Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

A matheuristic for large-scale capacitated clustering

Mario Gnägi [*], Philipp Baumann

Department of Business Administration, University of Bern, Schützenmattstrasse 14, 3012 Bern, Switzerland

ARTICLE INFO

Keywords:
Large-scale capacitated clustering
Capacitated p-median problem
Capacitated centered clustering problem
Binary linear programming
Matheuristic

ABSTRACT

Clustering addresses the problem of assigning similar objects to groups. 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. Numerous 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. We propose a matheuristic with new problem decomposition strategies that can deal with instances comprising up to 500,000 objects. In a computational experiment, the proposed matheuristic consistently outperformed the state-of-the-art approach on medium- and large-scale instances while having similar 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.

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 (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 (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 (e.g., Lorena and Senne 2004). Given a set of n weighted objects that are described by m 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 (Osman and Ahmadi, 2007), and the problem of finding a feasible solution to an instance of the CPMP is NP-complete (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 (e.g., Hansen et al. 2009). Since the CPMP is an extension of the uncapacitated p-median problem, we are interested in developing an approach for large-scale instances of the CPMP.

Several exact solution approaches (e.g., Ceselli and Righini 2005, Boccia et al. 2008) and numerous heuristic solution approaches (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 (Jánošíková et al., 2017). The approach of Stefanello et al. (2015) iteratively constructs an initial solution with a randomized procedure

[*] Corresponding author.
E-mail addresses: mario.gnaegi@pqm.unibe.ch (M. Gnägi), philipp.baumann@pqm.unibe.ch (P. Baumann).

https://doi.org/10.1016/j.cor.2021.105304
Received 18 May 2020; Received in revised form 22 March 2021; Accepted 22 March 2021
Available online 29 March 2021

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 feasibility with respect to the capacity constraints during the entire solution process, and (c) apply efficient data structures (k-d trees; 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 solution 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 k-d 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 k-d trees in the local optimization phase to considerably 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 (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 given the available computational resources. 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. Note that the implementation of the proposed matheuristic and the generated instances are publicly available.

Table 1
Notation used for the binary linear program (M-CPMP).

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

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) (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 2, we describe the CPMP in more detail. In Section 3, we review the related literature. In Section 4, we describe the proposed matheuristic. In Section 5, we report the computational results. In Section 6, we apply the proposed matheuristic to the CCCP, and in Section 7, we provide some conclusions and give an outlook on future research.

2. Capacitated p-median problem

In this section, we describe the CPMP in more detail (Section 2.1) and provide a small illustrative example (Section 2.2).

2.1. Description of the problem

The CPMP can be stated as follows (e.g., Lorena and Senne 2004). Given is a set of n objects denoted as $I = \{1,\dots,n\}$. Each object $i \in I$ corresponds to an m-dimensional feature vector $v_i \in \mathbb{R}^m$ and is associated with a weight q_i. Based on these feature vectors, a distance $d(v_i, v_j)$ can be computed for each pair of objects $i, j \in I$ (e.g., Euclidean distance). Note that the distances do not constitute an input to the problem as they must be calculated based on the feature vectors. The goal is to partition the objects into a prescribed number of clusters by selecting p objects as cluster centers (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 (Lorena and Senne, 2004); the notation used is summarized in Table 1. Note that an object $j \in I$ is selected as a median if it is assigned to itself, i.e., $x_{jj} = 1$.

Table 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

$$\text{(M-CPMP)}\left\{\begin{array}{llll}\text{Min.} & \sum_{i\in I}\sum_{j\in I} d(v_i, v_j)x_{ij} & & \text{(a)}\\ \text{s.t.} & \sum_{j\in I} x_{jj} = p & & \text{(b)}\\ & \sum_{j\in I} x_{ij} = 1 & (i\in I) & \text{(c)}\\ & \sum_{i\in I} q_i x_{ij} \le Q x_{jj} & (j\in I) & \text{(d)}\\ & x_{ij}\in\{0,1\} & (i,j\in I) & \text{(e)}\end{array}\right. \quad (1)$$

The objective function given in (1)(a) captures the total distance between the medians and their assigned objects. Constraint (1)(b) ensures that exactly p objects are selected as medians. Constraints (1)(c) assure that each object is assigned to exactly one selected median. Constraints (1)(d) impose the capacity limit for each object that is selected as a median. Finally, the domains of the decision variables are defined in (1)(e).

The CPMP has various real-world applications that have been discussed in the literature. Many of these real-world applications arise in facility location (e.g., Medaglia et al. 2009, Jánošíková et al. 2017). Other exemplary applications are the consolidation of customer orders into truckload shipments (Koskosidis and Powell, 1992) and the structuring of multiprotocol-label switching networks (El-Alfy, 2007). For a broad overview of real-world applications of the CPMP, we refer to Ahmadi and Osman (2005).

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 Section 4.4.

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 vectors) and the number of employees (weights) of the stores are given in Table 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 Fig. 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.

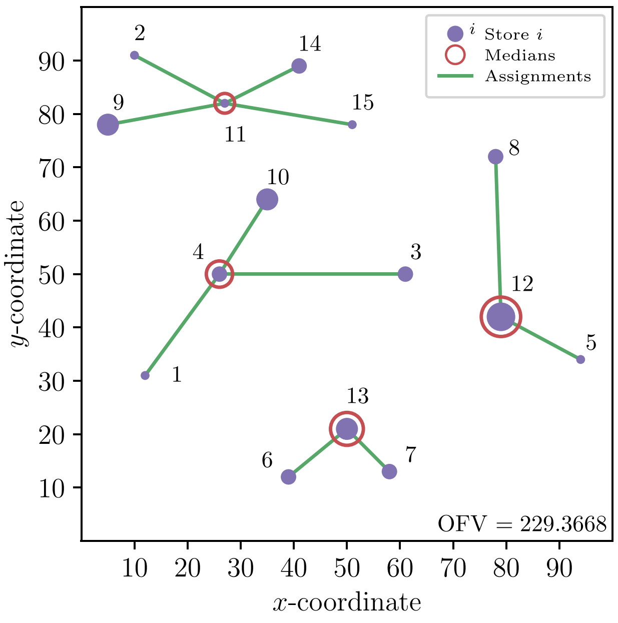

Fig. 1. Optimal solution of the illustrative example.

3. Literature review

Capacitated clustering has received a lot of attention recently. In this section, we focus only on solution approaches for the CPMP. Other closely related capacitated clustering problems differ from the CPMP with respect to the objective function (e.g., Brimberg et al. 2019, Zhou et al. 2019, Puerto et al. 2020), the constraints (e.g., Espejo et al. 2021), or both (e.g., Ríos-Mercado et al. 2021).

We categorize and discuss the papers dealing with the CPMP according to the types of proposed solution approaches. In Sections 3.1 to 3.4, we review exact approaches, classic heuristics, metaheuristics, and matheuristics. Table 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. Note that given the considerable improvement in available software and hardware, the approaches might be applicable to larger instances than those listed in the last column of Table 3.

3.1. Exact approaches

Almost all papers listed in Table 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 (Table 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.

Table 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
Mai et al. (2018)		✓			156
Proposed approach				✓	498,378

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. The order in which the objects are assigned is determined based on a regret value that is computed for each object. The regret value of an object is defined as the difference between the distance to its second nearest fixed median and the distance to its first 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. Finally, Mai et al. (2018) proposed a construction and an improvement heuristic for the CPMP. The construction heuristic uses a Gaussian mixture modeling approach that incorporates the capacity constraints. The improvement heuristic shifts or swaps objects between different clusters.

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 (Table 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 (e.g., Erkut and Bozkaya 1999).

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.

3.4. 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 (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.

4. Proposed matheuristic

In this section, we present the global optimization phase (Section 4.1) and the local optimization phase (Section 4.2) of our proposed matheuristic in detail. Moreover, we briefly describe the data structure k-d trees (Section 4.3), which we use in both phases of the proposed matheuristic. Finally, we illustrate the proposed matheuristic by means of the illustrative example provided in Section 2.2. For the total running time of the proposed matheuristic, we prescribe a maximum time limit denoted as τ^{total}.

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 $|J^f| = p$. 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 can be determined efficiently using k-d trees without computing all pairwise distances (Section 4.3). 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:

$$\text{(M-G)}\begin{cases}\text{Min.} & \sum_{i\in I\setminus J^f}\sum_{j\in J_i^f} d(v_i,v_j)x_{ij} & & \text{(a)}\\ \text{s.t.} & \sum_{j\in J_i^f} x_{ij}=1 & (i\in I\setminus J^f) & \text{(b)}\\ & \sum_{i\in I_j} q_i x_{ij} \le Q-q_j & (j\in J^f) & \text{(c)}\\ & x_{ij}\in\{0,1\} & (i\in I\setminus J^f;\ j\in J_i^f) & \text{(d)}\end{cases} \quad (2)$$

The objective function given in (2)(a) captures the total distance between the fixed medians and their assigned objects. Constraints (2)(b) ensure that each object is assigned to exactly one fixed median. Constraints (2)(c) 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 (2)(d). 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 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. For this initialization, we propose two alternative methods which we describe further below. 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 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 (lines 12–17) 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 (lines 18–24). 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.

Algorithm 1 Global optimization phase

```
1: procedure GlobalOptimization($g^{initial}$, $\tau^{total}$, $init$)
2:     $g \leftarrow g^{initial}$;
3:     $J^f \leftarrow$ set of initial medians with $|J^f| = p$ using method $init$;
4:     while time limit $\tau^{total}$ has not been reached do
5:         Determine sets $J_i^f$ for objects $i \in I \setminus J^f$ and sets $I_j$ for medians $j \in J^f$;
6:         Calculate distances $d(v_i, v_j)$ between objects $i \in I \setminus J^f$ and medians $j \in J_i^f$;
7:         Solve (M-G) until MIP Gap $\leq 1\%$;
8:         if no feasible solution exists then
9:             $g \leftarrow g \times 2$;
10:        else
11:            $S \leftarrow$ new feasible solution found;
12:            for medians $j \in J^f$ do
13:                $A_j \leftarrow$ set of objects assigned to median $j$ in solution $S$;
14:                Calculate distances $d(v_i, v_{i'})$ between objects $i, i' \in A_j$;
15:                $j' \leftarrow$ new median $j' \in \text{argmin}_{i' \in A_j} \sum_{i \in A_j} d_{ii'}$;
16:                Update median $j$ to $j'$ in solution $S$;
17:            end for
18:            if solution $S$ is new best solution then
19:                $g \leftarrow g^{initial}$;
20:                $S^* \leftarrow$ S;
21:                $J^f \leftarrow$ set of medians in solution $S^*$;
22:            else
23:                break;
24:            end if
25:        end if
26:    end while
27:    return best solution $S^*$;
28: end procedure
```

To determine the initial medians (line 3 of Algorithm 1), we consider two alternative methods:

- The k-means++ algorithm proposed by Arthur and Vassilvitskii (2007). This method aims at spreading out the initial medians as far as possible. Thereby, objects are iteratively selected to be medians with a probability that is proportional to the squared distance between an object and its closest already selected median. Note that this method does not consider any capacity constraints.
- A capacity-based initialization method that corresponds to an accelerated version of the procedure GlobalOptimization($g^{initial}$, τ^{total}, $init$) with $init = k$-means++. To accelerate this procedure, we introduce the following two modifications. First, instead of assigning the objects to the fixed medians by using the binary linear program (M-G) (line 7 of Algorithm 1), each object is assigned greedily to its nearest fixed median that has a sufficient amount of unused capacity. The order in which the objects are assigned is determined based on the regret function proposed by Mulvey and Beck (1984), i.e., a regret value r_i is calculated for each object i with $r_i = d(v_i, v_{j''}) - d(v_i, v_{j'})$ where j' and j'' are the first and second nearest medians of object i. The objects are assigned in decreasing order of their regret values. Second, instead of returning the best feasible solution, only the set of medians in the best feasible solution found is returned.

The k-means++ algorithm quickly returns a set of initial medians but neglects the capacity restriction. The novel capacity-based method requires a longer running time but takes into account the capacity restrictions and thus provides more promising initial medians. A set of promising initial medians allows to choose a smaller value for $g^{initial}$. This is beneficial especially for large-scale instances since it considerably reduces the size of the binary linear program (M-G).

Note that in both initialization methods, we use 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 optimization phase as v^{start}.

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. The cluster-selection procedure starts with selecting the median that has the largest amount of unused capacity. Then, the $w - 1$ medians that are nearest to the already selected median are selected. These nearest medians can be determined efficiently using k-d trees (Section 4.3). Second, we identify the set of objects that are assigned to the selected medians. 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 can again be determined efficiently using k-d trees (Section 4.3). 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 track 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:

$$
\text{(M-L)}\begin{cases}
\text{Min.} & \sum_{i\in I^s}\sum_{j\in J^s} d(v_i,v_j)x_{ij} & & \text{(a)}\\
\text{s.t.} & \sum_{j\in J^s} x_{jj} = w & & \text{(b)}\\
& \sum_{j\in J^s} x_{ij} = 1 & (i \in I^s) & \text{(c)}\\
& \sum_{i\in I^s} q_i x_{ij} \leq Q x_{jj} & (j \in J^s) & \text{(d)}\\
& x_{ij} \leq x_{jj} & (i \in I^s;\ j \in J^s) & \text{(e)}\\
& x_{ij} \in \{0,1\} & (i \in I^s;\ j \in J^s) & \text{(f)}
\end{cases} \quad (3)
$$

The objective function given in (3)(a) captures the total distance between the medians and their assigned objects in the selected subset of clusters. Constraint (3)(b) ensures that exactly w objects are selected as medians. Constraints (3)(c) ensure that each object is assigned to a median, and constraints (3)(d) impose the capacity limits. Constraints (3)(e) are valid inequalities that substantially speed up the solution process since they tighten the linear relaxation of the binary linear program (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)(f).

Algorithm 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 initialize an 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 (lines 6–15). First, we determine the set of potential new medians J^s and the set of objects that belong to the selected clusters I^s. 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. Then, we include the $w - 1$ medians that are nearest to median j' in the set J^s. At this point, we copy set J^s and denote the copy as J^w. Then, we determine the set I^s by including all objects assigned to the medians $j \in J^w$ in the current best solution S^*. We then further enlarge set J^s by including the l-nearest objects $i \in I^s$ of each median $j \in J^w$. Second, we calculate the distances 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 provide a warm start based on the current best solution S^* and the set of selected clusters. This typically speeds up the solution process of the solver since a first feasible solution is already provided. If the current best solution has been improved, we update set L by excluding all medians $j \in J^w$ (line 25) such that these medians can be selected again in subsequent iterations; otherwise, we include all medians $j \in J^w$ in set L (line 27). 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 (lines 29 and 30 of Algorithm 2). 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.

Algorithm 2 Local optimization phase

```
1: procedure LocalOptimization($S^{initial}$, $n^{target}$, $l$, $\tau^{local}$, $\tau^{total}$)
2:     $S^* \leftarrow S^{initial}$;
3:     $w \leftarrow \max\{\lceil \frac{n^{target} \times p}{n} \rceil, 2\}$;
4:     $L \leftarrow \{\}$;
5:     while time limit $\tau^{total}$ has not been reached do
6:         $J^* \leftarrow$ set of medians in solution $S^*$;
7:         $j' \leftarrow$ median $j' \in J^* \setminus L$ with largest amount of unused capacity in solution $S^*$;
8:         $J^s \leftarrow \{j'\}$;
9:         $J^s \leftarrow J^s \cup$ set of $(w-1)$-nearest medians $j \in J^*$ of median $j'$;
10:        $J^w \leftarrow$ copy set $J^s$;
11:        $I^s \leftarrow$ set of objects assigned to medians $j \in J^w$ in solution $S^*$;
12:        for medians $j \in J^w$ do
13:            $J^s \leftarrow J^s \cup$ set of $l$-nearest objects $i \in I^s$ of median $j$;
14:        end for
15:        Calculate distances $d(v_i, v_j)$ between objects $i \in I^s$ and medians $j \in J^s$;
16:        Solve (M-L) with warm start based on solution $S^*$ until MIP Gap $\leq 1\%$ or time limit $\tau^{local}$ is reached;
17:        $S \leftarrow$ update solution $S^*$ according to the solution found to (M-L);
18:        if $w = p$ then
19:            $S^* \leftarrow$ new solution $S$;
20:            break;
21:        end if
22:        if solution $S$ is better than solution $S^*$ then
23:            $S^* \leftarrow$ new best solution $S$;
24:            $J^* \leftarrow$ set of medians in solution $S$;
25:            $L \leftarrow L \setminus J^w$;
26:        else
27:            $L \leftarrow L \cup J^w$;
28:            if $J^* = L$ then
29:                $w \leftarrow \min\{w \times 2, p\}$;
30:                $L \leftarrow \{\}$;
31:            end if
32:        end if
33:    end while
34:    return best solution $S^*$;
35: end procedure
```

A 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 the cluster that has the largest amount of unused capacity as the initial cluster. In contrast, the local search matheuristic of Stefanello et al. (2015), for example, does not prioritize subproblems with great potential for improving the objective function value. In Appendix A, we provide results which indicate that this cluster-selection procedure outperforms other cluster-selection procedures.

4.3. Search for nearest neighbors using k-d trees

Both phases of the proposed matheuristic require to search for nearest neighbors. This task can be performed efficiently using k-d trees without calculating all pairwise distances (Bentley, 1975). A k-d tree is a binary tree, in which each node represents an object. Moreover, each node is associated with a feature by which the feature space is divided into two halves. Once such a k-d tree is constructed, the tree structure can be exploited to eliminate large portions of the search space, such that the nearest neighbors of any object described by the same features can be determined efficiently. Note that in high-dimensional feature spaces smaller parts of the search space can be eliminated than in low-dimensional feature spaces, such that the search for nearest neighbors using k-d trees becomes more time consuming.

4.4. Illustrative example

Fig. 2 illustrates the solution process of the proposed matheuristic for the illustrative example from Section 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 Fig. 2 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 Fig. 2. Thus, the global optimization phase terminates with the solution depicted in the subfigure at the bottom of the first column of Fig. 2.

The second column of Fig. 2 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 $l < n$.

5. Computational experiment

In this section, we evaluate the performance of the proposed matheuristic in terms of solution quality and running time. In Section 5.1, we introduce the test set. In Section 5.2, we choose the control parameters of our matheuristic. In Section 5.3, we explain the design of our computational experiment. In Section 5.4, we report and discuss the computational results.

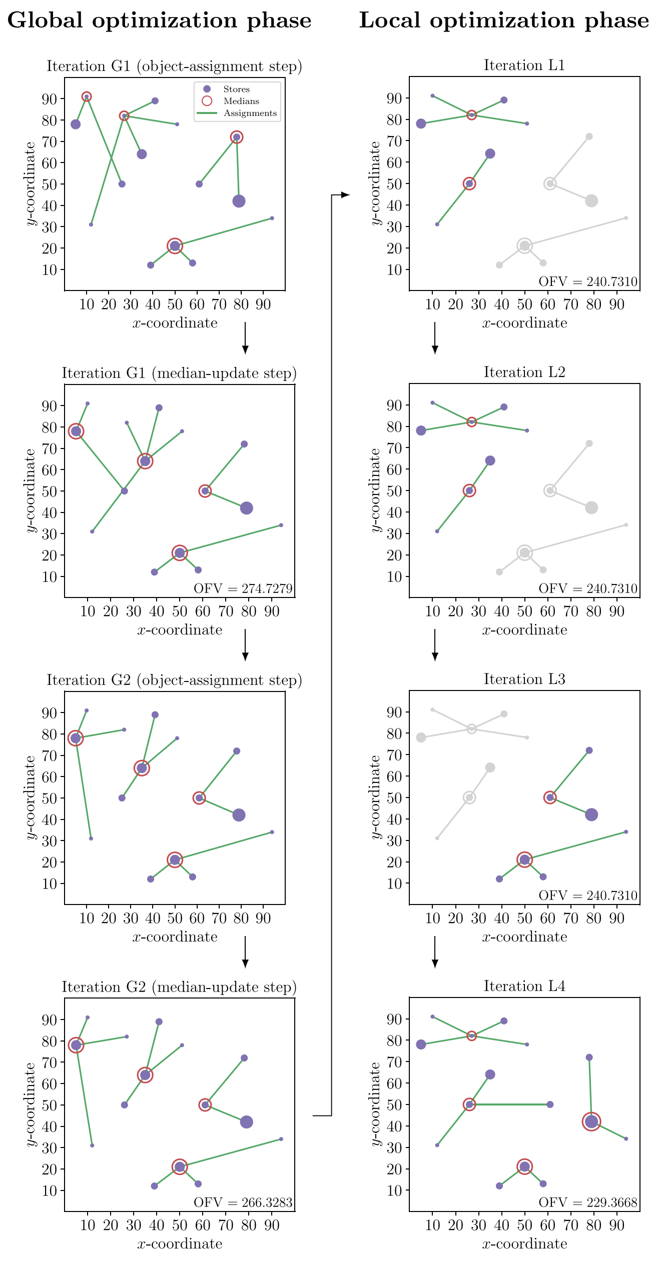

Fig. 2. Solution process of the proposed matheuristic for the illustrative example.

5.1. *Test set*

The complete test set comprises the 31 instances described in Table 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 (Reinelt, 1991) with 3,038 and 4,461 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 (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 VLSI collection (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 with $p = \{100, 1{,}000, 2{,}000\}$ being the number of clusters to be found. Based on these numbers of clusters, we determined the capacity Q by using Eq. (4), where r was randomly drawn from the range $[0.8, 0.9]$.

$$Q = \left\lceil \sum_{i \in I} \frac{q_i}{p \times r} \right\rceil \tag{4}$$

Note that instances 7 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 (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 (Kather et al., 2016). The instance digits60000_100_784 is based on the images from the MNIST database of handwritten digits (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 (KDD, 2004), which is a popular dataset for testing clustering methods (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 all consider a given capacity that is the same for all objects. However, Mulvey and Beck (1984), for example, considered the more general case in which any two objects i and i' may have different capacities, i.e., $Q_i \neq Q_{i'}$. The proposed matheuristic is implemented such that it can 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/phil85/GB21-MH.

5.2. *Selection of control parameters*

The proposed matheuristic has several parameters that can control the trade-off between solution quality and running time. The goal of this subsection is to provide guidelines on how to select values for these control parameters.

We recommend to perform three runs of the global optimization phase due to the random initialization ($v^{start} = 3$). The size of the subproblems in the local optimization phase should be determined based on the performance of the mathematical programming solver. Currently, state-of-the-art solvers can generally cope quite well with problems that include up to 200 objects. We therefore chose $n^{target} =$

Table 4
Overview of instances.

	ID	Name	n	p	$\frac{n}{p}$	m	Source
Small-scale	1	SJC1	100	10	10.0	2	Lorena and Senne (2003)
	2	SJC2	200	15	13.3	2	Lorena and Senne (2003)
	3	SJC3a	300	25	12.0	2	Lorena and Senne (2003)
	4	SJC3b	300	30	10.0	2	Lorena and Senne (2003)
	5	SJC4a	402	30	13.4	2	Lorena and Senne (2003)
	6	SJC4b	402	40	10.1	2	Lorena and Senne (2003)
	7	p3038_600	3,038	600	5.1	2	Lorena and Senne (2004)
	8	p3038_700	3,038	700	4.3	2	Lorena and Senne (2004)
	9	p3038_800	3,038	800	3.8	2	Lorena and Senne (2004)
	10	p3038_900	3,038	900	3.4	2	Lorena and Senne (2004)
	11	p3038_1000	3,038	1,000	3.0	2	Lorena and Senne (2004)
	12	fnl4461_0020	4,461	20	223.1	2	Stefanello et al. (2015)
	13	fnl4461_0100	4,461	100	44.6	2	Stefanello et al. (2015)
	14	fnl4461_0250	4,461	250	17.8	2	Stefanello et al. (2015)
	15	fnl4461_0500	4,461	500	8.9	2	Stefanello et al. (2015)
	16	fnl4461_1000	4,461	1,000	4.5	2	Stefanello et al. (2015)
Medium-scale	17	XMC10150_100	10,150	100	101.5	2	This paper
	18	XMC10150_1000	10,150	1,000	10.2	2	This paper
	19	XMC10150_2000	10,150	2,000	5.1	2	This paper
	20	FNA52057_100	52,057	100	520.6	2	This paper
	21	FNA52057_1000	52,057	1,000	52.1	2	This paper
	22	FNA52057_2000	52,057	2,000	26.0	2	This paper
Large-scale	23	SRA104814_100	104,814	100	1,048.1	2	This paper
	24	SRA104814_1000	104,814	1,000	104.8	2	This paper
	25	SRA104814_2000	104,814	2,000	52.4	2	This paper
	26	LRA498378_100	498,378	100	4,983.8	2	This paper
	27	LRA498378_1000	498,378	1,000	498.4	2	This paper
	28	LRA498378_2000	498,378	2,000	249.2	2	This paper
High-dimensional	29	cancer5000_10_784	5,000	10	500.0	784	This paper
	30	digits60000_100_784	60,000	100	600.0	784	This paper
	31	KDD145751_100_74	145,751	100	1,457.5	74	This paper

200. To avoid investing an inordinate amount of time on proving the optimality of the solution found for a subproblem in the local optimization phase, we recommend to impose a time limit $\tau^{local} = 300$.

To devise promising values for the remaining control parameters, we conducted the following parameter tuning: we applied the matheuristic to a subset of nine test instances with different values for the control parameters $g^{initial}$, l, and $init$. The selected instances comprise three small-, three medium- and three large-scale instances. For the three control parameters, we tested the values $g^{initial} = \{5, 10, 20\}$, $l = \{10, 50, 100\}$ and $init = \{k\text{-means++, capacity-based}\}$. For each instance and each combination of parameter values ($3 \times 3 \times 2 = 18$ combinations), we ran the proposed matheuristic with a total running time limit of $\tau^{total} = 3{,}600$ seconds for each run.

The results of this parameter tuning are summarized in Table 5. We report for each combination of parameter values the average relative gap between the objective function value of the feasible solution found and the objective function value of the best feasible solution found among all tested combinations of parameter values. Bold values indicate the best results among all tested combinations of parameter values for the small-scale instances (columns three to five) and for the medium- and large-scale instances (columns six to eight). Based on these results, we determined the parameter setting presented below in Table 6 for the main computational experiment.

5.3. Experimental design

We compared 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 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 implementation is publicly available on https://github.com/phil85/GB21-MH. 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 6. For the matheuristic of Stefanello et al. (2015), we used the default parameter values proposed in their paper. We ran each approach for each tested instance three times, and for each run, we imposed a running time limit of 3,600 seconds.

5.4. 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 7 for the small-scale instances and in Table 8 for the medium-scale, the large-scale, and the high-dimensional instances. We refer to the proposed matheuristic as GB21$^{\mathrm{MH}}$, to the matheuristic proposed by Stefanello et al. (2015) as SAM15$^{\mathrm{MH}}$ and to the exact approach based on the binary linear program formulation presented by Lorena and Senne (2004) as LS04$^{\mathrm{BLP}}$. Bold values indicate the best results among all tested approaches. In the first five columns of both tables, we list the characteristics of the instances. In the sixth column of Table 7, we report the objective function value of the best known solution reported by Stefanello et al. (2015) (OFV$^{\mathrm{SAM15}}$), and in the sixth column of Table 8, we report the objective function value of the best feasible solution found among all tested approaches within the prescribed time limit (OFV$^{\mathrm{BEST}}$). In the seventh column of both tables, we report the best lower bound on the objective function value obtained with the exact approach based on the binary linear program formulation (OFV$^{\mathrm{LB}}$). Finally, in the last six columns of both tables, we report for each tested approach the relative gap between the objective function value of the best feasible solution found over the three runs and the reported value for OFV$^{\mathrm{SAM15}}$ and OFV$^{\mathrm{BEST}}$, respectively, as well as the total required running time for the run in which the best feasible solution has been found. Additionally, we provide some

Table 5
Results for varying values of parameters $g^{initial}$, l, and $init$.

Method	$init$	Small-scale			Medium- and Large-scale		
		$l = 10$	$l = 50$	$l = 100$	$l = 10$	$l = 50$	$l = 100$
k-means++	$g^{initial} = 5$	0.141	0.100	0.160	1.164	0.786	0.811
	$g^{initial} = 10$	0.154	**0.085**	0.145	6.360	0.745	0.766
	$g^{initial} = 20$	0.095	0.192	0.337	6.927	0.779	0.788
capacity-based	$g^{initial} = 5$	0.159	0.238	0.383	0.493	0.437	0.449
	$g^{initial} = 10$	0.118	0.220	0.365	**0.141**	0.142	0.161
	$g^{initial} = 20$	0.093	0.157	0.302	0.404	0.383	0.377

Table 6
Parameter values used for the proposed matheuristic.

Parameter	Phase	Description	Value
v^{start}	Global	Number of runs of global optimization phase	3
$init$	Global	Initialization method to determine the initial set of fixed medians	k-means++ ($n < 5{,}000$) capacity-based ($n \geq 5{,}000$)
$g^{initial}$	Global	Initial number of nearest medians to which an object can be assigned	10
τ^{local}	Local	Time limit for solving formulation (M-L) [sec]	300
n^{target}	Local	Target number of objects in initial subset	200
l	Local	Number of nearest objects to 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 [sec]	3,600

further statistics to assess the robustness of the approaches GB21^{MH} and SAM15^{MH} in the Tables B.1 and B.2 in Appendix B.

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.

The approach SAM15^{MH} could not find a feasible solution for the three largest instances because the computation of all pairwise distances exceeded the memory of the used workstation. We tested the approach SAM15^{MH} also with an extended running time limit. It turns out that the benchmark approach cannot find better solutions than the proposed matheuristic even if we extend the running time limit to 10 hours. For the instance SRA104814_2000, for example, the relative gap could be reduced from 9.56% to 5.84%.

Next, we compared the two heuristic approaches GB21^{MH} and SAM15^{MH} with respect to the ability to find good first feasible solutions quickly. These results are summarized in Table 9 for the small-scale instances and in Table 10 for the medium-scale, the large-scale, and the high-dimensional instances. The first six columns of both tables provide information analogously to Tables 7 and 8. In the following four columns, we report the relative gap between the objective function value of the best first feasible solution found over the three runs and the reported value for OFV^{SAM15} and OFV^{BEST}, respectively, as well as the required running time to find the best 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 best 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 135 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.

Finally, we highlight the importance of the two phases of the proposed matheuristic. Fig. 3 depicts the improvement of the best feasible solutions in terms of the objective function value over time for the proposed matheuristic and, as a reference, also for the approach SAM15^{MH}. Each subfigure reports the results for the first of the three runs for one of the following three exemplary instances: fnl4461_1000 (small-scale), FNA52057_1000 (medium-scale) and LRA498378_1000 (large-scale). Note that in the third subfigure, no results are shown for the approach SAM15^{MH} since no feasible solution has been found for this instance. 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 a similar 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.

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 (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 (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

Table 7
Best feasible solutions for CPMP instances from the literature.

	ID	Name	*n*	*p*	*m*	OFV^{SAM15}	LS04^{BLP}			SAM15^{MH}		GB21^{MH}	
							OFV^{LB}	Gap [%]	CPU [s]	Gap^{MIN} [%]	CPU [s]	Gap^{MIN} [%]	CPU [s]
Small-scale	1	SJC1	100	10	2	17,288.99	17,288.99	**0.00**	22.85	**0.00**	6.08	**0.00**	3.94
	2	SJC2	200	15	2	33,270.94	33,269.51	**0.00**	74.26	**0.00**	11.34	**0.00**	4.78
	3	SJC3a	300	25	2	45,335.16	45,335.16	**0.00**	174.18	**0.00**	28.21	**0.00**	30.57
	4	SJC3b	300	30	2	40,635.90	40,635.90	**0.00**	46.34	**0.00**	14.84	**0.00**	31.04
	5	SJC4a	402	30	2	61,925.51	61,925.51	**0.00**	482.06	**0.00**	42.71	0.15	59.52
	6	SJC4b	402	40	2	52,458.02	52,458.02	**0.00**	154.00	**0.00**	24.18	0.17	32.87
	7	p3038_600	3,038	600	2	122,711.17	121,596.38	0.73	limit	**0.03**	limit	0.04	limit
	8	p3038_700	3,038	700	2	109,677.30	108,679.98	0.93	limit	**0.03**	limit	0.04	limit
	9	p3038_800	3,038	800	2	100,064.94	99,089.87	1.67	limit	**0.03**	limit	0.03	limit
	10	p3038_900	3,038	900	2	92,310.09	91,258.33	0.26	limit	0.05	2,608.49	**0.02**	limit
	11	p3038_1000	3,038	1,000	2	85,854.05	85,102.19	0.15	limit	0.05	1,010.10	**0.04**	limit
	12	fnl4461_0020	4,461	20	2	1,283,536.73	220,323.41	45.45	limit	0.44	1,193.97	**0.28**	limit
	13	fnl4461_0100	4,461	100	2	548,909.01	212,458.56	47.44	limit	2.13	limit	**0.55**	limit
	14	fnl4461_0250	4,461	250	2	335,888.87	197,570.01	54.79	limit	1.02	limit	**0.46**	limit
	15	fnl4461_0500	4,461	500	2	224,662.49	172,895.18	53.57	limit	**0.23**	limit	0.39	limit
	16	fnl4461_1000	4,461	1,000	2	145,862.38	138,569.83	38.50	limit	**0.03**	limit	0.10	limit
						Average		15.22		0.25		0.14	

(limit) Time limit of 3,600 seconds reached

Table 8
Best feasible solutions for CPMP instances introduced in this paper.

	ID	Name	*n*	*p*	*m*	OFV^{BEST}	LS04^{BLP}			SAM15^{MH}		GB21^{MH}	
							OFV^{LB}	Gap [%]	CPU [s]	Gap^{MIN} [%]	CPU [s]	Gap^{MIN} [%]	CPU [s]
Medium-scale	17	XMC10150_100	10,150	100	2	181,472.17	–	–	limit	1.85	limit	**0.00**	limit
	18	XMC10150_1000	10,150	1,000	2	46,617.10	–	–	limit	13.48	limit	**0.00**	limit
	19	XMC10150_2000	10,150	2,000	2	27,670.82	–	–	limit	26.37	limit	**0.00**	limit
	20	FNA52057_100	52,057	100	2	2,099,669.49	–	–	limit	1.82	limit	**0.00**	limit
	21	FNA52057_1000	52,057	1,000	2	629,746.16	–	–	limit	6.46	limit	**0.00**	limit
	22	FNA52057_2000	52,057	2,000	2	410,025.26	–	–	limit	12.55	limit	**0.00**	limit
Large-scale	23	SRA104814_100	104,814	100	2	4,769,940.01	–	–	limit	3.44	limit	**0.00**	limit
	24	SRA104814_1000	104,814	1,000	2	1,482,994.81	–	–	limit	6.10	limit	**0.00**	limit
	25	SRA104814_2000	104,814	2,000	2	1,009,192.64	–	–	limit	9.56	limit	**0.00**	limit
	26	LRA498378_100	498,378	100	2	103,567,233.39	–	–	limit	–	limit	**0.00**	limit
	27	LRA498378_1000	498,378	1,000	2	30,384,292.98	–	–	limit	–	limit	**0.00**	limit
	28	LRA498378_2000	498,378	2,000	2	21,151,169.63	–	–	limit	–	limit	**0.00**	limit
High-dimensional	29	cancer5000_10_784	5,000	10	784	4,900,310.28	4,745,521.58	6.21	limit	**0.00**	2,128.11	0.09	1,201.89
	30	digits60000_100_784	60,000	100	784	94,814,027.62	–	–	limit	1.80	limit	**0.00**	limit
	31	KDD145751_100_74	145,751	100	74	149,117,569.40	–	–	limit	7.21	limit	**0.00**	limit
				Average (without instances 26–28)						7.55		0.01	

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

Table 9
Best first feasible solutions for CPMP instances from the literature.

	ID	Name	*n*	*p*	*m*	OFV^{SAM15}	SAM15^{MH}		GB21^{MH}		Speed-up
							Gap^{MIN} [%]	CPU [s]	Gap^{MIN} [%]	CPU [s]	
Small-scale	1	SJC1	100	10	2	17,288.99	74.50	0.89	**12.55**	**0.14**	6.52
	2	SJC2	200	15	2	33,270.94	54.65	0.86	**12.19**	**0.09**	9.22
	3	SJC3a	300	25	2	45,335.16	43.30	0.87	**12.92**	**0.12**	6.94
	4	SJC3b	300	30	2	40,635.90	46.76	0.87	**15.67**	**0.17**	5.06
	5	SJC4a	402	30	2	61,925.51	52.18	0.93	**11.73**	**0.19**	4.94
	6	SJC4b	402	40	2	52,458.02	35.06	0.92	**11.89**	**0.12**	7.36
	7	p3038_600	3,038	600	2	122,711.17	74.42	**1.26**	**16.70**	4.16	0.30
	8	p3038_700	3,038	700	2	109,677.30	78.95	**1.17**	**19.12**	6.80	0.17
	9	p3038_800	3,038	800	2	100,064.94	84.94	**1.18**	**21.47**	9.56	0.12
	10	p3038_900	3,038	900	2	92,310.09	85.42	**1.18**	**26.56**	35.38	0.03
	11	p3038_1000	3,038	1,000	2	85,854.05	89.42	**1.20**	**24.35**	7.65	0.16
	12	fnl4461_0020	4,461	20	2	1,283,536.73	59.16	1.35	**12.54**	**1.02**	1.33
	13	fnl4461_0100	4,461	100	2	548,909.01	59.80	1.35	**13.09**	**1.25**	1.08
	14	fnl4461_0250	4,461	250	2	335,888.87	67.49	**1.39**	**11.69**	2.48	0.56
	15	fnl4461_0500	4,461	500	2	224,662.49	72.45	**1.42**	**13.56**	2.20	0.65
	16	fnl4461_1000	4,461	1,000	2	145,862.38	80.27	**1.48**	**16.10**	4.78	0.31
						Average	66.17	1.14	15.76	4.76	

Table 10
Best first feasible solutions for CPMP instances introduced in this paper.

	ID	Name	n	p	m	OFV^{BEST}	SAM15^{MH}		GB21^{MH}		Speed-up
							Gap^{MIN} [%]	CPU [s]	Gap^{MIN} [%]	CPU [s]	
Medium-scale	17	XMC10150_100	10,150	100	2	181,472.17	59.84	3.45	**10.40**	**0.26**	13.22
	18	XMC10150_1000	10,150	1,000	2	46,617.10	96.33	3.62	**27.72**	**1.59**	2.27
	19	XMC10150_2000	10,150	2,000	2	27,670.82	126.45	3.96	**43.05**	**3.34**	1.18
	20	FNA52057_100	52,057	100	2	2,099,669.49	78.02	78.27	**12.43**	**1.78**	43.95
	21	FNA52057_1000	52,057	1,000	2	629,746.16	76.47	80.03	**11.24**	**7.75**	10.33
	22	FNA52057_2000	52,057	2,000	2	410,025.26	94.33	82.35	**20.72**	**18.59**	4.43
Large-scale	23	SRA104814_100	104,814	100	2	4,769,940.01	71.44	350.67	**12.64**	**2.59**	135.22
	24	SRA104814_1000	104,814	1,000	2	1,482,994.81	77.86	345.51	**11.51**	**18.11**	19.08
	25	SRA104814_2000	104,814	2,000	2	1,009,192.64	79.44	347.77	**14.09**	**32.95**	10.56
	26	LRA498378_100	498,378	100	2	103,567,233.39	–	–	**47.65**	**25.92**	–
	27	LRA498378_1000	498,378	1,000	2	30,384,292.98	–	–	**89.62**	**128.19**	–
	28	LRA498378_2000	498,378	2,000	2	21,151,169.63	–	–	**104.94**	**201.74**	–
High-dimensional	29	cancer5000_10_784	5,000	10	784	4,900,310.28	29.34	17.85	**22.73**	**3.06**	5.83
	30	digits60000_100_784	60,000	100	784	94,814,027.62	13.48	2,247.98	**4.92**	**64.50**	34.85
	31	KDD145751_100_74	145,751	100	74	149,117,569.40	**22.02**	1,758.51	119.05	**25.36**	69.33
				Average (without instances 26–28)			68.75	443.33	25.87	14.99	

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

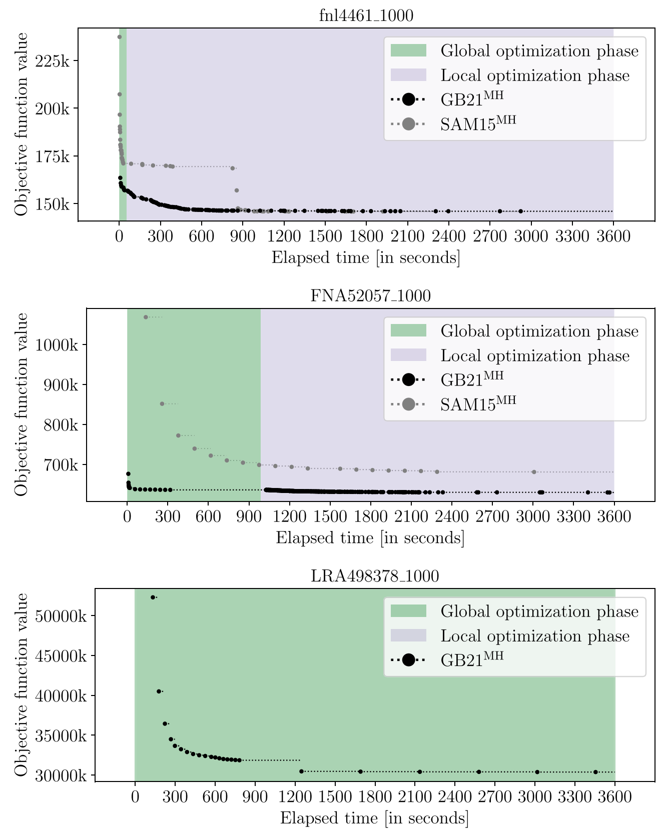

Fig. 3. Improvement of the best feasible solution over time.

Table 11
Best feasible solutions for CCCP instances.

ID	Name	n	p	OFV^{BKS}	GB21^{MH}		
					OFV^{MIN}	Gap^{MIN} [%]	CPU [s]
1	SJC1	100	10	17,359.75	17,363.47	0.02	3.94
2	SJC2	200	15	33,181.65	33,425.61	0.74	4.78
3	SJC3a	300	25	45,354.29	45,470.41	0.26	30.57
4	SJC3b	300	30	40,660.55	40,839.40	0.44	31.04
5	SJC4a	402	30	61,931.60	62,030.00	0.16	64.22
6	SJC4b	402	40	52,202.48	52,551.74	0.67	41.10
7	p3038_600	3,038	600	126,172.76	123,477.60	−2.14	limit
8	p3038_700	3,038	700	113,462.26	111,083.58	−2.10	limit
9	p3038_800	3,038	800	105,352.33	101,738.90	−3.43	limit
10	p3038_900	3,038	900	97,319.54	94,285.58	−3.12	limit
11	p3038_1000	3,038	1,000	89,896.55	87,390.20	−2.79	limit
12	fnl4461_0020	4,461	20	1,282,694.80	1,286,767.42	0.32	limit
13	fnl4461_0100	4,461	100	560,148.77	551,405.33	−1.56	limit
14	fnl4461_0250	4,461	250	348,101.42	337,433.96	−3.06	limit
15	fnl4461_0500	4,461	500	237,491.70	225,789.57	−4.93	limit
16	fnl4461_1000	4,461	1,000	159,672.99	148,551.56	−6.97	limit
					Average	−1.72	

(limit) Time limit of 3,600 seconds reached

Table A.1
Overview of tested cluster-selection procedures.

	First median		Shape of selection	
	Random	Unused capacity	Ball-shaped	Queue-shaped
Procedure 1:	✓		✓	
Procedure 2:		✓		✓
Procedure 3:	✓			✓
Procedure 4:		✓	✓	

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 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 the 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 11, 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^{MIN}), the relative gap between the reported value for

Table A.2
Results for tested cluster-selection procedures.

ID	Name	OFV^{BEST}	Procedure 1		Procedure 2		Procedure 3		Procedure 4	
			Gap^{MIN} [%]	Gap^{AVG} [%]	Gap^{MIN} [%]	Gap^{AVG} [%]	Gap^{MIN} [%]	Gap^{AVG} [%]	Gap^{MIN} [%]	Gap^{AVG} [%]
17	XMC10150_100	181,237.62	**0.00**	**0.23**	0.33	0.37	0.31	0.39	0.13	0.38
18	XMC10150_1000	46,617.10	1.83	2.23	0.79	0.92	1.36	1.83	**0.00**	**0.30**
19	XMC10150_2000	27,670.82	2.43	2.74	0.58	0.96	2.45	3.02	**0.00**	**0.24**
20	FNA52057_100	2,099,669.49	0.00	0.33	0.01	0.33	0.01	0.34	**0.00**	**0.31**
21	FNA52057_1000	629,346.32	**0.00**	**0.25**	0.34	0.59	0.29	0.58	0.11	0.34
22	FNA52057_2000	410,025.26	0.27	0.79	0.31	0.66	0.55	0.82	**0.00**	**0.37**
		Average	0.75	1.10	0.39	0.64	0.83	1.17	0.04	0.32

Table B.1
Further statistics for CPMP instances from the literature.

	ID	Name	n	p	m	OFV^{SAM15}	SAM15^{MH}		GB21^{MH}	
							Gap^{AVG} [%]	Gap^{SD} [%]	Gap^{AVG} [%]	Gap^{SD} [%]
Small-scale	1	SJC1	100	10	2	17,288.99	**0.00**	**0.00**	0.15	0.21
	2	SJC2	200	15	2	33,270.94	**0.00**	**0.00**	**0.00**	**0.00**
	3	SJC3a	300	25	2	45,335.16	**0.00**	**0.00**	0.05	0.07
	4	SJC3b	300	30	2	40,635.90	**0.00**	**0.00**	0.12	0.12
	5	SJC4a	402	30	2	61,925.51	**0.00**	**0.00**	0.32	0.12
	6	SJC4b	402	40	2	52,458.02	**0.00**	**0.00**	0.25	0.06
	7	p3038_600	3,038	600	2	122,711.17	**0.03**	**0.00**	0.11	0.07
	8	p3038_700	3,038	700	2	109,677.30	**0.03**	**0.00**	0.08	0.03
	9	p3038_800	3,038	800	2	100,064.94	**0.03**	**0.00**	0.11	0.07
	10	p3038_900	3,038	900	2	92,310.09	**0.05**	**0.00**	0.07	0.05
	11	p3038_1000	3,038	1,000	2	85,854.05	**0.05**	**0.00**	0.12	0.08
	12	fnl4461_0020	4,461	20	2	1,283,536.73	0.95	0.70	**0.62**	**0.25**
	13	fnl4461_0100	4,461	100	2	548,909.01	2.33	**0.17**	**0.80**	0.30
	14	fnl4461_0250	4,461	250	2	335,888.87	1.37	0.25	**0.62**	**0.12**
	15	fnl4461_0500	4,461	500	2	224,662.49	**0.25**	**0.02**	0.51	0.08
	16	fnl4461_1000	4,461	1,000	2	145,862.38	**0.03**	**0.00**	0.14	0.04
						Average	0.32	0.07	0.25	0.10

Table B.2
Further statistics for CPMP instances introduced in this paper.

	ID	Name	n	p	m	OFV^{BEST}	SAM15^{MH}		GB21^{MH}	
							Gap^{AVG} [%]	Gap^{SD} [%]	Gap^{AVG} [%]	Gap^{SD} [%]
Medium-scale	17	XMC10150_100	10,150	100	2	181,472.17	3.99	1.52	**0.25**	**0.26**
	18	XMC10150_1000	10,150	1,000	2	46,617.10	16.51	2.40	**0.32**	**0.22**
	19	XMC10150_2000	10,150	2,000	2	27,670.82	27.21	1.10	**0.24**	**0.20**
	20	FNA52057_100	52,057	100	2	2,099,669.49	2.46	0.56	**0.31**	**0.39**
	21	FNA52057_1000	52,057	1,000	2	629,746.16	8.39	1.68	**0.26**	**0.19**
	22	FNA52057_2000	52,057	2,000	2	410,025.26	12.75	**0.26**	**0.37**	0.33
Large-scale	23	SRA104814_100	104,814	100	2	4,769,940.01	5.14	1.24	**0.22**	**0.16**
	24	SRA104814_1000	104,814	1,000	2	1,482,994.81	6.75	0.49	**0.04**	**0.03**
	25	SRA104814_2000	104,814	2,000	2	1,009,192.64	10.90	1.46	**0.18**	**0.22**
	26	LRA498378_100	498,378	100	2	103,567,233.39	–	–	**0.84**	**0.61**
	27	LRA498378_1000	498,378	1,000	2	30,384,292.98	–	–	**0.12**	**0.09**
	28	LRA498378_2000	498,378	2,000	2	21,151,169.63	–	–	**0.04**	**0.06**
High-dimen-sional	29	cancer5000_10_784	5,000	10	784	4,900,310.28	0.38	0.39	**0.13**	**0.05**
	30	digits60000_100_784	60,000	100	784	94,814,027.62	2.21	0.30	**0.14**	**0.10**
	31	KDD145751_100_74	145,751	100	74	149,117,569.40	9.94	3.04	**0.15**	**0.11**
						Average (without instances 26–28)	8.89	1.20	0.22	0.19

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

OFV^{MIN} 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.

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 lower bounds on the capacities of the clusters or must-link and cannot-link constraints as in Baumann (2020). 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 (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.

CRediT authorship contribution statement

Mario Gnägi: Conceptualization, Methodology, Software, Data curation, Visualization, Writing - original draft. **Philipp Baumann:** Conceptualization, Methodology, Validation, Supervision, Project administration, Writing - review & editing.

Appendix A. Analysis of different cluster-selection procedures

We tested four different cluster-selection procedures for the local improvement phase. The cluster-selection procedures are summarized in Table A.1. The procedures differ with respect to the selection of the first cluster, and the selection of the remaining clusters. The first cluster is either selected randomly or based on the largest amount of unused capacity. The remaining clusters are either determined by iteratively selecting the cluster whose median is nearest to the median of the cluster selected at last, or by selecting the clusters represented by the nearest medians to the median of the first cluster. While the former procedure results in queue-shaped cluster selections, the latter procedure results in ball-shaped cluster selections. For this analysis, we used the medium-scale instances since we expected the effect of different cluster-selection procedures to be large for these instances. For each of these instances and each cluster-selection procedure, we ran the proposed matheuristic three times with a total running time limit of $\tau^{total} = 3{,}600$ seconds for each run, and we used the default parameter values provided in Table 6.

The results of this analysis are summarized in Table A.2. In the first two columns, we list the number and name of the instances. In the third column, we report the objective function value of the best feasible solution found among all tested cluster-selection procedures (OFV$^{\text{BEST}}$). In the last eight columns, we report for each cluster-selection procedure the relative gap between the objective function value of the best feasible solution found over the three runs and the reported value for OFV$^{\text{BEST}}$, as well as the average relative gap between the objective function value of the feasible solutions found over the three runs and the reported value for OFV$^{\text{BEST}}$. From these results, we concluded that the fourth procedure performs best among all tested procedures. We therefore implemented this procedure in our matheuristic presented in Section 4.2.

Appendix B. Further statistics of computational experiment

In Tables B.1 and B.2, we provide further statistics for the results of our computational experiment. The first six columns of both tables provide information analogously to Tables 7 and 8. In the last four columns of both tables, we report the average and the standard deviation over the three runs of the relative gaps between the objective function value of the best feasible solutions found and the reported value for OFV$^{\text{SAM15}}$ and OFV$^{\text{BEST}}$, respectively.

References

Ahmadi, S., Osman, I.H., 2004. Density based problem space search for the capacitated clustering p-median problem. Ann. Oper. Res. 131 (1–4), 21–43.

Ahmadi, S., Osman, I.H., 2005. Greedy random adaptive memory programming search for the capacitated clustering problem. European J. Oper. Res. 162 (1), 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. Comput. Oper. Res. 39 (7), 1625–1632.

Bachem, O., Lucic, M., Hassani, S.H., Krause, A., 2016. Approximate k-means++ in sublinear time. In: Schuurmans, D., Wellman, M. (Ed.), 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. Comput. Oper. Res. 29 (4), 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.

Baumann, P., 2020. A binary linear programming-based K-means algorithm for clustering with must-link and cannot-link constraints. In: 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 324–328.

Bentley, J.L., 1975. Multidimensional binary search trees used for associative searching. Commun. ACM 18 (9), 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. J. Math. Model. Algorithms 7 (1), 43–58.

Brimberg, J., Mladenović, N., Todosijević, R., Urošević, D., 2019. Solving the capacitated clustering problem with variable neighborhood search. Ann. Oper. Res. 272 (1–2), 289–321.

Carrizosa, E., Guerrero, V., Morales, D.R., 2018. On mathematical optimization for the visualization of frequencies and adjacencies as rectangular maps. European J. Oper. Res. 265 (1), 290–302.

Ceselli, A., Righini, G., 2005. A branch-and-price algorithm for the capacitated p-median problem. Netw.: Int. J. 45 (3), 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. Comput. Ind. Eng. 124, 331–346.

Deng, Y., Bard, J.F., 2011. A reactive GRASP with path relinking for capacitated clustering. J. Heuristics 17 (2), 119–152.

Díaz, J.A., Fernandez, E., 2006. Hybrid scatter search and path relinking for the capacitated p-median problem. European J. Oper. Res. 169 (2), 570–585.

El-Alfy, E.-S.M., 2007. Applications of genetic algorithms to optimal multilevel design of MPLS-based networks. Comput. Commun. 30 (9), 2010–2020.

Erkut, E., Bozkaya, B., 1999. Analysis of aggregation errors for the p-median problem. Comput. Oper. Res. 26 (10–11), 1075–1096.

Espejo, I., Puerto, J., Rodríguez-Chía, A., 2021. A comparative study of different formulations for the capacitated discrete ordered median problem. Comput. Oper. Res. 125, 105067.

Fleszar, K., Hindi, K.S., 2008. An effective VNS for the capacitated p-median problem. European J. Oper. Res. 191 (3), 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 Min. Knowl. Discov. 19 (3), 351–375.

Jánošíková, L., Herda, M., Haviar, M., 2017. Hybrid genetic algorithms with selective crossover for the capacitated p-median problem. CEJOR Cent. Eur. J. Oper. Res. 25 (3), 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. Sci. Rep. 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. Transp. Res. B 26 (5), 365–379.

Kramer, R., Iori, M., Vidal, T., 2019. Mathematical models and search algorithms for the capacitated p-center problem. INFORMS J. Comput. 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. Comput. Oper. Res. 39 (9), 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.N., Senne, E.L.F., 2003. Local search heuristics for capacitated p-median problems. Netw. Spat. Econ. 3 (4), 407–419.

Lorena, L.A., Senne, E.L., 2004. A column generation approach to capacitated p-median problems. Comput. Oper. Res. 31 (6), 863–876.

Mai, F., Fry, M.J., Ohlmann, J.W., 2018. Model-based capacitated clustering with posterior regularization. European J. Oper. Res. 271 (2), 594–605.

Maniezzo, V., Mingozzi, A., Baldacci, R., 1998. A bionomic approach to the capacitated p-median problem. J. Heuristics 4 (3), 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. J. Heuristics 15 (2), 153.

Mulvey, J.M., Beck, M.P., 1984. Solving capacitated clustering problems. European J. Oper. Res. 18 (3), 339–348.

Negreiros, M., Palhano, A., 2006. The capacitated centred clustering problem. Comput. Oper. Res. 33 (6), 1639–1663.

Osman, I.H., Ahmadi, S., 2007. Guided construction search metaheuristics for the capacitated p-median problem with single source constraint. J. Oper. Res. Soc. 58 (1), 100–114.

Osman, I.H., Christofides, N., 1994. Capacitated clustering problems by hybrid simulated annealing and tabu search. Int. Trans. Oper. Res. 1 (3), 317–336.

Pirkul, H., 1987. Efficient algorithms for the capacitated concentrator location problem. Comput. Oper. Res. 14 (3), 197–208.

Puerto, J., Rodríguez-Madrena, M., Scozzari, A., 2020. Clustering and portfolio selection problems: A unified framework. Comput. Oper. Res. 117, 104891.

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

Ríos-Mercado, R.Z., Álvarez Socarrás, A.M., Castrillón, A., López-Locés, M.C., 2021. A location-allocation-improvement heuristic for districting with multiple-activity balancing constraints and p-median-based dispersion minimization. Comput. Oper. Res. 126, 105106.

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 J. Oper. Res. 169 (2), 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. Int. Trans. Oper. Res. 22 (1), 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. Appl. Soft Comput. 13 (9), 3922–3930.

Zhou, Q., Benlic, U., Wu, Q., Hao, J.-K., 2019. Heuristic search to the capacitated clustering problem. European J. Oper. Res. 273 (2), 464–487.