview). The models can be classified into discrete-time (DT) models and continuous-time (CT) models. In DT models, the planning horizon is divided into a set of equal-length time intervals, and the activities can start only at the beginning of each of these intervals. In general, DT models involve time-indexed binary variables (cf., e.g., Talbot 1982; Maniezzo and Mingozzi 1999; Zhu et al. 2006); hence, the number of binary variables increases with the number of time intervals considered, which constitutes a potential drawback for projects that consist of activities with long durations, i.e., projects with a long planning horizon. By contrast, in CT models (cf., e.g., Kyriakidis et al. 2012), activities can start at any point in time over the planning horizon. In the known models, however, the formulation of the resource-capacity constraints requires a computation of the resource utilization based on the activities' start times, which is rather cumbersome. Further MILP models have been proposed for problems that extend the MRCPSP by, e.g., mode dependent time lags (cf. Sabzehparvar and Seyed-Hosseini 2008), multi-project scheduling (cf. Zapata et al. 2008) or non-preemptive activity splitting (cf. Cheng et al. 2015). In general, two major advantages of MILP models are their flexibility with respect to modifications of the planning situation and the possibility to apply standard solver software such as Gurobi or CPLEX (cf. Vielma 2015). The performance of an MILP-based solution approach for a specific planning problem, however, depends on the MILP model used and should be evaluated in an experimental analysis; for other scheduling problems, such analyzes have been performed by, e.g., Keha et al. (2009), Baker and Keller (2010) or Unlu and Mason (2010).

In this paper, we present two novel CT models for the MRCPSP. In both models, similar to the approach proposed in Trautmann et al. (2018) for the single-mode RCPSP, we use two types of binary variables to formulate the resource-capacity constraints: assignment variables specify which individual renewable-resource units are used for the execution of each activity, and sequencing variables specify the order in which pairs of activities that are assigned to the same renewable-resource unit are processed. In the second model, similar to the idea presented in Gnägi et al. (2018b), we use some additional auxiliary resource-overlap variables to identify these pairs of activities. To enhance the performance of the two novel models, we eliminate some symmetric solutions from the search space and add some redundant constraints for pairs and for triples of activities that cannot be processed in parallel due to the resource capacities. In a comparative analysis, we have applied the novel models and three reference models from the literature to two standard test sets and two novel test sets. Our computational results indicate a superior performance of the novel models when the range of the activities' durations is relatively large.

The remainder of this paper is structured as follows. In Section 2.2, we describe the MRCPSP in detail. In Section 2.3, we give an overview of the DT and CT models that we use as reference models. In Section 2.4, we present the novel MILP models for the MRCPSP. In Section 2.5, we report the computational results. In Section 2.6, we provide some concluding remarks and an outlook on future research.

2.2 Planning problem

In this section, we describe the MRCPSP in detail (cf. Subsection 2.2.1), and we illustrate the planning problem by means of an illustrative example (cf. Subsection 2.2.2). In the remainder of this paper, we use the sets and parameters listed in Table 2.1.

2.2.1 Multi-mode resource-constrained project scheduling problem

We assume that the project consists of a set V of activities. For each activity $i \in V$, a set M_i of alternative execution modes is given. The duration of activity $i \in V$ when executed in mode $m \in M_i$ is denoted as p_{im}. Furthermore, a set R of renewable resources and a set N of non-renewable resources are given. For the renewable resources $k \in R$, the resource capacities are denoted as R_k, and for the non-renewable resources $k \in N$, the resource capacities are denoted as W_k. Furthermore, we denote the resource requirements for the

Table 2.1: Sets and parameters.

Set	Description
V	Set of all activities ($V = \{0, 1, \ldots, n, n+1\}$)
M_i	Set of all execution modes of activity $i \in V$
R	Set of all renewable resources
N	Set of all non-renewable resources
E	Set of all completion-start precedence relationships
G	Activity-on-node graph
TE	Set of all pairs of activities that cannot be executed in parallel due to their precedence relationships
Parameter	**Description**
p_{im}	Duration of activity $i \in V$ when executed in mode $m \in M_i$
R_k	Capacity of renewable resource $k \in R$
W_k	Capacity of non-renewable resource $k \in N$
r_{ikm}	Requirement of activity $i \in V$ of renewable resource $k \in R$ per period when executed in mode $m \in M_i$
w_{ikm}	Requirement of activity $i \in V$ of non-renewable resource $k \in N$ when executed in mode $m \in M_i$
p_i^{max}	Maximum duration of activity $i \in V$
T	Planning horizon
ES_i	Earliest start time of activity $i \in V$
LS_i	Latest start time of activity $i \in V$

activities $i \in V$ when executed in mode $m \in M_i$ for the renewable resources $k \in R$ as r_{ikm} and for the non-renewable resources $k \in N$ as w_{ikm}. The set V of activities contains n real activities and two dummy activities 0 and $n+1$ representing the start and the completion of the project, respectively; both have a single execution mode with duration zero and no resource requirements. Furthermore, some completion-start precedence relationships are given among some pairs of activities $(i, j) \in V \times V$; these pairs of activities form the set E. The activity-on-node graph G depicts the activities $i \in V$ as the set of nodes and the completion-start precedence relationships $(i, j) \in E$ as the set of directed arcs between the nodes. The set TE consists of all pairs of activities $(i, j) \in V \times V$ for which the graph G contains a path from i to j or from j to i.

The planning horizon T is calculated as the sum of the maximum durations of all activities, i.e., $T = \sum_{i \in V} p_i^{max}$, where p_i^{max} denotes the maximum duration of activity $i \in V$, i.e., $p_i^{max} = \max_{m \in M_i}\{p_{im}\}$. The earliest start time ES_i and the latest start time LS_i of the activities $i \in V$ are determined by forward and backward pass calculations (cf. Demeulemeester and Herroelen 2002), respectively; for each activity, the mode with the shortest duration is used for both passes (cf. Talbot 1982).

Figure 2.1: Illustrative example: completion-start precedence relationships, durations and resource requirements.

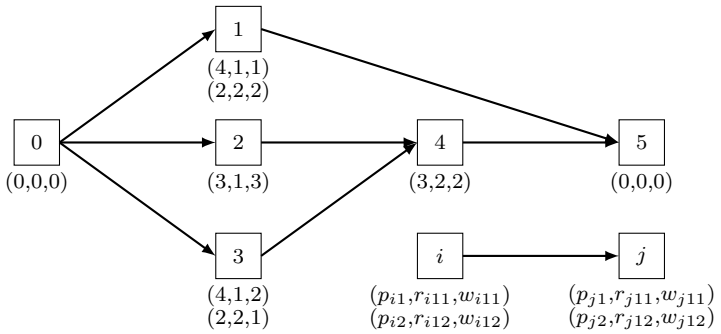


Table 2.2: Illustrative example: earliest and latest start times.

Activity i	0	1	2	3	4	5
Earliest start time ES_i	0	0	0	0	3	6
Latest start time LS_i	8	12	8	9	11	14

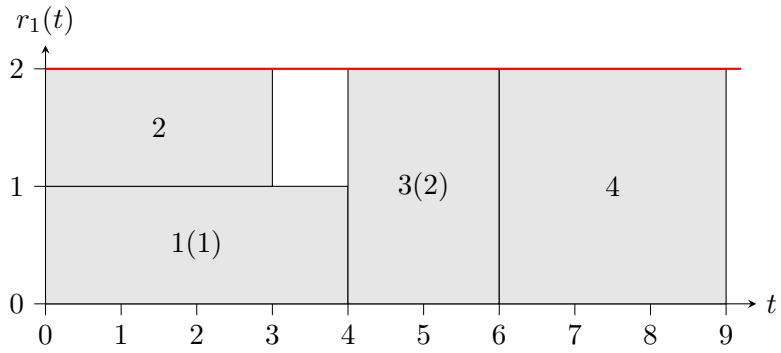
2.2.2 Illustration of the planning problem

We illustrate the MRCPSP by means of an illustrative example. Given are four real activities, i.e., $V = \{0, 1, \ldots, 4, 5\}$; two execution modes are given for the activities $i \in \{1, 3\}$, and one execution mode is given for the activities $i \in \{0, 2, 4, 5\}$. The activities require one renewable resource with a capacity of two units, i.e., $R = \{1\}$ and $R_1 = 2$, and one non-renewable resource with a capacity of eight units, i.e., $N = \{1\}$ and $W_1 = 8$, for their execution. Figure 2.1 shows the activity-on-node graph G that depicts the completion-start precedence relationships among the activities and, below each node, the mode-dependent durations and resource requirements for each activity. The earliest and the latest start times are shown in Table 2.2; they are calculated based on the given precedence relationships and the planning horizon $T = 14$. In Figure 2.2, an optimal project schedule is shown; the usage for the renewable resource $r_1(t)$ is depicted as a function of the time t, and the selected execution modes are indicated in brackets. The minimal project makespan is nine time periods.

2.3 MILP models from the literature

In this section, we describe the types of variables used in the DT and CT models that we use as reference models. We selected the model of Talbot (1982), because it is the

Figure 2.2: Illustrative example: optimal project schedule.



most commonly used DT model (cf., e.g., Mika et al. 2015), and the two CT models of Kyriakidis et al. (2012), because they are, to the best of our knowledge, the only known CT models for the MRCPSP.

The DT model introduced by Talbot (1982) is an extension of the well-known model proposed by Pritsker et al. (1969) for the single-mode RCPSP. The planning horizon is divided into a set of equal-length time intervals. The model employs some time-indexed variables that indicate whether an activity starts at the beginning of a specific time interval. Furthermore, each of these variables has an additional index that indicates the selected execution mode for the activity. In the following, we will refer to the model of Talbot (1982) as Tal82.

In the two CT models presented by Kyriakidis et al. (2012), the planning horizon is divided into a set of time intervals with variable length. The first model (referred to as MMRTN1) involves two types of binary variables that are used to indicate whether an activity starts at the beginning of a specific time interval and whether an activity spans over multiple consecutive time intervals. Furthermore, three types of continuous variables are used to model the variable length of the planning horizon, the variable length of the time intervals and the resource availability at the beginning of each time interval. The second model (referred to as MMRTN2) employs similar types of variables as the first model; furthermore, two additional types of variables are introduced to indicate for each activity the resource usage at the beginning and the completion of the activity and to express the selected execution mode for each activity. In the following, we will refer to the first and the second model of Kyriakidis et al. (2012) as KyrKopGeo12-1 and KyrKopGeo12-2, respectively.

Table 2.3: Variables used in the MCTAB model.

Variable	Description
S_i	Start time of activity i
y_{ij}	$\begin{cases} = 1, \text{ if activity } i \text{ must be completed before the start of activity } j \\ = 0, \text{ otherwise} \end{cases}$
r_{ik}^l	$\begin{cases} = 1, \text{ if activity } i \text{ is assigned to unit } l \text{ of renewable resource } k \\ = 0, \text{ otherwise} \end{cases}$
x_{im}	$\begin{cases} = 1, \text{ if activity } i \text{ is executed in mode } m \\ = 0, \text{ otherwise} \end{cases}$

2.4 Novel MILP models for the MRCPSP

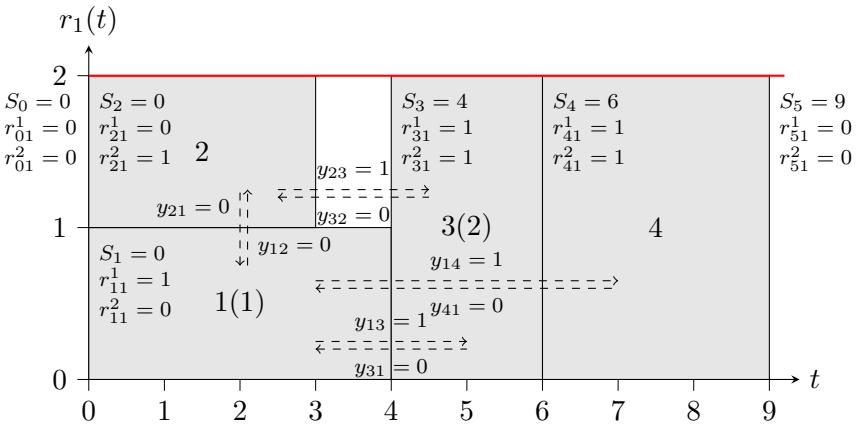
In this section, we present the two novel continuous-time assignment-based models for the MRCPSP. In Subsection 2.4.1, we present the model without auxiliary resource-overlap variables and in Subsection 2.4.2 the model with auxiliary resource-overlap variables. In Subsection 2.4.3, we present some supplements for the two models which shall improve their performance.

2.4.1 Continuous-time assignment-based model without auxiliary variables

The continuous-time assignment-based model, hereafter referred to as the MCTAB model, is based on the four types of variables listed in Table 2.3; a preliminary version of this model has been presented in Gnägi et al. (2018a). The model employs the start-time variables S_i $(i \in V)$ and the sequencing variables y_{ij} $(i, j \in V : i \neq j, (i, j) \notin TE)$; similar variables have been used in the model proposed by Artigues et al. (2003) for the single-mode RCPSP. The sequencing variables y_{ij} are only defined for all pairs of activities (i, j) with $i \neq j$ which can be executed simultaneously with respect to the precedence relationships, i.e., $(i, j) \notin TE$. Furthermore, the model includes the resource-assignment variables r_{ik}^l $(i \in V;\ k \in R;\ l = 1, \ldots, R_k)$ that explicitly assign activities to individual renewable-resource units. Similar variables are used in Correia et al. (2012) to model the multi-skill project scheduling problem and in Rihm et al. (2018) to model an assessment center planning problem; both of these problems are extensions of the single-mode RCPSP. Finally, the binary mode-selection variables x_{im} $(i \in V;\ m \in M_i)$ are used to indicate the selected execution mode for each activity. We illustrate the types of variables used in the MCTAB model in Figure 2.3 by means of our illustrative example.

The objective is to minimize the project makespan, i.e. the start time of the dummy

Figure 2.3: Types of variables used in the MCTAB model.



activity $n + 1$:

$$\text{Min. } S_{n+1}$$

Constraints (2.1) link the resource-assignment variables to the sequencing variables. If two activities i and j are assigned to the same resource unit, these activities must be scheduled sequentially, i.e, either $y_{ij} = 1$ or $y_{ji} = 1$.

$$r^l_{ik} + r^l_{jk} \leq 1 + y_{ij} + y_{ji} \quad (i, j \in V;\ k \in R;\ l = 1, \ldots, R_k : i < j, (i, j) \notin TE) \tag{2.1}$$

Constraints (2.2) indicate that for each activity the number of assigned renewable-resource units is equal to the number required by the activity in its selected mode.

$$\sum_{l=1}^{R_k} r^l_{ik} = \sum_{m \in M_i} r_{ikm} x_{im} \quad (i \in V;\ k \in R) \tag{2.2}$$

Constraints (2.3) ensure that the capacities of the non-renewable resources are not exceeded.

$$\sum_{i \in V} \sum_{m \in M_i} w_{ikm} x_{im} \leq W_k \quad (k \in N) \tag{2.3}$$

Constraints (2.4) represent the precedence relationships among the activities.

$$S_i + \sum_{m \in M_i} p_{im} x_{im} \leq S_j \quad ((i, j) \in E) \tag{2.4}$$

Constraints (2.5) link the sequencing variables and the start-time variables. If two activities i and j are assigned to the same resource unit and are therefore forced to be

scheduled sequentially by the constraints (2.1), then either activity i must be completed before the start of activity j, i.e, $y_{ij} = 1$, or activity j must be completed before the start of activity i, i.e, $y_{ji} = 1$.

$$S_i + \sum_{m \in M_i} p_{im} x_{im} \leq S_j + (LS_i + p_i^{max} - ES_j)(1 - y_{ij}) \quad (i, j \in V : i \neq j, (i, j) \notin TE) \quad (2.5)$$

Constraints (2.6) assure that each activity is executed in exactly one mode.

$$\sum_{m \in M_i} x_{im} = 1 \quad (i \in V) \quad (2.6)$$

The model reads as follows.

$$\text{(MCTAB)} \begin{cases} \text{Min. } S_{n+1} \\ \text{s.t. } (2.1) - (2.6) \\ \quad S_i \geq 0 \quad (i \in V) \\ \quad y_{ij} \in \{0, 1\} \quad (i, j \in V : i \neq j, (i, j) \notin TE) \\ \quad r_{ik}^l \in \{0, 1\} \quad (i \in V;\ k \in R;\ l \in \{1, \ldots, R_k\}) \\ \quad x_{im} \in \{0, 1\} \quad (i \in V;\ m \in M_i) \end{cases}$$

2.4.2 Model with auxiliary resource-overlap variables

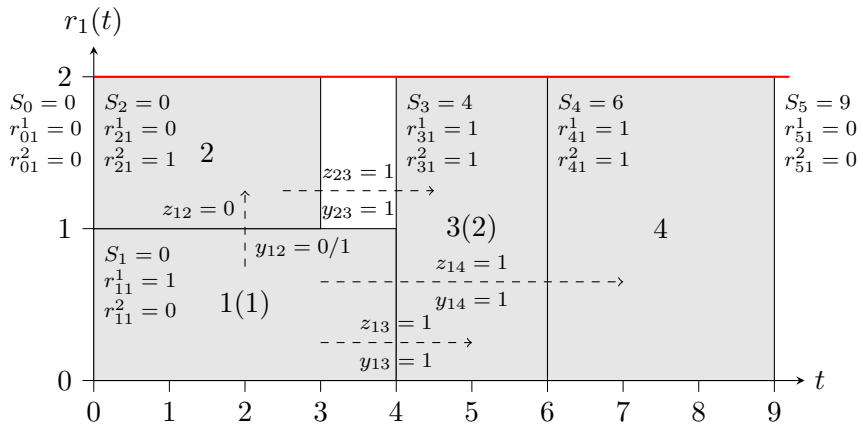
The model with auxiliary resource-overlap variables, hereafter referred to as the MCTABO model, is based on the five types of variables listed in Table 2.4. The start-time variables, the resource-assignment variables and the mode-selection variables are used analogously to the MCTAB model. In contrast, the sequencing variables y_{ij} are used such that $y_{ij} = 1$ if activity i starts before or at the same time as activity j and $y_{ij} = 0$ if activity j starts before or at the same time as activity i. Furthermore, the auxiliary resource-overlap variables z_{ij} $(i, j \in V : i < j, (i, j) \notin TE)$ are used to indicate a possible overlap between the activities i and j with regard to at least one renewable resource. The sequencing variables y_{ij} and the auxiliary resource-overlap variables z_{ij} are only defined for all pairs of activities (i, j) with $i < j$ which can be executed simultaneously with respect to the precedence relationships, i.e., $(i, j) \notin TE$. We illustrate the types of variables used in the MCTABO model in Figure 2.4 by means of our illustrative example.

With regard to the constraints, the differences between the MCTAB model and the MCTABO model are as follows. Constraints (2.1) are replaced by constraints (2.7).

Table 2.4: Variables used in the MCTABO model.

Variable	Description
S_i	Start time of activity i
y_{ij}	$\begin{cases} = 1, \text{ if activity } i \text{ starts before or at the same time as activity } j \\ = 0, \text{ if activity } j \text{ starts before or at the same time as activity } i \end{cases}$
r_{ik}^l	$\begin{cases} = 1, \text{ if activity } i \text{ is assigned to unit } l \text{ of renewable resource } k \\ = 0, \text{ otherwise} \end{cases}$
z_{ij}	$\begin{cases} = 1, \text{ if activities } i \text{ and } j \text{ use the same renewable-resource unit} \\ = 0, \text{ otherwise} \end{cases}$
x_{im}	$\begin{cases} = 1, \text{ if activity } i \text{ is executed in mode } m \\ = 0, \text{ otherwise} \end{cases}$

Figure 2.4: Types of variables used in the MCTABO model.



These constraints link the resource-assignment variables to the auxiliary resource-overlap variables. If two activities i and j are assigned to the same resource unit, then the corresponding auxiliary resource-overlap variable is forced to be one, i.e., $z_{ij} = 1$.

$$r_{ik}^l + r_{jk}^l \leq 1 + z_{ij} \quad (i, j \in V;\ k \in R;\ l = 1, \ldots, R_k : i < j, (i, j) \notin TE) \tag{2.7}$$

Constraints (2.5) are replaced by constraints (2.8) and (2.9). These constraints link the start-time variables, the sequencing variables and the auxiliary resource-overlap variables. If two activities i and j are assigned to the same resource unit, i.e., $z_{ij} = 1$, then these activities must be scheduled sequentially; then, either activity i must be completed before the start of activity j, i.e, $y_{ij} = 1$, or activity j must be completed before the start of activity i, i.e, $y_{ij} = 0$. If there is no resource overlap between the activities i and j, i.e., $z_{ij} = 0$, then either activity i must start before or at the same time as activity j, i.e,

$y_{ij} = 1$, or activity j must start before or at the same time as activity i, i.e, $y_{ij} = 0$.

$$S_i + \sum_{m \in M_i} p_{im} x_{im} - p_i^{max}(1 - z_{ij}) \leq S_j + (LS_i + p_i^{max} - ES_j)(1 - y_{ij})$$
$$(i, j \in V : i < j, (i, j) \notin TE) \quad (2.8)$$

$$S_j + \sum_{m \in M_j} p_{jm} x_{jm} - p_j^{max}(1 - z_{ij}) \leq S_i + (LS_j + p_j^{max} - ES_i) y_{ij}$$
$$(i, j \in V : i < j, (i, j) \notin TE) \quad (2.9)$$

The model reads as follows.

$$(\text{MCTABO}) \begin{cases} \text{Min. } S_{n+1} \\ \text{s.t. } (2.2) - (2.4), (2.6) - (2.9) \\ S_i \geq 0 \quad (i \in V) \\ y_{ij} \in \{0, 1\} \quad (i, j \in V : i < j, (i, j) \notin TE) \\ r_{ik}^l \in \{0, 1\} \quad (i \in V;\ k \in R;\ l \in \{1, \ldots, R_k\}) \\ z_{ij} \in \{0, 1\} \quad (i, j \in V : i < j, (i, j) \notin TE) \\ x_{im} \in \{0, 1\} \quad (i \in V;\ m \in M_i) \end{cases}$$

2.4.3 Model supplements

In this subsection, we present various supplements for the models MCTAB and MCTABO. These supplements shall improve the performance of the models by eliminating some symmetric solutions from the search space and by adding some redundant constraints that explicitly enforce the sequential scheduling of pairs or triples of activities that cannot be processed in parallel due to the resource capacities.

First, to eliminate some symmetric solutions from the search space w.l.o.g., we can assign ex ante some units of each renewable resource to an arbitrary activity, because all individual units of a renewable resource are identical. For each renewable resource $k \in R$, we select an activity i_k^* with the largest minimum requirement $r_{ik}^{min} = \min_{m \in M_i}\{r_{ikm}\}$ for this resource, and we add the constraints (2.10) to both models MCTAB and MCTABO; for each renewable resource $k \in R$, these constraints assign the first $r_{i_k^*,k}^{min}$ renewable-resource units to the execution of activity i_k^*.

$$r_{i_k^*,k}^l = 1 \quad (k \in R, l \in \{1, \ldots, r_{i_k^*,k}^{min}\}) \quad (2.10)$$

Figure 2.5: Elimination of some symmetric resource-unit assignments.

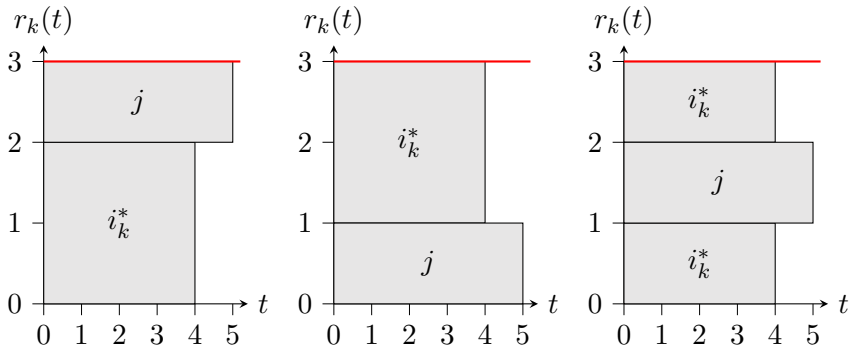


Figure 2.5 illustrates the constraints (2.10); by explicitly assigning the first and the second renewable-resource unit to activity i_k^*, the two symmetric resource-unit assignments in the middle and on the right-hand side are eliminated from the search space.

Second, we analyze all pairs of activity-mode combinations $((i, m_i), (j, m_j))$ with $i, j \in V$, $m_i \in M_i$ and $m_j \in M_j$ for which the requirement for some renewable resource exceeds the resource capacity, i.e., $r_{ikm_i} + r_{jkm_j} > R_k$ for some renewable resource $k \in R$; let the set V^2 contain all these pairs of activity-mode combinations with $i < j$, excluding all pairs of activity-mode combinations that contain pairs of activities that are in TE. For all pairs of activity-mode combinations $((i, m_i), (j, m_j)) \in V^2$, if activities i and j are executed in modes m_i and m_j, respectively, in each feasible solution, at least one unit of resource k will be assigned to both activities i and j. Thus, the activities i and j must be scheduled sequentially; therefore, we add the (redundant) constraints

$$y_{ij} + y_{ji} \geq x_{im_i} + x_{jm_j} - 1 \quad (((i, m_i), (j, m_j)) \in V^2) \tag{2.11}$$

to the MCTAB model and the (redundant) constraints

$$z_{ij} \geq x_{im_i} + x_{jm_j} - 1 \quad (((i, m_i), (j, m_j)) \in V^2) \tag{2.12}$$

to the MCTABO model. Analogously, we analyze all triples of activity-mode combinations $((i, m_i), (j, m_j), (h, m_h))$, where $i, j, h \in V$, $m_i \in M_i$, $m_j \in M_j$ and $m_h \in M_h$, with $r_{ikm_i} + r_{jkm_j} + r_{hkm_h} > R_k$ for some renewable resource $k \in R$; let the set V^3 contain all these triples of activity-mode combinations with $i < j < h$, excluding all triples of activity-mode combinations that contain pairs of activities that are in TE or pairs of activity-mode combinations that are in V^2. For all triples of activity-mode combinations $((i, m_i), (j, m_j), (h, m_h)) \in V^3$, we add the (redundant) constraints (2.13) and (2.14) to

the models MCTAB and MCTABO, respectively.

$$y_{ij} + y_{ji} + y_{ih} + y_{hi} + y_{jh} + y_{hj} \geq x_{im_i} + x_{jm_j} + x_{hm_h} - 2 \quad (((i,m_i),(j,m_j),(h,m_h)) \in V^3) \quad (2.13)$$

$$z_{ij} + z_{ih} + z_{jh} \geq x_{im_i} + x_{jm_j} + x_{hm_h} - 2 \quad (((i,m_i),(j,m_j),(h,m_h)) \in V^3) \quad (2.14)$$

The extended models read as follows.

$$\text{(MCTAB extended)} \begin{cases} \text{Min. } S_{n+1} \\ \text{s.t. } (2.1) - (2.6), (2.10) - (2.11), (2.13) \\ \quad S_i \geq 0 \quad (i \in V) \\ \quad y_{ij} \in \{0,1\} \quad (i,j \in V : i \neq j, (i,j) \notin TE) \\ \quad r_{ik}^l \in \{0,1\} \quad (i \in V;\ k \in R;\ l \in \{1,\ldots,R_k\}) \\ \quad x_{im} \in \{0,1\} \quad (i \in V;\ m \in M_i) \end{cases}$$

$$\text{(MCTABO extended)} \begin{cases} \text{Min. } S_{n+1} \\ \text{s.t. } (2.2) - (2.4), (2.6) - (2.10), (2.12), (2.14) \\ \quad S_i \geq 0 \quad (i \in V) \\ \quad y_{ij} \in \{0,1\} \quad (i,j \in V : i < j, (i,j) \notin TE) \\ \quad r_{ik}^l \in \{0,1\} \quad (i \in V;\ k \in R;\ l \in \{1,\ldots,R_k\}) \\ \quad z_{ij} \in \{0,1\} \quad (i,j \in V : i < j, (i,j) \notin TE) \\ \quad x_{im} \in \{0,1\} \quad (i \in V;\ m \in M_i) \end{cases}$$

2.5 Computational results

In this section, we present the design (cf. Subsection 2.5.1) and the numerical results (cf. Subsection 2.5.2) of the experimental performance analysis.

2.5.1 Test design

We compare the performance of the novel MILP models with the performance of the DT model of Talbot (1982) and the two CT models of Kyriakidis et al. (2012). We

implemented these models in AMPL, and we used the Gurobi Optimizer 8.1 with the default solver settings. All computations were performed on a workstation with two 8-core Intel Xeon E5-2687W CPUs (3.1 GHz) and 128 GB RAM. We set a time limit of 300 seconds per instance, and we limited the number of threads to four.

For the comparative analysis, we used the test sets J20 and J30 from the PSPLIB (cf. Kolisch and Sprecher 1996). Each of these test sets contains 640 instances that consist of 20 and 30 real activities, respectively, with three alternative execution modes each. Furthermore, the activities require two renewable and two non-renewable resources for their execution. For 86 instances of the set J20 and for 88 instances of the set J30, no feasible solution exists; we excluded these instances from our comparative analysis. The instances of the sets J20 and J30 consist of activities with relatively short durations ranging from one to ten time units; this could be an advantage for time-indexed models such as the model of Talbot (1982). Therefore, to broaden our comparative analysis, we generated two novel test sets that are based on the instances of the sets J20 and J30, but consist of instances with a relatively large range of the activities' durations. We generated these instances by adopting the procedure that has been proposed by Koné et al. (2011) for single-mode RCPSP instances. For each of the sets J20 and J30, we randomly selected α real activities. For these activities, we multiplied the duration for each mode by $\beta + \varepsilon$ with ε being a uniformly distributed random number between zero and one and rounded the resulting duration to the nearest integer. Analogously to Koné et al. (2011), we set $\alpha = n/2$, i.e., $\alpha = 10$ for set J20 and $\alpha = 15$ for set J30, and we set $\beta = 25$ resulting in two novel sets of instances consisting of activities with durations ranging from one to up to 260 time units. In the following, we will refer to these novel test sets as D20 and D30.

2.5.2 Numerical results

We use the following metrics to evaluate the performance of the tested models:

- Feas (%): Fraction of instances for which a feasible schedule has been found within the prescribed time limit.
- Opt (%): Fraction of instances for which a feasible schedule has been found and proven to be optimal within the prescribed time limit.
- Best (%): Fraction of instances for which, within the prescribed time limit, the best schedule among those found with all tested models has been found.
- Gap^{LB} (%): Average relative deviation between the objective function value of the best schedule returned by the solver (OFV) and the best lower bound returned by

the solver (LB), calculated as $(OFV - LB)/LB$.

- Gap^{CP} (%): Average relative deviation between OFV and the critical-path based lower bound (CP), calculated as $(OFV - CP)/CP$.
- Gap^{BEST} (%): Average relative deviation between OFV and the objective function value of the best schedule ($BEST$) found among those found with all tested models, calculated as $(OFV - BEST)/BEST$.
- # Cons: Average number of constraints.
- # Vars: Average number of variables.

The results of the comparative analysis are summarized in Tables 2.5 and 2.6; bold values indicate the best results among all tested models.

For the sets J20 and J30, all tested models except the two models proposed by Kyriakidis et al. (2012) obtain at least a feasible schedule for all considered instances. The model of Talbot (1982) performs best in terms of the number of instances that are proven to be solved to optimality and of the average deviation from the best lower bound returned by the solver; this may be attributed to the relatively strong linear programming relaxations of models with time-indexed binary variables (cf. Artigues et al. 2015). The novel models, and especially those including the proposed supplements, perform second best for the sets J20 and J30; they even achieve a matchable performance in terms of the average deviation from the critical-path based lower bound.

For the sets D20 and D30, the novel models clearly outperform the other tested models regarding all reported metrics; this might be attributed to the strong increase of the average number of variables used for the DT model of Talbot (1982). In contrast, the average number of variables used for the presented novel CT models is relatively small and remains constant also for the sets D20 and D30. For the CT models proposed by Kyriakidis et al. (2012), the average number of variables used is also constant but very large; both models are clearly outperformed by the other tested models. In Table 2.7, we provide the results of the comparative analysis for all considered instances of the sets D20 and D30 for which both the model of Talbot (1982) and the novel models yield at least a feasible solution; the results for the models of Kyriakidis et al. (2012) are omitted because they do not obtain a feasible solution for many instances, especially for the set D30. Also with respect to these results, the novel models outperform the model of Talbot (1982).

Without the model supplements, both models MCTAB and MCTABO achieve a comparable performance for all test sets. For both models, the proposed sup-

Table 2.5: Computational results: all considered instances of the sets J20 and J30.

Set	Model	Feas (%)	Opt (%)	Best (%)	Gap^{LB} (%)	Gap^{CP} (%)	Gap^{BEST} (%)	# Cons	# Vars
J20	Tal82	**100.00**	**97.47**	**99.10**	**0.22**	**17.13**	**0.05**	**382**	8,676
	KyrKopGeo12-1	96.75	0.00	8.84	607.97	45.55	24.19	6,159	3,513
	KyrKopGeo12-2	99.64	5.96	43.68	118.48	25.19	6.31	22,229	18,623
	MCTAB	**100.00**	81.41	88.27	3.05	17.90	0.59	4,310	**1,223**
	MCTAB extended	**100.00**	88.99	96.21	1.71	17.30	0.17	4,841	**1,223**
	MCTABO	**100.00**	80.69	89.35	3.20	17.91	0.59	4,310	**1,223**
	MCTABO extended	**100.00**	85.74	92.06	2.47	17.53	0.33	4,841	**1,223**
J30	Tal82	**100.00**	**88.41**	**97.64**	**2.50**	**15.35**	**0.81**	**570**	19,859
	KyrKopGeo12-1	9.24	0.00	0.00	1,558.85	110.03	81.19	13,993	7,881
	KyrKopGeo12-2	47.28	0.00	5.62	383.02	34.23	27.98	64,939	57,229
	MCTAB	**100.00**	74.09	77.54	8.06	15.95	1.79	13,495	**2,328**
	MCTAB extended	**100.00**	76.09	79.89	7.85	15.77	1.71	15,306	**2,328**
	MCTABO	**100.00**	73.55	78.26	8.15	15.93	1.80	13,495	**2,328**
	MCTABO extended	**100.00**	75.54	78.99	7.87	15.75	1.67	15,306	**2,328**

plements result in a considerable performance improvement with regard to all reported metrics. The supplements proposed for the MCTAB model, however, tend to be more effective and thus lead to slightly larger improvements.

In Table 2.8, the results for further supplements of the novel models are summarized, representatively for the set J30. We tested the performance of the models MCTAB extended and MCTABO extended considering pairs of activity-mode combinations that cannot be process in parallel due their resource requirements only, i.e., without the constraints (2.13) and (2.14), respectively. Furthermore, we also tested the performance of the models MCTAB extended and MCTABO extended with analogous additional constraints for quadruples of activity-mode combinations that cannot be process in parallel due their resource requirements. The models MCTAB extended and MCTABO extended perform best among all tested models; this may be attributed to the relatively small number of redundant constraints when considering pairs of activity-mode combinations only, while the large number of redundant constraints for the models also considering quadruples of activity-mode combinations seems to slow down the solution process.

Table 2.6: Computational results: all considered instances of the sets D20 and D30.

Set	Model	Feas (%)	Opt (%)	Best (%)	Gap^{LB} (%)	Gap^{CP} (%)	Gap^{BEST} (%)	# Cons	# Vars
D20	Tal82	87.36	73.83	74.19	34.64	42.34	28.30	**4,282**	113,440
	KyrKopGeo12-1	73.10	0.00	9.57	580.52	38.66	25.51	6,159	3,513
	KyrKopGeo12-2	**100.00**	2.71	65.34	71.54	12.99	2.02	22,229	18,623
	MCTAB	**100.00**	95.49	99.46	0.16	10.37	0.01	4,310	**1,223**
	MCTAB extended	**100.00**	**99.82**	**100.00**	**0.01**	**10.36**	**0.00**	4,841	**1,223**
	MCTABO	**100.00**	93.50	98.01	0.32	10.38	0.02	4,310	**1,223**
	MCTABO extended	**100.00**	99.28	99.82	0.10	**10.36**	**0.00**	4,841	**1,223**
D30	Tal82	71.92	63.04	63.04	45.03	47.06	39.40	**6,434**	260,446
	KyrKopGeo12-1	3.99	0.00	0.00	857.53	43.33	41.67	13,993	7,881
	KyrKopGeo12-2	23.91	0.00	4.17	209.71	19.76	17.88	64,939	57,229
	MCTAB	**100.00**	84.06	87.86	3.48	8.38	0.48	13,495	**2,328**
	MCTAB extended	**100.00**	**87.14**	**93.30**	**2.69**	**7.96**	**0.16**	15,306	**2,328**
	MCTABO	**100.00**	84.42	89.13	3.51	8.21	0.34	13,495	**2,328**
	MCTABO extended	**100.00**	86.96	92.75	2.81	8.00	0.19	15,306	**2,328**

Table 2.7: Computational results: instances of the sets D20 and D30 with at least a feasible solution obtained by all considered models (Tal82 and novel models).

Set	Model	Feas (%)	Opt (%)	Best (%)	Gap^{LB} (%)	Gap^{CP} (%)	Gap^{BEST} (%)	# Cons	# Vars
D20	Tal82	**100.00**	84.50	84.92	34.64	42.34	28.30	**4,280**	113,377
	MCTAB	**100.00**	96.49	99.59	0.12	6.26	0.01	4,460	**1,257**
	MCTAB extended	**100.00**	**100.00**	**100.00**	**0.00**	**6.25**	**0.00**	4,889	**1,257**
	MCTABO	**100.00**	96.28	99.17	0.24	6.27	0.01	4,460	**1,257**
	MCTABO extended	**100.00**	99.79	99.79	0.05	**6.25**	**0.00**	4,889	**1,257**
D30	Tal82	**100.00**	87.66	87.66	45.03	47.06	39.40	**6,426**	259,873
	MCTAB	**100.00**	94.96	95.97	0.89	1.93	0.16	14,263	**2,426**
	MCTAB extended	**100.00**	**96.22**	97.23	**0.63**	**1.80**	**0.05**	15,084	**2,426**
	MCTABO	**100.00**	94.96	97.23	0.84	1.82	0.08	14,263	**2,426**
	MCTABO extended	**100.00**	95.97	**98.24**	0.66	**1.80**	**0.05**	15,084	**2,426**

Table 2.8: Computational results: models MCTAB and MCTABO with further model supplements.

Set	Model	Feas (%)	Opt (%)	Best (%)	Gap^{LB} (%)	Gap^{CP} (%)	Gap^{BEST} (%)	# Cons	# Vars
J30	MCTAB	**100.00**	74.09	82.79	8.06	15.95	0.94	**13,495**	**2,328**
	MCTAB (pairs)	**100.00**	74.64	**84.60**	**7.58**	**15.69**	**0.80**	13,554	**2,328**
	MCTAB extended	**100.00**	**76.09**	84.24	7.85	15.77	0.85	15,306	**2,328**
	MCTAB (quadruples)	**100.00**	75.54	80.62	9.81	17.62	1.97	24,559	**2,328**
	MCTABO	**100.00**	73.55	83.51	8.15	15.93	0.95	**13,495**	**2,328**
	MCTABO (pairs)	**100.00**	73.73	83.70	**7.75**	15.78	0.87	13,554	**2,328**
	MCTABO extended	**100.00**	**75.54**	**84.42**	7.87	**15.75**	**0.81**	15,306	**2,328**
	MCTABO (quadruples)	99.82	75.18	82.61	8.23	16.06	1.16	24,559	**2,328**

2.6 Conclusions

In this paper, we have proposed two novel continuous-time MILP models for the multi-mode resource-constrained project scheduling problem MRCPSP. The models are based on continuous variables that represent the start times of the activities and binary variables that represent the assignment of the project activities to the individual resource units, the sequential relationships between activities that are assigned to at least one identical resource unit, and the selection of an execution mode for each activity. Compared to the continuous-time models known from the literature, the novel models have a simpler structure. Our computational results indicate that the novel models outperform all reference models when the range of the activities' durations is relatively high.

In future research, the efficient elimination of additional symmetric solutions from the search space should be investigated. Furthermore, analogous models for related project scheduling problems such as, e.g., the resource-constrained project scheduling problem with minimum and maximum time lags, should be analyzed.

Bibliography

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

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

Baker, K.R., Keller, B., 2010. Solving the single-machine sequencing problem using integer programming. Computers & Industrial Engineering 59, 730–735.

Bartels, J.H., Zimmermann, J., 2015. Scheduling tests in automotive R&D projects using a genetic algorithm, in: Schwindt, C., Zimmermann, J. (Eds.), Handbook on Project Management and Scheduling Vol. 2. Springer, Cham, pp. 1157–1185.

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

Cheng, J., Fowler, J., Kempf, K., Mason, S., 2015. Multi-mode resource-constrained project scheduling problems with non-preemptive activity splitting. Computers & Operations Research 53, 275–287.

Correia, I., Lourenço, L.L., Saldanha-da Gama, F., 2012. Project scheduling with flexible resources: formulation and inequalities. OR Spectrum 34, 635–663.

Demeulemeester, E.L., Herroelen, W., 2002. Project Scheduling: a research handbook. Kluwer Academic Publishers, Boston.

Gnägi, M., Rihm, T., Trautmann, N., 2018a. A continuous-time MILP model for the multi-mode resource-constrained project scheduling problem, in: 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Bangkok. To appear.

Gnägi, M., Zimmermann, A., Trautmann, N., 2018b. A novel assignment-based continuous-time MILP model for the resource-constrained project scheduling problem, in: 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Bangkok. To appear.

Keha, A.B., Khowala, K., Fowler, J.W., 2009. Mixed integer programming formulations for single machine scheduling problems. Computers & Industrial Engineering 56, 357–367.

Knust, S., 2010. Scheduling non-professional table-tennis leagues. European Journal of Operational Research 200, 358–367.

Kolisch, R., Sprecher, A., 1996. PSPLIB—a project scheduling problem library. European Journal of Operational Research 96, 205–216.

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

Kyriakidis, T.S., Kopanos, G.M., Georgiadis, M.C., 2012. MILP formulations for single- and multi-mode resource-constrained project scheduling problems. Computers & Chemical Engineering 36, 369–385.

Maniezzo, V., Mingozzi, A., 1999. A heuristic procedure for the multi-mode project scheduling problem based on benders' decomposition, in: Węglarz, J. (Ed.), Project Scheduling: Recent Models, Algorithms and Applications. Springer, Boston, pp. 179–196.

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

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

Rihm, T., Trautmann, N., Zimmermann, A., 2018. MIP formulations for an application of project scheduling in human resource management. Flexible Services and Manufacturing Journal 30, 609–639.

Sabzehparvar, M., Seyed-Hosseini, S.M., 2008. A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent time lags. The Journal of Supercomputing 44, 257–273.

Słowiński, R., 1980. Two approaches to problems of resource allocation among project activities – a comparative study. Journal of the Operational Research Society 31, 711–723.

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

Trautmann, N., Rihm, T., Saner, N.J., Zimmermann, A., 2018. A continuous-time assignment-based MILP formulation for the resource-constrained project scheduling problem, in: Caramia, M., Bianco, L., Giordani, S. (Eds.), Proceedings of the 16th International Conference on Project Management and Scheduling, Rome. pp. 242–245.

Unlu, Y., Mason, S.J., 2010. Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. Computers & Industrial Engineering 58, 785–800.

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

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

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

Zhu, G., Bard, J.F., Yu, G., 2006. A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. INFORMS Journal on Computing 18, 377–390.

Paper III

A matheuristic for large-scale capacitated clustering[3]

Mario Gnägi Philipp Baumann

Department of Business Administration
University of Bern

Contents

[3]Paper under review, submitted to Computers & Operations Research

Abstract

Clustering addresses the problem of grouping similar objects into clusters. Since the size of the clusters is often constrained in practical clustering applications, various capacitated clustering problems have received increasing attention. We consider here the capacitated p-median problem (CPMP) in which p objects are selected as cluster centers (medians) such that the total distance from these medians to their assigned objects is minimized. Each object is associated with a weight, and the total weight in each cluster must not exceed a given capacity. Various exact and heuristic solution approaches have been proposed for the CPMP. The state-of-the-art approach performs well for instances with up to 5,000 *objects but becomes computationally expensive for instances with a much larger number of objects. Since clustering problems typically comprise a very large number of objects, we propose a matheuristic that can deal with instances comprising up to* 500,000 *objects because it employs new problem decomposition strategies. In a computational experiment, the proposed matheuristic consistently outperformed the state-of-the-art approach on medium- and large-scale instances while matching the performance for small-scale instances. As an extension, we show that our matheuristic can be applied to related capacitated clustering problems, such as the capacitated centered clustering problem (CCCP). For several test instances of the CCCP, our matheuristic found new best-known solutions.*

3.1 Introduction

Clustering is the task of assigning similar objects to groups (clusters), where the similarity between a pair of objects is determined by a distance measure based on features of the objects. Since clustering is used in many different domains for a broad range of applications, numerous different clustering problems have been discussed in the literature. The widely studied p-median problem is an example of such a clustering problem. This problem consists of selecting a given number of p objects as cluster centers (medians) such that the total distance between the objects and their nearest median is minimized. The p-median problem has been studied mainly in the context of facility location but also in other contexts, such as large-scale data mining (cf., e.g., Avella et al. 2012). In practical clustering applications, the size of the clusters is often constrained. For example, when

grouping customers to form sales force territories, the workload of an individual salesperson must be restricted to guarantee adequate service quality (cf. Mulvey and Beck 1984). This gives rise to an extension of the p-median problem, namely, the capacitated p-median problem (CPMP).

The CPMP can be stated as follows (cf., e.g., Lorena and Senne 2004). Given a set of n weighted objects that are described by d features, the goal is to form a prescribed number of p clusters by selecting p objects as medians and by assigning the objects to these medians such that the total distance (e.g., Euclidean distance) between the objects and their medians is minimized. Furthermore, for each median, the sum of the weights of the objects assigned to it must not exceed a given capacity limit. As an extension of the uncapacitated p-median problem, which is known to be NP-hard, the CPMP is also NP-hard (cf. Osman and Ahmadi 2007), and the problem of finding a feasible solution to an instance of the CPMP is NP-complete (cf. Ceselli and Righini 2005). The largest publicly available test instances for the CPMP that have been tested in the literature thus far comprise up to 5,000 objects. In contrast, existing test instances for the uncapacitated p-median problem comprise up to 100,000 objects (cf., e.g., Hansen et al. 2009). Since the CPMP is an extension of the uncapacitated p-median problem, this motivates our interest in addressing large-scale instances of the CPMP.

Several exact solution approaches (cf., e.g., Ceselli and Righini 2005; Boccia et al. 2008) and numerous heuristic solution approaches (cf., e.g., Mulvey and Beck 1984; Scheuerer and Wendolsky 2006; Stefanello et al. 2015) have been proposed for the CPMP. Existing exact approaches can solve instances with up to 1,000 objects within a reasonable running time. For instances that involve more than 1,000 objects, the iterated reduction matheuristic algorithm (IRMA) proposed by Stefanello et al. (2015) is considered the state-of-the-art approach (cf. also Jánošíková et al. 2017). The approach of Stefanello et al. (2015) iteratively constructs an initial solution with a randomized procedure and improves this initial solution by first solving a mathematical model for the entire problem and then iteratively for subproblems. Reduction heuristics are applied to eliminate variables from the models. In a comprehensive computational experiment based on instances with up to 5,000 objects, Stefanello et al. (2015) demonstrated the superior performance of their approach in comparison to recent benchmark approaches from the literature. For instances that comprise many more than 5,000 objects, however, the approach of Stefanello et al. (2015) becomes computationally expensive for three reasons. First, the randomized procedure requires many iterations to construct a good initial solution, especially when the capacity limit is tight. Second, solving the mathematical model for the entire problem becomes intractable for large-scale instances, despite the reduction

heuristics. Third, the subproblem selection procedure does not prioritize subproblems with great potential for improving the objective function value. Another challenge that was not specifically discussed by Stefanello et al. (2015) is that the number of distances between objects and potential medians grows quadratically with an increasing number of objects. For instances with much more than 5,000 objects, the computation of all these distances becomes prohibitively time consuming and exceeds the available memory of most standard workstations.

In this paper, we propose a matheuristic with new problem decomposition strategies that are specifically designed for large-scale instances. These strategies a) focus on subproblems with the potential for substantially improving the objective function value, b) exploit the power of binary linear programming to ensure the capacity constraints during the entire solution process, and c) apply efficient data structures (kd-trees; cf. Bentley 1975) to avoid computing a large number of pairwise distances. The proposed matheuristic comprises two phases: a global optimization phase in which the subproblems involve all objects and a local optimization phase in which the subproblems involve only a subset of objects. In the global optimization phase, we decompose the CPMP into a series of generalized assignment problems, which are formulated as binary linear programs and solved using a mathematical programming solver. In each of these subproblems, objects are optimally assigned to fixed medians subject to the capacity constraints. The fixed medians are updated between the solutions of two consecutive subproblems. By fixing the medians and allowing objects to be assigned only to one of their g-nearest fixed medians, the number of required distance computations is reduced from $\frac{n(n-1)}{2}$ to ng per subproblem, where parameter g can be controlled by the user. To efficiently identify the g-nearest fixed medians of each object and to compute the corresponding distances, we use kd-trees. In the local optimization phase, we decompose the entire problem into subproblems that comprise groups of clusters only. A binary linear programming formulation of the CPMP is then solved for these groups of clusters individually using a mathematical programming solver. The proposed subproblem selection procedure focuses on groups of clusters with spare capacity and thus prioritizes subproblems with the potential for substantially improving the objective function value. We also use kd-trees in the local optimization phase to substantially reduce the number of required distance computations.

In a computational experiment, we compare the performance of the proposed matheuristic to the performance of the state-of-the-art approach proposed by Stefanello et al. (2015). Furthermore, we provide the results of an exact approach based on the binary linear program presented by Lorena and Senne (2004) and a mathematical programming solver. We apply all three approaches to a set of standard test instances from the literature, including

the largest existing instances. In comparison to instances for the uncapacitated p-median problem, these largest existing instances for the CPMP are considered small-scale (cf., e.g., Avella et al. 2012). To assess the performance of the three approaches on instances that are comparable in size to large instances of the uncapacitated p-median problem, we additionally generate some medium-scale instances with up to approximately 50,000 objects and some large-scale instances with up to approximately 500,000 objects. It turns out that, for small-scale instances, the proposed matheuristic matches the performance of the state-of-the-art approach, and for medium- and large-scale instances, the proposed matheuristic consistently delivers superior results. For the largest instances, only the proposed matheuristic identifies feasible solutions within a limited running time of one hour. Furthermore, we generated some high-dimensional instances with up to approximately 800 features. The proposed matheuristic performs best among the tested approaches also for these high-dimensional instances.

As an extension, we show that the proposed matheuristic can easily be applied to other capacitated clustering problems, such as the capacitated centered clustering problem (CCCP) (cf. Negreiros and Palhano 2006). In the CCCP, the cluster centers are computed as the geometric mean of the assigned objects and are not selected among the set of objects. For the largest problem instances of the CCCP tested in this paper, we were able to find new best-known solutions.

The remainder of this paper is organized as follows. In Section 3.2, we describe the CPMP in more detail. In Section 3.3, we review the related literature. In Section 3.4, we describe the proposed matheuristic. In Section 3.5, we report the computational results. Finally, in Section 3.6, we apply the proposed matheuristic to the CCCP, and in Section 3.7, we provide some conclusions and give an outlook on future research.

3.2 Capacitated p-median problem

In this section, we describe the CPMP in more detail (cf. Subsection 3.2.1) and provide a small illustrative example (cf. Subsection 3.2.2).

3.2.1 Description of the problem

The CPMP can be stated as follows (cf., e.g., Lorena and Senne 2004). Given a set of n objects, denoted as $I = \{1, \ldots, n\}$, each object $i \in I$ is associated with a given weight q_i and is described by d features. Based on these features, a distance d_{ij} (e.g., Euclidean distance) can be computed for each pair of objects $i, j \in I$. The goal is to partition the objects into a prescribed number of p clusters by selecting p objects as cluster centers

Table 3.1: Notation used for the binary linear program (M-CPMP)

Parameters and sets		
n	Number of objects	
I	Set of objects ($I = \{1, \ldots, n\}$)	
p	Number of clusters	
d_{ij}	Distance between objects $i, j \in I$	
q_i	Weight of object $i \in I$	
Q	Capacity limit	
Decision variables		
$*$ x_{ij}	$\begin{cases} = 1, \text{ if object } i \text{ is assigned to median } j \\ = 0, \text{ otherwise} \end{cases}$	$(i, j \in I)$

(medians) and by assigning the objects to these medians such that the total distance between the medians and their assigned objects is minimized. In doing so, the total weight in each cluster must not exceed a given capacity limit Q.

The CPMP can be formulated as a binary linear program (cf. Lorena and Senne 2004); the notation used is summarized in Table 3.1. Note that an object $j \in I$ is selected as a median if it is assigned to itself, i.e., $x_{jj} = 1$.

$$
\text{(M-CPMP)} \begin{cases}
\text{Min.} & \displaystyle\sum_{i \in I} \sum_{j \in I} d_{ij} x_{ij} & & (3.1) \\
\text{s.t.} & \displaystyle\sum_{j \in I} x_{jj} = p & & (3.2) \\
& \displaystyle\sum_{j \in I} x_{ij} = 1 & (i \in I) & (3.3) \\
& \displaystyle\sum_{i \in I} q_i x_{ij} \leq Q x_{jj} & (j \in I) & (3.4) \\
& x_{ij} \in \{0, 1\} & (i, j \in I) & (3.5)
\end{cases}
$$

The objective function given in (3.1) captures the total distance between the medians and their assigned objects. Constraint (3.2) ensures that exactly p objects are selected as medians. Constraints (3.3) assure that each object is assigned to exactly one selected median. Constraints (3.4) impose the capacity limit for each object that is selected as a median. Finally, the domains of the decision variables are defined in (3.5).

The CPMP has various real-world applications that have been discussed in the literature. Many of these real-world applications arise in facility location (cf., e.g., Medaglia et al. 2009; Jánošíková et al. 2017). Other exemplary applications are the consolidation of

customer orders into truckload shipments (cf. Koskosidis and Powell 1992) and the structuring of multiprotocol-label switching networks (cf. El-Alfy 2007). For a broad overview of real-world applications of the CPMP, we refer to Ahmadi and Osman (2005).

3.2.2 Illustrative example

We present a small example to illustrate the description of the CPMP provided above. Furthermore, we use this example to illustrate the proposed matheuristic in Subsection 3.4.3.

We consider a coffeehouse chain that wants to group its stores into a given number of clusters such that stores in the same cluster are close to each other. A manager is then put in charge of each resulting cluster. The selected median of a cluster represents the store at which the office of the assigned manager should be located. To ensure that the stores within a given cluster can be managed adequately, capacity constraints are required that limit the total number of employees within a cluster.

The coffeehouse chain has $n = 15$ stores that must be grouped into $p = 4$ clusters. The coordinates (feature values) and the number of employees (weights) of the stores are given in Table 3.2. The total number of employees within a cluster is limited to $Q = 8$. The pairwise distances between the stores are calculated as Euclidean distances. In Figure 3.1, an optimal solution for the illustrative example is depicted; the objective function value (OFV) of the depicted solution is provided in the bottom-right corner. The size of a point in the figure represents the number of employees of the corresponding store. Stores that are selected as medians are indicated with a red circle, and the assignments of the stores to the medians are indicated with green lines.

3.3 Literature review

In this section, we provide an overview of existing solution approaches for the CPMP. We categorize and discuss the papers according to the types of proposed solution approaches. In Subsections 3.3.1 to 3.3.4, we review exact approaches, classic heuristics, metaheuristics, and matheuristics. Table 3.3 gives an overview of the discussed approaches and lists the number of objects of the largest instance that was used to test the corresponding approach.

Table 3.2: Coordinates and number of employees of stores in the illustrative example

Store (i)	x-coordinate	y-coordinate	Number of employees (q_i)
1	12	31	1
2	10	91	1
3	61	50	2
4	26	50	2
5	94	34	1
6	39	12	2
7	58	13	2
8	78	72	2
9	5	78	3
10	35	64	3
11	27	82	1
12	79	42	4
13	50	21	3
14	41	89	2
15	51	78	1

3.3.1 Exact approaches

Almost all papers listed in Table 3.3 provide a formulation of the CPMP as a binary linear program. These formulations can be used to solve small-scale instances to optimality by applying a mathematical programming solver such as Gurobi or CPLEX. In addition, a few problem-specific exact approaches have been proposed. Pirkul (1987) proposed a branch-and-bound algorithm for the capacitated concentrator location problem that can be adapted to the CPMP. Baldacci et al. (2002) presented an exact approach based on a set partitioning formulation of the CPMP, and Ceselli and Righini (2005) proposed a branch-and-price algorithm with different branching strategies and pricing methods. Finally, Boccia et al. (2008) developed a cutting plane algorithm based on Fenchel cuts.

These exact approaches have been used to devise provably optimal solutions for small-scale instances with up to approximately 1,000 objects (cf. Table 3.3). For instances that comprise many more than 1,000 objects, the required running time of these exact approaches becomes prohibitively large since the number of distinct clusterings grows drastically with the increasing number of objects.

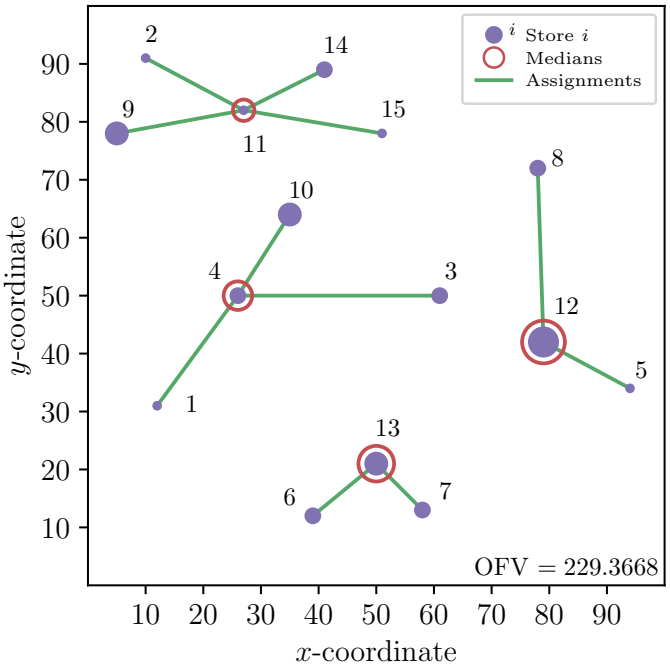
3.3.2 Classic heuristics

The category of classic heuristics comprises problem-specific heuristics that are not based on metaheuristic concepts. Mulvey and Beck (1984) proposed a classic heuristic based on alternately applying an object-assignment step and a median-update step. The objects are assigned in a greedy manner to their nearest median that has sufficient unused capacity.

Table 3.3: Solution approaches for the CPMP

Paper	Exact approach	Classic heuristic	Meta-heuristic	Math-heuristic	Largest instance (n)
Mulvey and Beck (1984)		✓			100
Pirkul (1987)	✓				100
Koskosidis and Powell (1992)		✓			100
Osman and Christofides (1994)			✓		100
Maniezzo et al. (1998)			✓		100
Baldacci et al. (2002)	✓				200
Lorena and Senne (2003)		✓			402
Ahmadi and Osman (2004)			✓		150
Ahmadi and Osman (2005)			✓		150
Ceselli and Righini (2005)	✓				900
Díaz and Fernandez (2006)			✓		737
Scheuerer and Wendolsky (2006)			✓		402
Chaves et al. (2007)			✓		402
Osman and Ahmadi (2007)			✓		150
Boccia et al. (2008)	✓				402
Fleszar and Hindi (2008)				✓	402
Landa-Torres et al. (2012)			✓		500
Yaghini et al. (2013)				✓	200
Stefanello et al. (2015)				✓	4,461
Jánošíková et al. (2017)				✓	3,038
Proposed approach				✓	498,378

Figure 3.1: Optimal solution of the illustrative example



The order in which the objects are assigned is determined based on a regret value that is computed for each object. The regret value is defined as the absolute value of the difference in distance between an object's first and its second nearest fixed median. Furthermore, an improvement heuristic based on local switches of objects between clusters was proposed. The approach of Mulvey and Beck (1984) was extended in Koskosidis and Powell (1992) by new initialization methods for the initial set of fixed medians and a new definition of the regret value. Lorena and Senne (2003) presented a local search heuristic based on Lagrangian/surrogate relaxation techniques introduced by Senne and Lorena (2000) for the uncapacitated p-median problem. The best upper bounds obtained by the local search heuristic of Lorena and Senne (2003) were compared in Lorena and Senne (2004) with lower bounds devised by a column generation approach based on a set partitioning formulation of the CPMP.

These classic heuristics are based on the idea of performing many iterations, where each iteration slightly improves the solution quality. For instances that comprise up to approximately 5,000 objects (cf. Table 3.3), good-quality solutions can be devised since each iteration can be performed extremely fast. For instances that comprise much more than 5,000 objects, however, each iteration becomes expensive in terms of the required running time. Moreover, for large-scale instances with tight capacities, the classic heuristics that

are based on a greedy assignment strategy often need many time-consuming attempts to even generate a first feasible solution. Because of these limitations, individual objects are often aggregated to reduce the problem size. Lorena and Senne (2003), for example, reduced the problem size by aggregating houses (apartments) to blocks. This aggregation, however, leads to a loss of information and aggregation errors in the solutions (cf., e.g., Erkut and Bozkaya 1999).

3.3.3 Metaheuristics

In addition to classic heuristics, many metaheuristics have been proposed, such as the bionomic algorithm presented by Maniezzo et al. (1998), the problem-space search algorithm developed by Ahmadi and Osman (2004), the scatter search heuristics proposed by Scheuerer and Wendolsky (2006), the guided construction search heuristics introduced by Osman and Ahmadi (2007), and the grouping evolutionary algorithms developed by Landa-Torres et al. (2012). In addition, various approaches have been proposed that combine multiple metaheuristic concepts. Osman and Christofides (1994) combined the concepts simulated annealing and tabu search, Ahmadi and Osman (2005) merged a greedy random adaptive search procedure and adaptive memory programming, Díaz and Fernandez (2006) proposed an approach that combines scatter search and path relinking, and Chaves et al. (2007) linked the concepts clustering search and simulated annealing.

Like the classic heuristics, these metaheuristics also require many iterations to substantially improve the solution quality, which becomes costly in terms of the required running time for large-scale instances. In addition, they either apply manual checks while generating new solutions to guarantee that the capacity constraints are satisfied, or they apply repair operators to fix newly generated infeasible solutions. Both tasks are time consuming as well for large-scale instances.

3.3.4 Matheuristics

Recently, matheuristics have received increasing attention. Matheuristics, in general, are a powerful tool because they combine heuristic approaches with the continuously improved performance of mathematical programming solvers (cf., e.g., Carrizosa et al. 2018; Gnägi and Strub 2020). Fleszar and Hindi (2008) presented a variable neighborhood search matheuristic. Neighbors are found by randomly switching some selected medians of the current best solution to objects that are currently not selected as medians. To quickly assess the quality of the neighbors, approximation methods are used, such as assigning the objects to their nearest selected median without considering the capacity constraints.

For the most promising neighbors, feasible solutions to the CPMP are devised by solving a general assignment problem formulated as a binary linear program. Stefanello et al. (2015) proposed their iterated reduction matheuristic algorithm (IRMA) that comprises three phases. First, a simplified version of the greedy construction heuristic of Mulvey and Beck (1984) is applied. In contrast to the approach of Mulvey and Beck (1984), the order in which the objects are assigned to the fixed medians is drawn randomly and is not determined based on a regret value. Second, a mathematical programming solver is used to solve a binary linear programming formulation of the CPMP until an optimal solution is found or a time limit is reached. Third, if the optimality has not been proven in the second phase, a local search heuristic is applied that iteratively solves a binary linear programming formulation of the CPMP for subsets of clusters only. In the second and third phases, two heuristics (referred to as reduction heuristics) are applied to eliminate variables that are unlikely to be nonzero in an optimal solution. Finally, Jánošíková et al. (2017) presented two combinations of a genetic algorithm with binary linear programming. Binary linear programming is either used to generate elite individuals during the solution process of the genetic algorithm or as a postprocessing technique to improve the best solution returned by the genetic algorithm.

These matheuristics overcome some of the abovementioned limits of classic heuristics and metaheuristics by applying binary linear programming to efficiently handle the capacity constraints. However, they are either combined with a greedy assignment heuristic and/or need to compute and store large distance matrices at some point, which is challenging in terms of the required running time and prohibitive in terms of the required storage space.

3.4 Proposed matheuristic

In this section, we present the global optimization phase (cf. Subsection 3.4.1) and the local optimization phase (cf. Subsection 3.4.2) of our proposed matheuristic in detail. Moreover, we illustrate the proposed matheuristic by means of the illustrative example provided in Subsection 3.2.2. For the total running time of the proposed matheuristic, we prescribe a maximum time limit denoted as τ^{total}.

3.4.1 Global optimization phase

In the global optimization phase, we aim to devise a good-quality feasible solution by performing only a few iterations of the procedure described below; this phase builds on the approach of Baumann (2019) that was proposed for the CCCP. First, we identify and

fix a set of promising medians, denoted as J^f with $|J^f| = p$, by applying the k-means++ algorithm proposed by Arthur and Vassilvitskii (2007). Second, we assign the remaining objects to the fixed medians by using the binary linear program (M-G) provided below. By fixing the medians, we avoid computing all $\frac{n(n-1)}{2}$ pairwise distances such that only np distances between objects and fixed medians must be computed. To further reduce the number of required distance computations, we exploit the idea that objects are rarely assigned to medians that are far away and thus only allow objects to be assigned to their g-nearest medians. The g-nearest medians of each object and the corresponding distances can be determined efficiently without computing all pairwise distances using kd-trees when the number of features is much smaller than the number of objects (cf. Bentley 1975). Consequently, only ng distances between objects and fixed medians must be computed. We denote the set that comprises the g-nearest medians of object $i \in I \setminus J^f$ as J_i^f with $J_i^f \subseteq J^f$. Accordingly, we denote the set consisting of all objects that are not selected as fixed medians and that have the fixed median $j \in J^f$ among their g-nearest medians as I_j with $I_j \subseteq I \setminus J^f$. The binary linear program that we use to assign the objects to the fixed medians, referred to as (M-G), reads as follows:

$$
\text{(M-G)} \begin{cases} \text{Min.} & \displaystyle\sum_{i \in I \setminus J^f} \sum_{j \in J_i^f} d_{ij} x_{ij} & & (3.6) \\ \text{s.t.} & \displaystyle\sum_{j \in J_i^f} x_{ij} = 1 & (i \in I \setminus J^f) & (3.7) \\ & \displaystyle\sum_{i \in I_j} q_i x_{ij} \leq Q - q_j & (j \in J^f) & (3.8) \\ & x_{ij} \in \{0,1\} & (i \in I \setminus J^f;\ j \in J_i^f) & (3.9) \end{cases}
$$

The objective function given in (3.6) captures the total distance between the fixed medians and their assigned objects. Constraints (3.7) ensure that each object is assigned to exactly one fixed median. Constraints (3.8) impose the capacity limit for each of the fixed medians; the capacity limit for each fixed median $j \in J^f$ is $Q - q_j$ because it is assigned to itself a priori and thus must accommodate its own weight. Finally, the domains of the decision variables are defined in (3.9). The binary linear program (M-G) represents a special case of the generalized assignment problem in which the weight of an object is independent of the cluster to which the object is assigned. We continue by alternating between a median-update step and an object-assignment step with the goal of improving the solution quality of the initial solution. In the object-assignment step,

we again use the binary linear program (M-G) as described above to assign the objects to the currently fixed medians. In the median-update step, we update the currently fixed medians based on the new assignments obtained in the previous object-assignment step. We determine for each cluster the object that minimizes the total distance to all other objects assigned to this cluster. These objects are then used as the new fixed medians in the next object-assignment step. We perform these two steps iteratively until the current solution can no longer be improved.

Algorithm 3.1 describes the global optimization phase in detail. We start by initializing the parameter g, which defines the number of nearest medians to which an object can be assigned. Moreover, we initialize the set of fixed medians J^f by applying the k-means++ algorithm. Then, to set up the binary linear program (M-G), we determine the sets J_i^f for the objects $i \in I \setminus J^f$ and I_j for the fixed medians $j \in J^f$ based on the current value of g, and we calculate the distances d_{ij} between the objects $i \in I \setminus J^f$ and the medians $j \in J_i^f$. Thereafter, we attempt to solve the binary linear program (M-G) using a mathematical programming solver. We stop the solver as soon as the MIP gap reaches a value of 1% or lower. This setup exploits our observation that optimal or near-optimal solutions are often found quickly, while a rather long additional running time is spent on proving the optimality of these solutions. If it is found that no feasible solution exists, we double the value of g, set up the binary linear program (M-G) based on the increased value of g, and try to solve the binary linear program (M-G) again. This process is repeated until a feasible solution, denoted as S, has been found. Then, we update each median of the solution S by determining the objects assigned to the median, calculating the pairwise distances between these assigned objects, and determining the object that minimizes the total distances to all other assigned objects. For instances with $\frac{n}{p} > 10{,}000$, we propose applying an approximate median-update step to further reduce the number of distance computations. In this case, we update each median by selecting the object that is nearest to the center of gravity of the assigned objects. After all medians have been updated, we evaluate whether a new best solution, denoted as S^*, has been found. If this is the case, we reset the value of g to $g^{initial}$, we update the set of fixed medians J^f to comprise the updated medians in the new best solution S^*, and we start the next iteration if the time limit τ^{total} has not been reached. Otherwise, we stop the algorithm and return the best solution found.

Note that we determine the initial medians using the k-means++ algorithm, which is a randomized procedure. We generate multiple different solutions by running the global optimization phase multiple times, each time with a different random seed. We then return the best solution found over all runs. We denote the number of runs of the global

Algorithm 3.1 Global optimization phase

procedure GLOBALOPTIMIZATION($g^{initial}$, τ^{total})
  $g \leftarrow g^{initial}$;
  $J^f \leftarrow$ set of initial medians with $|J^f| = p$ using k-means++;
  while time limit τ^{total} has not been reached **do**
    Determine sets J_i^f for objects $i \in I \setminus J^f$ and sets I_j for medians $j \in J^f$;
    Calculate distances d_{ij} between objects $i \in I$ and medians $j \in J_i^f$;
    Solve (M-G) until MIP Gap $\leq 1\%$;
    if no feasible solution exists **then**
      $g \leftarrow g \times 2$;
    else
      $S \leftarrow$ new feasible solution found;
      for medians $j \in J^f$ **do**
        $A_j \leftarrow$ set of objects assigned to median j in solution S;
        Calculate distances $d_{ii'}$ between objects $i, i' \in A_j$;
        $j' \leftarrow$ new median $j' \in \text{argmin}_{i' \in A_j} \sum_{i \in A_j} d_{ii'}$;
        Update median j to j' in solution S;
      end for
      if solution S is new best solution **then**
        $g \leftarrow g^{initial}$;
        $S^* \leftarrow$ S;
        $J^f \leftarrow$ set of medians in solution S^*;
      else
        break;
      end if
    end if
  end while
  return best solution S^*;
end procedure

optimization phase as ν^{start}.

3.4.2 Local optimization phase

In the local optimization phase, we iteratively apply the following procedure to further improve the best solution obtained from the global optimization phase. First, we select a subset of w clusters from the set of clusters in the current best solution. Second, we identify the set of objects that belong to the selected clusters. We denote this set of objects as I^s with $I^s \subseteq I$. Third, we solve the binary linear program (M-L) given below for this subset of objects only. To speed up the solution process of the binary linear program, we consider as potential new medians only the medians of the selected subset of clusters and their l-nearest objects. This procedure is similar to the neighborhood median size-reduction heuristic proposed by Stefanello et al. (2015). The l-nearest objects of each median and the corresponding distances can again be determined efficiently using kd-trees. We denote the set of potential new medians as J^s with $J^s \subseteq I^s$. Starting with a

small subset of clusters, we enlarge the size of the subset after several iterations without improvement. Furthermore, we introduce the set L that tracks the clusters (represented by their medians) of the current best solution for which no further local improvement has been found. The binary linear program that we use during the local optimization phase, referred to as (M-L), reads as follows:

$$
\text{(M-L)} \begin{cases}
\text{Min.} & \displaystyle\sum_{i \in I^s} \sum_{j \in J^s} d_{ij} x_{ij} & & (3.10) \\
\text{s.t.} & \displaystyle\sum_{j \in J^s} x_{jj} = w & & (3.11) \\
& \displaystyle\sum_{j \in J^s} x_{ij} = 1 & (i \in I^s) & (3.12) \\
& \displaystyle\sum_{i \in I^s} q_i x_{ij} \leq Q x_{jj} & (j \in J^s) & (3.13) \\
& x_{ij} \leq x_{jj} & (i \in I^s;\ j \in J^s) & (3.14) \\
& x_{ij} \in \{0, 1\} & (i \in I^s;\ j \in J^s) & (3.15)
\end{cases}
$$

The objective function given in (3.10) captures the total distance between the medians and their assigned objects in the selected subset of clusters. Constraint (3.11) ensures that exactly w objects are selected as medians. Constraints (3.12) ensure that each object is assigned to a median, and constraints (3.13) impose the capacity limits. Constraints (3.14) are valid inequalities that substantially speed up the solution process since they tighten the linear relaxation of the binary linear program (cf., e.g., Ceselli and Righini 2005; Deng and Bard 2011; Kramer et al. 2019). Finally, the domains of the decision variables are defined in (3.15).

Algorithm 3.2 describes the local optimization phase in detail. We start by initializing the best solution found so far, denoted as S^*, with the initial solution $S^{initial}$, which represents the best solution obtained at the end of the global optimization phase. We additionally initialize the parameter w based on the input parameter n^{target}, which represents a target number of objects in the initial subset of objects I^s. A suitable value for n^{target} should be chosen such that a mathematical programming solver can solve the resulting binary linear program (M-L) within a reasonable running time. Moreover, we introduce the empty set L, which is used throughout the local optimization phase to track the medians for which no further local improvement has been found. To set up the binary linear program (M-L), we perform the following steps. First, we set up the set of potential new medians J^s by gradually including the median of the current best solution S^* that

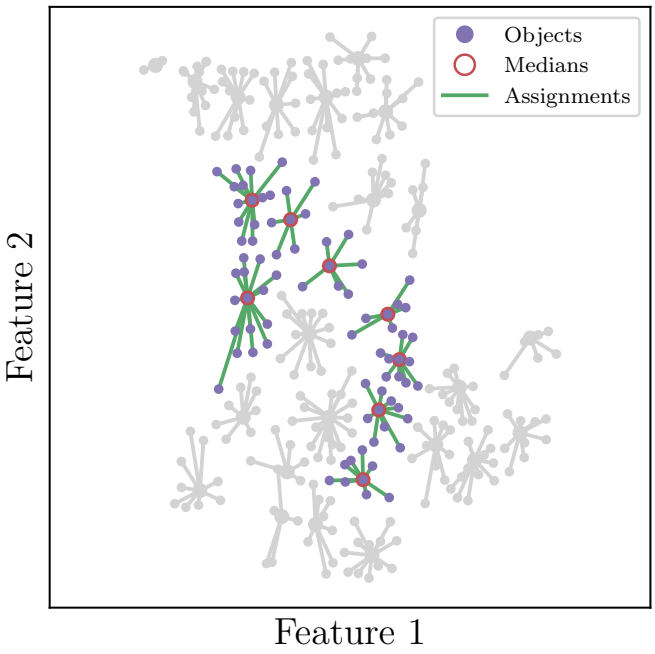
is nearest to the previously added median until $|J^s| = w$. We start with the median j' that has the largest amount of unused capacity in the current best solution S^* and that has not been marked as a median that has been optimized in previous iterations without finding any local improvement. At this point, we copy set J^s and denote the copy as J^w. We then further enlarge set J^s by including the l-nearest objects of each median $j \in J^w$. Second, we set up the set I^s by including all objects assigned to the medians $j \in J^w$ in the current best solution S^*. Third, we calculate the distances d_{ij} between the objects $i \in I^s$ and the medians $j \in J^s$. Thereafter, we try to improve the current best solution using the binary linear program (M-L) and a mathematical programming solver. To avoid very long running times for single iterations, we stop the solver after a time limit of τ^{local} has been reached, or as soon as the MIP gap reaches a value of 1% or lower. Furthermore, we set up a warm start based on the current best solution S^* to speed up the solution process. If the current best solution improved, we update set L by excluding all medians $j \in J^w$ such that these medians can be selected again in subsequent iterations; otherwise, we include all medians $j \in J^w$ in set L. This process of setting up the binary linear program (M-L), trying to solve the binary linear program (M-L), and updating set L is performed iteratively until the time limit τ^{total} is reached, or all medians selected in the current best solution are also in set L. In the latter case, we double the value of w and empty set L. As soon as the binary linear program (M-L) has been optimized with the number of medians to be selected w being equal to the total number of clusters p, we stop the algorithm and return the best solution found.

A key methodological novelty of the local improvement phase that distinguishes the proposed approach from other local search matheuristics is the cluster selection procedure. The procedure starts with an initial cluster and then iteratively adds the cluster that is closest to the previously added cluster. This strategy ensures that the resulting group of clusters forms a connected region in the feature space but does not impose any restriction on the shape of that region. As shown in Figure 3.2, it is thus possible to have clusters that are far away from each other (measured as the distance between the medians) in the same subproblem, even though the subproblem is still of manageable size. To ensure that large improvements are achieved early, the procedure selects the cluster that has the largest amount of unused capacity as the initial cluster. Such subproblems are not generated, for example, with the local search matheuristic of Stefanello et al. (2015), which does not prioritize subproblems with great potential for improving the objective function value and restricts the shape of the selected clusters to be ball-shaped.

Algorithm 3.2 Local optimization phase

procedure LocalOptimization($S^{initial}$, n^{target}, l, τ^{local}, τ^{total})
  $S^* \leftarrow S^{initial}$;
  $w \leftarrow \max\{\lceil \frac{n^{target} \times p}{n} \rceil,\ 2\}$;
  $L \leftarrow \{\}$;
  while time limit τ^{total} has not been reached **do**
    $J^* \leftarrow$ set of medians in solution S^*;
    $j' \leftarrow$ median $j' \in J^* \setminus L$ with largest amount of unused capacity in solution S^*;
    $J^s \leftarrow \{j'\}$;
    while $|J^s| < w$ **do**
      $j'' \leftarrow j'$;
      $j' \leftarrow$ nearest median $j' \in J^* \setminus J^s$ to median j'';
      $J^s \leftarrow J^s \cup \{j'\}$;
    end while
    $J^w \leftarrow$ copy set J^s;
    for medians $j \in J^w$ **do**
      $J^s \leftarrow J^s \cup$ set of l-nearest objects $i \in I^s$ of median j;
    end for
    $I^s \leftarrow$ set of objects assigned to medians $j \in J^w$ in solution S^*;
    Calculate distances d_{ij} between objects $i \in I^s$ and medians $j \in J^s$;
    Solve (M-L) with warm start based on solution S^* until MIP Gap $\leq 1\%$ or time limit τ^{local} is reached;
    $S \leftarrow$ update solution S^* according to the solution found to (M-L);
    if $w = p$ **then**
      $S^* \leftarrow$ new solution S;
      break;
    end if
    if solution S is new best solution **then**
      $S^* \leftarrow$ new best solution S;
      $J^* \leftarrow$ set of medians in solution S;
      $L \leftarrow L \setminus J^w$;
    else
      $L \leftarrow L \cup J^w$;
      if $J^* = L$ **then**
        $w \leftarrow \min\{w \times 2,\ p\}$;
        $L \leftarrow \{\}$;
      end if
    end if
  end while
  return best solution S^*;
end procedure

Figure 3.2: Illustration of the cluster selection procedure



3.4.3 Illustrative example

Figure 3.3 illustrates the solution process of the proposed matheuristic for the illustrative example from Subsection 3.2.2. Stores that are selected as medians are indicated with a red circle, and the assignments of the stores to the medians are indicated with green lines.

The first column of Figure 3.3 depicts the solution process of the global optimization phase. The objective function value (OFV) after each iteration is provided in the bottom-right corner of the corresponding subfigures. Starting with an initial set of medians determined by the k-means++ algorithm, three iterations (denoted as G1 to G3) are performed. In the first iteration, a first feasible solution is found. In the second iteration, the current solution is improved. A third iteration is performed, which does not improve the current solution and therefore is not depicted in Figure 3.3. Thus, the global optimization phase terminates with the solution depicted in the subfigure at the bottom of the first column of Figure 3.3.

The second column of Figure 3.3 depicts the solution process of the local optimization phase. We provide the objective function value (OFV) after each iteration in the bottom-right corner of the corresponding subfigures. Starting with the solution returned at the end of the global optimization phase, four iterations (denoted as L1 to L4) are performed.

We start with a subset consisting of $w = 2$ clusters, which includes the cluster with the largest amount of unused capacity. In the first iteration, a new best feasible solution is found by solving the binary linear program (M-L) for the two selected clusters only. In the second and third iterations, two subsets of clusters are considered for which no improvement is found. At this point, we double the number of clusters to be selected to $w = 4$ because all clusters in the current best solution have been examined once without achieving any improvements. Hence, in the fourth iteration, all $p = 4$ clusters are selected, and thus, all stores are considered. At the end of the fourth iteration, after finding again a new best feasible solution, we terminate the local optimization phase since all stores have been considered. Note that the best solution found at the end of the fourth iteration corresponds to an optimal solution. However, there is no guarantee of finding an optimal solution as long as the parameter value l is chosen such that $l < n$.

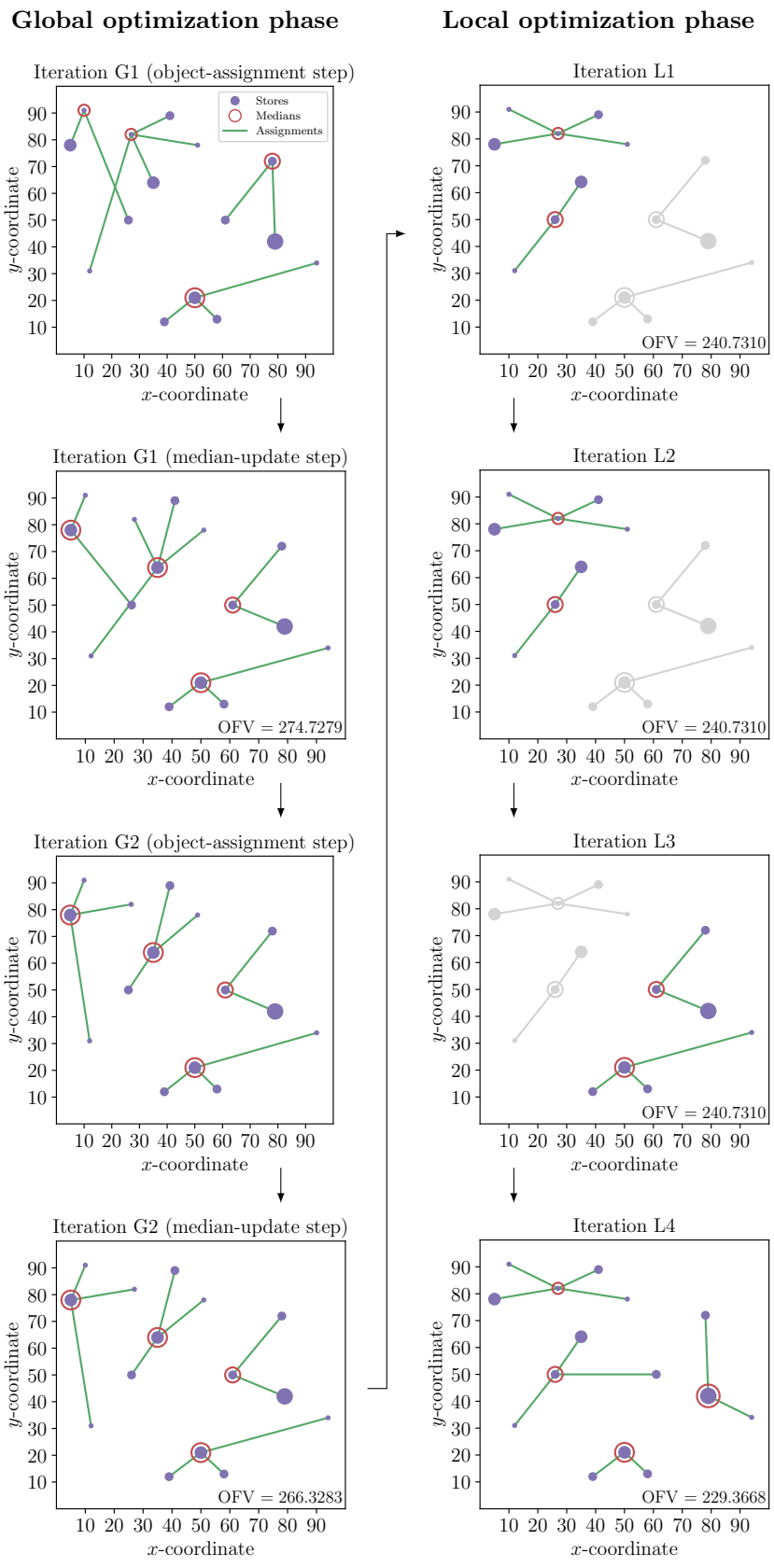
3.5 Computational experiment

In this section, we present the test set (cf. Subsection 3.5.1), the experimental design (cf. Subsection 3.5.2), and the main results of our computational experiment (cf. Subsection 3.5.3).

3.5.1 Test set

The complete test set comprises the 31 instances described in Table 3.4. The first 16 instances are well-known test instances from the literature. These include the six instances from the group SJC that were introduced by Lorena and Senne (2003) and the ten instances from the groups p3038 and fnl4461 that were generated by Lorena and Senne (2004) and Stefanello et al. (2015), respectively. The instances from the group SJC comprise real-world data of the central area of the Brazilian city São José dos Campos. The objects of these instances correspond to blocks, and the weights represent the number of houses belonging to each block. The authors did not provide any additional information regarding the interpretation of the clusters or the capacity limit. The ten instances from groups p3038 and fnl4461 are generic instances that are based on two TSP instances from the TSPLIB (cf. Reinelt 1991) with 3038 and 4461 nodes, respectively. To the best of our knowledge, the instances of group fnl4461 are the largest publicly available test instances that have been tested in the literature so far. Since large-scale instances for the uncapacitated p-median problem comprise up to 100,000 objects (cf., e.g., Hansen et al. 2009), these largest existing instances for the CPMP are considered small-scale. Therefore, we generated a set of medium- and large-scale instances based on four TSP instances from the

Figure 3.3: Solution process of the proposed matheuristic for the illustrative example



VLSI collection (cf. Rohe 2013) ranging from approximately 10,000 up to approximately 500,000 nodes. Following the procedure proposed by Stefanello et al. (2015), we took the coordinates of the nodes of the TSP instances as features for the objects of our CPMP instances. Furthermore, for each object $i \in I$, we determined a weight q_i by drawing a random integer from the set $\{1, 2, \ldots, 100\}$. Finally, for each of the four TSP instances, we generated a group of three CPMP instances that differ with respect to the number of clusters to be found. Based on the number of clusters to be found, we determined the capacity Q by using Equation (3.16), where r was randomly drawn from the range $[0.8, 0.9]$.

$$Q = \left\lceil \sum_{i \in I} \frac{q_i}{p \times r} \right\rceil \tag{3.16}$$

Note that instances seven to 28 are not based on real-world data. These generic instances were generated with the aim of testing the performance of different solution approaches when the number of objects to be clustered increases.

Additionally, we generated three high-dimensional instances with up to approximately 800 features. Since the clustering of images is an important task in data mining (cf., e.g., Ushakov and Vasilyev 2019), the objects of the first two instances represent images, and we took the grayscale level of the pixels as features for the objects. The instance cancer5000_-10_784 is based on the images from the collection of textures in colorectal cancer histology (cf. Kather et al. 2016). The instance digits60000_100_784 is based on the images from the MNIST database of handwritten digits (cf. LeCun et al. 1998). The number of features for both instances corresponds to the resolution (28x28 pixels) of the images. The third instance KDD145751_100_74 is based on the KDD protein homology dataset (cf. KDD 2004), which is a popular dataset for testing clustering methods (cf., e.g., Bachem et al. 2016). The objects of this instance correspond to protein sequences that are described by 74 different features. For each of these high-dimensional instances, we determined weights and capacities following the procedure described above. The number of clusters to be found was selected arbitrarily to be $p = 10$ for the small-scale instance cancer5000_-10_784 and $p = 100$ for the medium- and large-scale instances digits60000_100_784 and KDD145751_100_74. We generated these instances with the aim of assessing whether the proposed matheuristic can also deal with instances that comprise more than two features.

Note that the instances from the literature and the instances generated here comprise a given capacity that is the same for all objects. However, Mulvey and Beck (1984), for example, considered the more general case in which each object has an individual capacity, i.e., $Q_i \neq Q_{i'}$ with $i, i' \in I$. The proposed matheuristic is implemented such that it can

Table 3.4: Overview of instances

	ID	Name	n	p	d	Source
Small-scale	1	SJC1	100	10	2	Lorena and Senne (2003)
	2	SJC2	200	15	2	Lorena and Senne (2003)
	3	SJC3a	300	25	2	Lorena and Senne (2003)
	4	SJC3b	300	30	2	Lorena and Senne (2003)
	5	SJC4a	402	30	2	Lorena and Senne (2003)
	6	SJC4b	402	40	2	Lorena and Senne (2003)
	7	p3038_600	3,038	600	2	Lorena and Senne (2004)
	8	p3038_700	3,038	700	2	Lorena and Senne (2004)
	9	p3038_800	3,038	800	2	Lorena and Senne (2004)
	10	p3038_900	3,038	900	2	Lorena and Senne (2004)
	11	p3038_1000	3,038	1,000	2	Lorena and Senne (2004)
	12	fnl4461_0020	4,461	20	2	Stefanello et al. (2015)
	13	fnl4461_0100	4,461	100	2	Stefanello et al. (2015)
	14	fnl4461_0250	4,461	250	2	Stefanello et al. (2015)
	15	fnl4461_0500	4,461	500	2	Stefanello et al. (2015)
	16	fnl4461_1000	4,461	1,000	2	Stefanello et al. (2015)
Medium-scale	17	XMC10150_100	10,150	100	2	This paper
	18	XMC10150_500	10,150	500	2	This paper
	19	XMC10150_1000	10,150	1,000	2	This paper
	20	FNA52057_100	52,057	100	2	This paper
	21	FNA52057_500	52,057	500	2	This paper
	22	FNA52057_1000	52,057	1,000	2	This paper
Large-scale	23	SRA104814_100	104,814	100	2	This paper
	24	SRA104814_500	104,814	500	2	This paper
	25	SRA104814_1000	104,814	1,000	2	This paper
	26	LRA498378_100	498,378	100	2	This paper
	27	LRA498378_500	498,378	500	2	This paper
	28	LRA498378_1000	498,378	1,000	2	This paper
High-dimensional	29	cancer5000_10_784	5,000	10	784	This paper
	30	digits60000_100_784	60,000	100	784	This paper
	31	KDD145751_100_74	145,751	100	74	This paper

also deal with this more general case.

The generated medium- and large-scale instances, as well as the generated high-dimensional instances, can be downloaded from `https://github.com/mgnaegi/cpmp_instances`.

3.5.2 Experimental design

We compare the performance of the proposed matheuristic with the performance of the state-of-the-art approach for the CPMP presented by Stefanello et al. (2015). Since the implementation of this approach is not publicly available, we reimplemented it according

Table 3.5: Parameter values used for the proposed matheuristic

Parameter	Phase	Description	Value
ν^{start}	Global	Number of runs of global optimization phase	3
$g^{initial}$	Global	Initial number of nearest medians to which an object can be assigned	10
τ^{local}	Local	Time limit for solving formulation (M-L)	300
n^{target}	Local	Target number of objects in initial subset	200
l	Local	Number of nearest objects of each median to be considered as potential new medians	50 ($n < 5{,}000$) 10 ($n \geq 5{,}000$)
τ^{total}	Both	Time limit on total running time [in seconds]	3,600

to the description given in the paper. We also report results for an exact approach based on the binary linear program formulation presented by Lorena and Senne (2004) and a mathematical programming solver. We implemented all three approaches in Python 3.7, and we used the Gurobi Optimizer 8.1 as a mathematical programming solver with the default settings if not stated otherwise. Our computations were performed on an HP workstation with an Intel(R) Xeon(R) Silver 4114 CPU (2.20 GHz) with ten cores and 128 GB of RAM. For the proposed matheuristic, we used the parameter values that are listed in Table 3.5. For the matheuristic of Stefanello et al. (2015), we used the default parameter values proposed in their paper. For each tested instance, we imposed a running time limit of 3,600 seconds. Note that our results differ slightly from the results reported by Stefanello et al. (2015) for their approach since we performed our computations on different hardware and used a different mathematical programming solver. Furthermore, in contrast to Stefanello et al. (2015), we ran each approach for each tested instance only once to limit the required computation time for the entire experiment.

3.5.3 Numerical results

First, we compared the three approaches in terms of solution quality and their suitability for different problem sizes. These results are summarized in Table 3.6 for the small-scale instances and in Table 3.7 for the medium-scale, the large-scale, and the high-dimensional instances. We refer to the proposed matheuristic as GB20^{MH}, to the matheuristic proposed by Stefanello et al. (2015) as $\text{SAM15}^{\text{MH}}$and to the exact approach based on the binary linear program formulation presented by Lorena and Senne (2004) as LS04^{BLP}. Bold values indicate the best results among all tested approaches. In the first five columns, we list the characteristics of the instances. In the sixth column (OFV^{BEST}), we report the objective function value of the best feasible solution found among all tested approaches

within the prescribed time limit. In the next column (OFV^{LB}), we report the best lower bound on the objective function value obtained with the exact approach based on the binary linear program formulation. Finally, in the last six columns, we report for each tested approach the relative gap between the objective function value of the best feasible solution found and the reported value for OFV^{BEST}, as well as the total required running time.

In contrast to the two tested approaches from the literature, the proposed matheuristic found feasible solutions for all instances within the prescribed time limit. For the small-scale instances, the new matheuristic matched the performance of the other approaches, and for the medium- and large-scale instances, the new matheuristic consistently delivered the best feasible solutions. Additionally, for the high-dimensional instances, the proposed matheuristic performed best among all tested approaches.

Table 3.6: Best feasible solutions for CPMP instances from the literature

							LS04^{BLP}		SAM15^{MH}		GB20^{MH}	
	ID	Name	n	p	d	OFV^{BEST}	OFV^{LB}	Gap[%]	CPU[s]	Gap[%]	CPU[s]	Gap[%]	CPU[s]
Small-scale	1	SJC1	100	10	2	17,288.99	17,288.99	**0.00**	22.85	0.15	6.80	**0.00**	4.29
	2	SJC2	200	15	2	33,270.94	33,269.51	**0.00**	74.26	0.44	9.62	**0.00**	5.81
	3	SJC3a	300	25	2	45,335.16	45,335.16	**0.00**	174.18	0.55	17.43	0.04	45.48
	4	SJC3b	300	30	2	40,635.90	40,635.90	**0.00**	46.34	**0.00**	13.45	0.46	29.19
	5	SJC4a	402	30	2	61,925.51	61,925.51	**0.00**	482.06	1.07	31.27	0.41	88.31
	6	SJC4b	402	40	2	52,458.02	52,458.02	**0.00**	154.00	0.12	21.14	0.64	80.57
	7	p3038_600	3,038	600	2	122,748.81	121,596.38	0.70	limit	**0.00**	limit	0.13	limit
	8	p3038_700	3,038	700	2	109,706.83	108,679.98	0.90	limit	**0.00**	limit	0.15	limit
	9	p3038_800	3,038	800	2	100,094.76	99,089.87	1.64	limit	**0.00**	limit	0.37	limit
	10	p3038_900	3,038	900	2	92,346.43	91,258.33	0.22	limit	**0.00**	limit	0.20	limit
	11	p3038_1000	3,038	1,000	2	85,895.48	85,102.19	0.10	limit	**0.00**	1,016.34	0.09	limit
	12	fnl4461_0020	4,461	20	2	1,288,143.94	220,323.41	44.93	limit	0.08	1,191.68	**0.00**	limit
	13	fnl4461_0100	4,461	100	2	553,818.42	212,458.56	46.13	limit	1.42	2,693.60	**0.00**	limit
	14	fnl4461_0250	4,461	250	2	340,734.40	197,570.01	52.59	limit	**0.00**	limit	0.14	limit
	15	fnl4461_0500	4,461	500	2	225,285.82	172,895.18	53.15	limit	**0.00**	limit	0.73	limit
	16	fnl4461_1000	4,461	1,000	2	145,898.85	138,569.83	38.47	limit	**0.00**	limit	0.45	limit
						Average		14.93		0.24		0.24	

(limit) Time limit of 3,600 seconds reached

Table 3.7: Best feasible solutions for CPMP instances introduced in this paper

	ID	Name	n	p	d	OFV^{BEST}	$LS04^{BLP}$			$SAM15^{MH}$		$GB20^{MH}$	
							OFV^{LB}	Gap[%]	CPU[s]	Gap[%]	CPU[s]	Gap[%]	CPU[s]
Medium-scale	17	XMC10150_100	10,150	100	2	181,803.84	–	–	limit	4.71	limit	**0.00**	limit
	18	XMC10150_500	10,150	500	2	72,206.88	–	–	limit	13.10	limit	**0.00**	limit
	19	XMC10150_1000	10,150	1,000	2	46,805.19	–	–	limit	13.02	limit	**0.00**	limit
	20	FNA52057_100	52,057	100	2	2,106,162.53	–	–	limit	2.05	limit	**0.00**	limit
	21	FNA52057_500	52,057	500	2	925,436.91	–	–	limit	3.88	limit	**0.00**	limit
	22	FNA52057_1000	52,057	1,000	2	632,238.94	–	–	limit	7.71	limit	**0.00**	limit
Large-scale	23	SRA104814_100	104,814	100	2	4,791,303.49	–	–	limit	5.18	limit	**0.00**	limit
	24	SRA104814_500	104,814	500	2	2,123,756.22	–	–	limit	8.70	limit	**0.00**	limit
	25	SRA104814_1000	104,814	1,000	2	1,491,563.04	–	–	limit	6.67	limit	**0.00**	limit
	26	LRA498378_100	498,378	100	2	104,208,012.52	–	–	limit	–	limit	**0.00**	limit
	27	LRA498378_500	498,378	500	2	44,384,702.04	–	–	limit	–	limit	**0.00**	limit
	28	LRA498378_1000	498,378	1,000	2	30,836,227.02	–	–	limit	–	limit	**0.00**	limit
High-dimensional	29	cancer5000_10_784	5,000	10	784	4,915,501.60	4,745,521.58	5.88	limit	0.60	limit	**0.00**	1,117.09
	30	digits60000_100_784	60,000	100	784	94,867,884.32	–	–	limit	2.46	limit	**0.00**	limit
	31	KDD145751_100_74	145,751	100	74	149,729,640.61	–	–	limit	13.72	limit	**0.00**	limit
						Average		5.88		6.82		0.00	

(–) No feasible solution/lower bound found within 3,600 seconds; (limit) Time limit of 3,600 seconds reached

Next, we compared the two heuristic approaches GB20$^{\mathrm{MH}}$ and SAM15$^{\mathrm{MH}}$ with respect to the ability to find good first feasible solutions quickly. These results are summarized in Table 3.8 for the small-scale instances and in Table 3.9 for the medium-scale, the large-scale, and the high-dimensional instances. The first five columns provide information analogously to Tables 3.6 and 3.7. In the sixth column (OFV$^{\mathrm{BEST}}$), we again report the objective function value of the best feasible solution found among all tested approaches within the prescribed time limit. In the following four columns, we report the relative gap between the objective function value of the first feasible solution found and the reported value for OFV$^{\mathrm{BEST}}$, as well as the required running time to find the first feasible solution. In the last column, we report the speed-up factor between the proposed matheuristic and the benchmark approach regarding the required running time to find the first feasible solution.

With only one exception, the proposed matheuristic devised first feasible solutions with lower objective function values than the benchmark approach. Furthermore, the proposed matheuristic required substantially less running time to devise the first feasible solution for most of the tested instances. For some instances, the proposed matheuristic found the first feasible solution approximately 31 times faster than the benchmark approach. The ability of the proposed matheuristic to provide good first feasible solutions quickly might be valuable, for example, for clustering problems where p is not known in advance. If this is the case, the proposed matheuristic can be run multiple times, each time with another value of p, since each run can be performed quickly.

Table 3.8: First feasible solutions for CPMP instances from the literature

							SAM15^{MH}		GB20^{MH}		
	ID	Name	n	p	d	OFV^{BEST}	Gap[%]	CPU[s]	Gap[%]	CPU[s]	Speed-up
Small-scale	1	SJC1	100	10	2	17,288.99	137.63	0.92	**12.55**	**0.19**	4.89
	2	SJC2	200	15	2	33,270.94	62.44	0.91	**16.84**	**0.16**	5.72
	3	SJC3a	300	25	2	45,335.16	43.30	0.80	**16.05**	**0.19**	4.22
	4	SJC3b	300	30	2	40,635.90	46.76	0.85	**15.67**	**0.19**	4.55
	5	SJC4a	402	30	2	61,925.51	52.18	0.89	**25.53**	**0.22**	4.04
	6	SJC4b	402	40	2	52,458.02	35.06	0.83	**16.35**	**0.19**	4.32
	7	p3038_600	3,038	600	2	122,748.81	74.37	**1.09**	**16.66**	4.26	0.26
	8	p3038_700	3,038	700	2	109,706.83	78.90	**1.09**	**19.08**	6.89	0.16
	9	p3038_800	3,038	800	2	100,094.76	84.89	**1.19**	**24.88**	37.46	0.03
	10	p3038_900	3,038	900	2	92,346.43	85.35	**1.20**	**29.81**	26.77	0.04
	11	p3038_1000	3,038	1,000	2	85,895.48	89.33	**1.10**	**32.08**	10.44	0.11
	12	fnl4461_0020	4,461	20	2	1,288,143.94	61.88	1.35	**13.12**	**1.07**	1.26
	13	fnl4461_0100	4,461	100	2	553,818.42	58.38	1.35	**12.09**	**1.30**	1.04
	14	fnl4461_0250	4,461	250	2	340,734.40	65.10	**1.34**	**10.10**	2.55	0.53
	15	fnl4461_0500	4,461	500	2	225,285.82	71.97	**1.35**	**13.83**	2.39	0.57
	16	fnl4461_1000	4,461	1,000	2	145,898.85	80.22	**1.41**	**16.68**	7.91	0.18
						Average	70.48	1.10	18.21	6.39	

Table 3.9: First feasible solutions for CPMP instances introduced in this paper

	ID	Name	n	p	d	OFV^{BEST}	$SAM15^{MH}$ Gap[%]	$SAM15^{MH}$ CPU[s]	$GB20^{MH}$ Gap[%]	$GB20^{MH}$ CPU[s]	Speed-up
Medium-scale	17	XMC10150_100	10,150	100	2	181,803.84	112.36	3.22	**8.78**	**3.18**	1.01
	18	XMC10150_500	10,150	500	2	72,206.88	97.55	**3.36**	**16.35**	29.38	0.11
	19	XMC10150_1000	10,150	1,000	2	46,805.19	95.55	**3.54**	**17.24**	23.61	0.15
	20	FNA52057_100	52,057	100	2	2,106,162.53	99.62	77.55	**10.27**	**15.26**	5.08
	21	FNA52057_500	52,057	500	2	925,436.91	79.12	78.17	**8.51**	**28.35**	2.76
	22	FNA52057_1000	52,057	1,000	2	632,238.94	89.76	80.00	**8.91**	**45.28**	1.77
Large-scale	23	SRA104814_100	104,814	100	2	4,791,303.49	74.60	337.16	**10.43**	**31.90**	10.57
	24	SRA104814_500	104,814	500	2	2,123,756.22	85.49	338.87	**8.65**	**59.84**	5.66
	25	SRA104814_1000	104,814	1,000	2	1,491,563.04	83.52	342.37	**8.25**	**92.97**	3.68
	26	LRA498378_100	498,378	100	2	104,208,012.52	–	–	**26.89**	**237.19**	–
	27	LRA498378_500	498,378	500	2	44,384,702.04	–	–	**32.49**	**388.04**	–
	28	LRA498378_1000	498,378	1,000	2	30,836,227.02	–	–	**47.79**	**876.64**	–
High-dimensional	29	cancer5000_10_784	5,000	10	784	4,915,501.60	34.94	17.78	**19.90**	**4.08**	4.35
	30	digits60000_100_784	60,000	100	784	94,867,884.32	17.54	2,182.93	**3.94**	**69.92**	31.22
	31	KDD145751_100_74	145,751	100	74	149,729,640.61	**40.08**	1,678.86	65.30	**798.92**	2.10
						Average	75.84	428.65	19.58	180.30	

(–) No feasible solution found within 3,600 seconds

Finally, we highlight the importance of the two phases of the proposed matheuristic. Figure 3.4 depicts the improvement of the best feasible solutions in terms of the objective function value over time. Each subfigure reports the results for one of the following three exemplary instances: fnl4461_1000 (small-scale), FNA52057_1000 (medium-scale) and LRA498378_1000 (large-scale). For small-scale instances, the majority of the running time was spent in the local optimization phase, during which substantial improvements to the solution quality can be achieved. For medium-scale instances, both phases consume an equal amount of running time. While the best feasible solution can be improved quickly at the beginning, no further improvements can be attained after spending some time in the global optimization phase. At this point, the local optimization phase starts, during which substantial improvements can be achieved by locally reoptimizing the best feasible solution obtained at the end of the global optimization phase. For large-scale instances, the entire running time is spent in the global optimization phase. The first feasible solution is devised fairly quickly, particularly when considering the large number of objects to be clustered. Then, the best feasible solution is consecutively improved until the prescribed maximum running time has elapsed.

3.6 Capacitated centered clustering problem

In this section, we show that the proposed matheuristic can also be applied to other variants of capacitated clustering problems, such as the capacitated centered clustering problem (CCCP).

The CCCP differs from the CPMP as follows (cf. Negreiros and Palhano 2006). The cluster centers must correspond to the geometric center of the objects assigned to the clusters and are not selected among the objects themselves. Like the CPMP, the CCCP is NP-hard as well (cf., e.g., Chaves et al. 2018). Compared to the extensive literature dealing with the CPMP, only a few papers have considered the CCCP. The problem was first discussed by Negreiros and Palhano (2006), who proposed an approach comprising two phases that are based on a construction heuristic and variable neighborhood search heuristic. Most recently, Chaves et al. (2018) proposed an adaptive biased random-key genetic algorithm with a clustering search, and Baumann (2019) presented a matheuristic based on the well-known k-means algorithm. These two approaches devised numerous new best-known solutions for standard test instances from the literature.

The matheuristic proposed in this paper can be applied to the CCCP since a given feasible solution to an instance of the CPMP can be converted into a feasible solution to an instance of the CCCP that comprises the same objects and that prescribes the same

Figure 3.4: Improvement of the best feasible solution over time

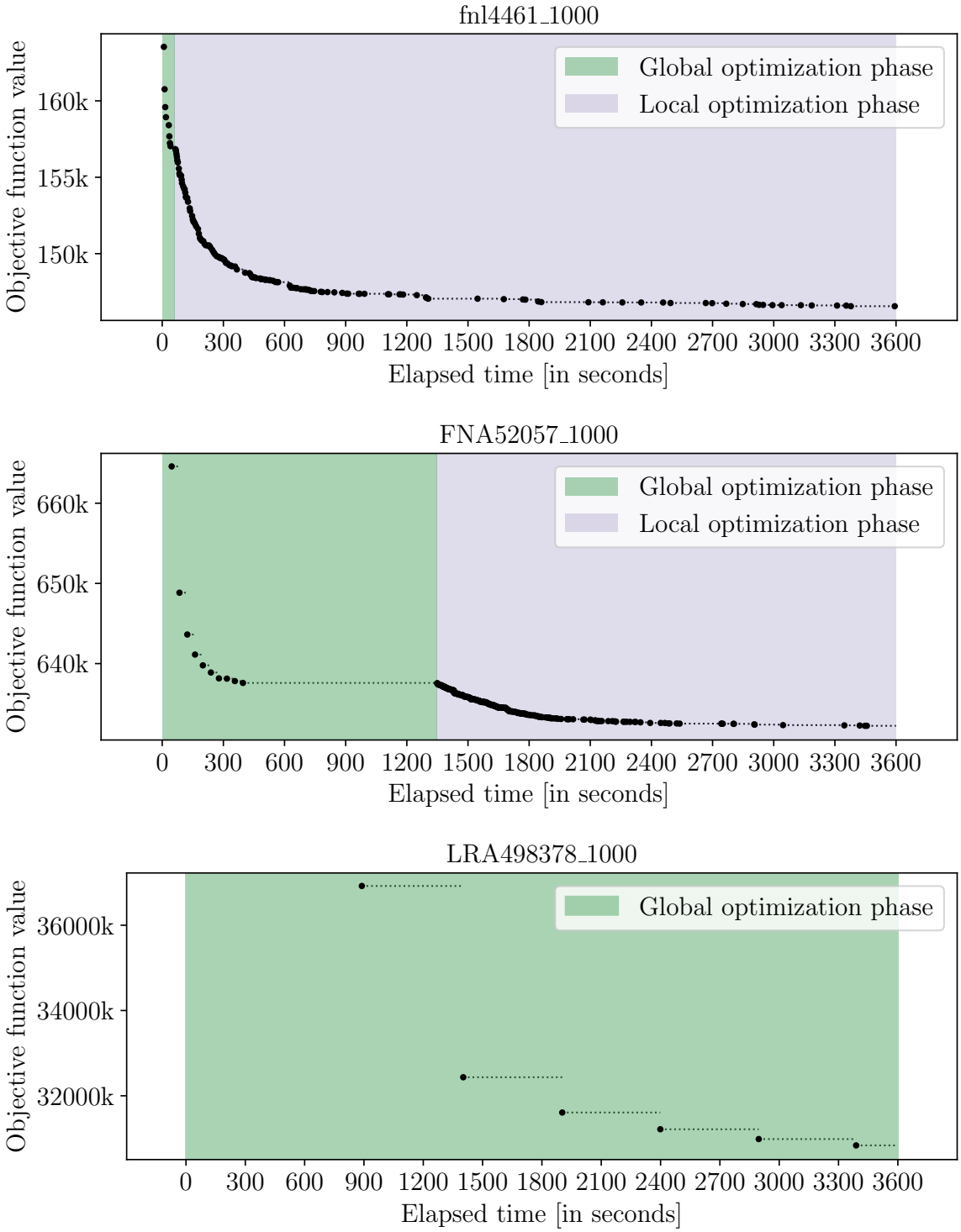


Table 3.10: Best feasible solutions for CCCP instances

					GB20^{MH}		
ID	Name	n	p	OFV^{BKS}	OFV	Gap[%]	CPU[s]
1	SJC1	100	10	17,359.75	17,363.47	0.02	4.29
2	SJC2	200	15	33,181.65	33,425.61	0.74	5.81
3	SJC3a	300	25	45,354.29	45,446.88	0.20	45.48
4	SJC3b	300	30	40,660.55	40,907.16	0.61	29.19
5	SJC4a	402	30	61,931.60	62,030.00	0.16	88.31
6	SJC4b	402	40	52,202.48	52,877.27	1.29	80.57
7	p3038_600	3,038	600	126,172.76	123,692.06	−1.97	limit
8	p3038_700	3,038	700	113,462.26	111,253.80	−1.95	limit
9	p3038_800	3,038	800	105,352.33	102,256.93	−2.94	limit
10	p3038_900	3,038	900	97,319.54	94,342.45	−3.06	limit
11	p3038_1000	3,038	1,000	89,896.55	87,407.06	−2.77	limit
12	fnl4461_0020	4,461	20	1,282,694.80	1,287,391.44	0.37	limit
13	fnl4461_0100	4,461	100	560,148.77	553,236.84	−1.23	limit
14	fnl4461_0250	4,461	250	348,101.42	340,864.91	−2.08	limit
15	fnl4461_0500	4,461	500	237,491.70	227,094.35	−4.38	limit
16	fnl4461_1000	4,461	1,000	159,672.99	149,062.05	−6.65	limit
					Average	−1.48	

(limit) Time limit of 3,600 seconds reached

capacity limit for all clusters. For each cluster of the feasible solution to be converted, the selected medians must be replaced by the geometric centers of the assigned objects. The objective function value is then calculated as the total distance between the newly computed cluster centers and their assigned objects.

The first 16 instances used in our computational experiment for the CPMP are also often analyzed in literature dealing with the CCCP. For these instances, we converted the best feasible solutions obtained by using the proposed matheuristic into feasible solutions of the CCCP by applying the procedure described above. In Table 3.10, in the first four columns, we list the characteristics of these instances. In the next column (OFV^{BKS}), we list the objective function values of the best-known solutions reported by Chaves et al. (2018) and Baumann (2019). Finally, in the last three columns, we report the objective function value of the best feasible solution found for the CCCP by the proposed matheuristic (OFV), the relative gap between the reported value for OFV and the reported value for OFV^{BKS}, as well as the total required running time. For the six smaller instances, the proposed matheuristic provides solutions with small gaps to the best-known solutions reported in the literature. For nine of the ten larger instances, we were able to find new best-known solutions even though the proposed matheuristic was not specifically designed for the CCCP.

3.7 Conclusions

In this paper, we considered the capacitated p-median problem (CPMP). For this problem, we proposed a matheuristic that is specifically designed for instances with a large number of objects. In a computational experiment, the proposed matheuristic consistently outperformed the state-of-the-art approach for the CPMP on medium- and large-scale instances while matching the performance for small-scale instances. Furthermore, we showed that the proposed matheuristic can also be applied to related capacitated clustering problems such as the capacitated centered clustering problem (CCCP). For the largest problem instances of the CCCP tested in this paper, the proposed matheuristic was able to find new best-known solutions.

We suggest the following directions for future research. The proposed matheuristic can be adapted to problems with additional side constraints, such as a lower bound on the capacities of the clusters. Since the proposed approach is based on binary linear programming, such additional side constraints can easily be incorporated. Moreover, the proposed problem decomposition strategies can be applied to related problems such as the capacitated p-center problem (CPCP), which differs from the CPMP only with respect to the objective function (cf., Kramer et al. 2019). Instead of minimizing the total distance between the objects and their assigned medians, the maximum distance over all assignments of objects to medians is minimized.

Bibliography

Ahmadi, S., Osman, I.H., 2004. Density based problem space search for the capacitated clustering p-median problem. Annals of Operations Research 131, 21–43.

Ahmadi, S., Osman, I.H., 2005. Greedy random adaptive memory programming search for the capacitated clustering problem. European Journal of Operational Research 162, 30–44.

Arthur, D., Vassilvitskii, S., 2007. k-means++: The advantages of careful seeding, in: Gabow, H. (Ed.), Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, Pennsylvania USA. pp. 1027–1035.

Avella, P., Boccia, M., Salerno, S., Vasilyev, I., 2012. An aggregation heuristic for large scale p-median problem. Computers & Operations Research 39, 1625–1632.

Bachem, O., Lucic, M., Hassani, S.H., Krause, A., 2016. Approximate k-means++ in sublinear time, in: Schuurmans, D., Wellman, M. (Eds.), Proceedings of the thirtieth AAAI Conference on Artificial Intelligence, Phoenix, Arizona USA. pp. 1459–1467.

Baldacci, R., Hadjiconstantinou, E., Maniezzo, V., Mingozzi, A., 2002. A new method for solving capacitated location problems based on a set partitioning approach. Computers & Operations Research 29, 365–386.

Baumann, P., 2019. A binary linear programming-based K-means approach for the capacitated centered clustering problem, in: Wang, M., Li, J., Tsung, F., Yeung, A. (Eds.), 2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Macau. pp. 382–365.

Bentley, J.L., 1975. Multidimensional binary search trees used for associative searching. Communications of the ACM 18, 509–517.

Boccia, M., Sforza, A., Sterle, C., Vasilyev, I., 2008. A cut and branch approach for the capacitated p-median problem based on Fenchel cutting planes. Journal of Mathematical Modelling and Algorithms 7, 43–58.

Carrizosa, E., Guerrero, V., Morales, D.R., 2018. On mathematical optimization for the visualization of frequencies and adjacencies as rectangular maps. European Journal of Operational Research 265, 290–302.

Ceselli, A., Righini, G., 2005. A branch-and-price algorithm for the capacitated p-median problem. Networks: An International Journal 45, 125–142.

Chaves, A.A., de Assis Correa, F., Lorena, L.A.N., 2007. Clustering search heuristic for the capacitated p-median problem, in: Corchado, E., Corchado, J., Abraham, A. (Eds.), Innovations in Hybrid Intelligent Systems. Springer, pp. 136–143.

Chaves, A.A., Gonçalves, J.F., Lorena, L.A.N., 2018. Adaptive biased random-key genetic algorithm with local search for the capacitated centered clustering problem. Computers & Industrial Engineering 124, 331–346.

Deng, Y., Bard, J.F., 2011. A reactive GRASP with path relinking for capacitated clustering. Journal of Heuristics 17, 119–152.

Díaz, J.A., Fernandez, E., 2006. Hybrid scatter search and path relinking for the capacitated p-median problem. European Journal of Operational Research 169, 570–585.

El-Alfy, E.S.M., 2007. Applications of genetic algorithms to optimal multilevel design of MPLS-based networks. Computer Communications 30, 2010–2020.

Erkut, E., Bozkaya, B., 1999. Analysis of aggregation errors for the p-median problem. Computers & Operations Research 26, 1075–1096.

Fleszar, K., Hindi, K.S., 2008. An effective VNS for the capacitated p-median problem. European Journal of Operational Research 191, 612–622.

Gnägi, M., Strub, O., 2020. Tracking and outperforming large stock-market indices. Omega 90, 101999.

Hansen, P., Brimberg, J., Urošević, D., Mladenović, N., 2009. Solving large p-median clustering problems by primal–dual variable neighborhood search. Data Mining and Knowledge Discovery 19, 351–375.

Jánošíková, L., Herda, M., Haviar, M., 2017. Hybrid genetic algorithms with selective crossover for the capacitated p-median problem. Central European Journal of Operations Research 25, 651–664.

Kather, J.N., Weis, C.A., Bianconi, F., Melchers, S.M., Schad, L.R., Gaiser, T., Marx, A., Zöllner, F.G., 2016. Multi-class texture analysis in colorectal cancer histology. Scientific Reports 6, 27988.

KDD, 2004. KDD cup 2004, protein homology dataset. Website. `https://www.kdd.org/kdd-cup/view/kdd-cup-2004/Data`. Accessed: 2020-02-27.

Koskosidis, Y.A., Powell, W.B., 1992. Clustering algorithms for consolidation of customer orders into vehicle shipments. Transportation Research Part B: Methodological 26, 365–379.

Kramer, R., Iori, M., Vidal, T., 2019. Mathematical models and search algorithms for the capacitated p-center problem. INFORMS Journal on Computing. To appear.

Landa-Torres, I., Del Ser, J., Salcedo-Sanz, S., Gil-Lopez, S., Portilla-Figueras, J.A., Alonso-Garrido, O., 2012. A comparative study of two hybrid grouping evolutionary techniques for the capacitated p-median problem. Computers & Operations Research 39, 2214–2222.

LeCun, Y., Cortes, C., Burges, C.J.C., 1998. The MNIST database of handwritten digits. Website. `http://yann.lecun.com/exdb/mnist/index.html`. Accessed: 2019-12-20.

Lorena, L.A., Senne, E.L., 2004. A column generation approach to capacitated p-median problems. Computers & Operations Research 31, 863–876.

Lorena, L.A.N., Senne, E.L.F., 2003. Local search heuristics for capacitated p-median problems. Networks and Spatial Economics 3, 407–419.

Maniezzo, V., Mingozzi, A., Baldacci, R., 1998. A bionomic approach to the capacitated p-median problem. Journal of Heuristics 4, 263–280.

Medaglia, A.L., Villegas, J.G., Rodríguez-Coca, D.M., 2009. Hybrid biobjective evolutionary algorithms for the design of a hospital waste management network. Journal of Heuristics 15, 153.

Mulvey, J.M., Beck, M.P., 1984. Solving capacitated clustering problems. European Journal of Operational Research 18, 339–348.

Negreiros, M., Palhano, A., 2006. The capacitated centred clustering problem. Computers & Operations Research 33, 1639–1663.

Osman, I.H., Ahmadi, S., 2007. Guided construction search metaheuristics for the capacitated p-median problem with single source constraint. Journal of the Operational Research Society 58, 100–114.

Osman, I.H., Christofides, N., 1994. Capacitated clustering problems by hybrid simulated annealing and tabu search. International Transactions in Operational Research 1, 317–336.

Pirkul, H., 1987. Efficient algorithms for the capacitated concentrator location problem. Computers & Operations Research 14, 197–208.

Reinelt, G., 1991. TSPLIB. Website. `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html`. Accessed: 2019-12-20.

Rohe, A., 2013. VLSI collection. Website. `http://www.math.uwaterloo.ca/tsp/vlsi/index.html`. Accessed: 2019-12-20.

Scheuerer, S., Wendolsky, R., 2006. A scatter search heuristic for the capacitated clustering problem. European Journal of Operational Research 169, 533–547.

Senne, E.L., Lorena, L.A., 2000. Lagrangean/surrogate heuristics for p-median problems, in: Laguna, M., Gonzalez-Velarde, J. (Eds.), Computing tools for modeling, optimization and simulation. Springer, pp. 115–130.

Stefanello, F., de Araújo, O.C., Müller, F.M., 2015. Matheuristics for the capacitated p-median problem. International Transactions in Operational Research 22, 149–167.

Ushakov, A.V., Vasilyev, I., 2019. A computational comparison of parallel and distributed k-median clustering algorithms on large-scale image data, in: Bykadorov, I., Strusevich, V., Tchemisova, T. (Eds.), International Conference on Mathematical Optimization Theory and Operations Research, Ekaterinburg, Russia. pp. 119–130.

Yaghini, M., Karimi, M., Rahbar, M., 2013. A hybrid metaheuristic approach for the capacitated p-median problem. Applied Soft Computing 13, 3922–3930.