



Efficient Capacitated Clustering algorithm through Convex Polygon Boundary Perturbation

Maria Afsharirad¹

¹Department of Mathematics, University of Science and Technology of Mazandaran, P.O. Box: 48518-78195, Behshahr, Iran

*Corresponding author; email address: m.afsharirad@mail.mazust.ac.ir

Abstract

Capacitated Clustering Problem (CCP) assigns objects to capacitated clusters. Among various CCPs, this work focuses on the capacitated P -median problem (CPMP). Imposing a strict capacity on each cluster, often leads to inefficient overlap for geographical data, which significantly diminish efficiency. We propose a four-stage clustering algorithm, applying convex polygon boundary perturbation to minimize overlap while satisfying capacity constraints. Necessitating a fixed number of clusters in many applications introduces a challenge in meeting capacity constraint. While most methods open a new cluster for points left unassigned, this paper designs a placement algorithm without allowing new clusters. The proposed algorithm demonstrates superior performance compared to three state-of-the-art approaches on specific instances, achieves similar performance on others, with some results reaching optimality. A novel metric is introduced to compare different methods on the same benchmarks.

Keywords: *Distributed network problems, Capacitated clustering, Capacitated P -median problem, Convex polygon*

1. Introduction

In recent decades, Distributed Network Problems (DNPs) have become a major focus in combinatorial optimization, driven by their broad applications in the service and production sectors. Consequently, the development of effective algorithms for DNPs is a critical research area. A prominent subset of these problems, identified in [7], involves clustering geographically distributed nodes into capacitated or uncapacitated groups, such as in the P -Median Problem (PMP) and its capacitated variant (CPMP).

Approaches to address various forms of capacitated clustering problem in DNP typically fall into three categories: exact methods for solving mathematical models, heuristics and matheuristic methods. In exact methods for solving mathematical models, clustering and subsequent step (according to the type of the problem, the next step might be finding medians, routing, etc.) are performed simultaneously. In contrast, heuristic and matheuristic approaches are sequential, where the performance of each step is

contingent on the quality of the previous one. Because exact mathematical models often struggle with the scale of real-world instances, heuristic methods are essential for finding practical and efficient solutions.

As this paper proposes a new clustering method that meets capacity constraint and as it is applicable for all variations of CCP, our focus centers on researches related to CCP. However, since the test instances are from CPMP libraries, we will formally define it in the next section. The CCP, first introduced by [24], is an NP-hard discrete clustering problem. The goal is to select exactly P centers from a candidate set to form clusters, each constrained by a maximum capacity, to serve a set of customers with known demands. According to [28], the CPMP is a special case of the CCP in which the capacity constraint for all clusters are homogeneous and the coefficients of the objective function are distances. Moreover, the set of candidate centers is identical to the set of customer points, whereas in the general CCP, these sets can be different. A comprehensive survey of other CCP variants can be found in [31].

Exact methods based on the mathematical models of CCP, include column generation [20], and branch and cut algorithms, [6, 12]. A recent survey in [27] reviews several mathematical models for the CCP, categorizing them by their applications and solution methods. We do not elaborate these methods, since the proposed method of this paper is based on a heuristic manner.

In contrast, heuristic methods for the CCP typically involve three key steps: clustering, cluster modification, and solution improvement. One of the earliest hierarchical clustering methods was proposed in [33], and a broader classification of clustering algorithms can be found in [7], which groups them into hierarchical, iterative partitioning, graph-based, nearest neighbor, and density-based methods. Although these general-purpose clustering methods are well-established, they typically do not account for capacity constraints, which are essential in problems like the CPMP. Consequently, applying them directly can leave some points unassigned, making a subsequent clustering modification step crucial. Two main strategies exist for this modification:

- **Adding a New Cluster for Unassigned Points:** A prevalent strategy for handling points left unassigned by initial clustering is to group them into a new, additional cluster. While simple to implement, it has two significant drawbacks: it violates the common constraint of a fixed number of clusters (p), and it can substantially degrade the overall solution quality by increasing the objective function value. This strategy is found across various heuristic methods in the literature. For example, the seminal two-phase heuristic by [24] adds a new cluster to accommodate any leftover points after the assignment phase. Similarly, algorithms for related problems, such as the Variable Neighborhood Search for the CCCP [28] and a greedy method for the capacitated location-routing problem [25], also resort to opening new clusters when capacity is exhausted. In a different context, the improved K-means algorithm in [13] may determine a final number of clusters that exceeds the predefined target, effectively creating an unplanned cluster. In all these cases, feasibility is achieved at the cost of violating a primary problem constraint.
- **Regret Function:** This strategy involves moving points between existing clusters, guided by a *regret function*, to create sufficient capacity for unassigned points. A clustering method for the location-routing problem provided in [4] illustrates a hybrid approach. Initially, points are clustered without

capacity limits. Then, a proximity measure is used to identify possible moves between clusters to resolve capacity violations. However, if the capacity constraints are still not met, the method resorts to opening a new cluster. More recently, a two-phase heuristic for large-scale CCPs was proposed in [8]. This work introduces an algorithm to convert an uncapacitated solution to a capacitated one by defining several regret functions to guide the necessary moves between clusters.

Each strategy presents a trade-off. Adding a new cluster is simple but violates the fixed-cluster-count constraint and can degrade solution quality. Conversely, reallocating points with a regret function preserves the cluster count but is computationally more intensive and may fail to place all points. This paper proposes a novel uncapacitated-to-capacitated algorithm for the CPMP that addresses these limitations. While a formal proof of completeness is not provided, our method successfully assigned all points across all benchmark instances tested.

A key challenge that arises during point reallocation is the creation of spatially overlapping clusters, defined here as the intersection of the clusters' convex hulls. Such overlaps degrade both the solution's objective value and its practical interpretability. The primary motivation for focusing on cluster overlap in this paper stems from real-world applications where geographically distributed points are grouped to define service regions. Examples include district zoning for waste collection vehicles or demand-point clustering for network-based service provision. In such contexts, geographical overlap between assigned zones can fundamentally undermine the purpose of regional partitioning. Therefore, we place particular emphasis on minimizing – or ideally eliminating – overlap between clusters. To mitigate this, we introduce a third stage to our algorithm designed specifically to improve cluster structure and minimize overlap. Furthermore, because our initial clustering method is based on constructing convex polygons, the propensity for overlap is inherently reduced from the outset.

It is important to distinguish our work from the Convex Clustering Problem. The goal of convex clustering is to partition data such that each resulting cluster is itself a convex set [16]. Moreover, a dual reformulation for the problem is devised in [30]. In contrast, the CCP does not impose this structural requirement. While our algorithm's initial phase naturally forms convex clusters, these may be altered and lose their convexity during subsequent capacity-driven modifications.

Several metaheuristic methods have been proposed for CCP, [29], [10], [14], [26] and [23]. A genetic algorithm with local search in which all parameters are controlled on-line is devised in [9]. A method based on neural network is proposed in [11] to solve neural clustering problem. They implement their method on the capacitated vehicle routing problem. See [22] for a comprehensive review of heuristic methods for the P -median problem.

As the performance of computers and mathematical programming techniques improve, a new class of algorithms called matheuristics has developed. Simultaneously, with the growth of Mixed Integer Programming (MIP) solvers, these methods managed to solve large amount of combinatorial optimization. Authors in [32] propose a matheuristic approach based on local search and MIP techniques for CPMP. A matheuristic based on cutting-plane neighborhood structure and Tabu search for CPMP is provided in [34]. Authors in [21] provide a model based method for CCP. A matheuristic based on the K -means algorithm is suggested in [5]. Recently, a matheuristic along with a new decomposition strategy for CPMP

has been suggested, [15]. Also, a new version of CCP has been defined, in which customers have different types of services, [2]. They provide a matheuristic based on the GRASP in [1] and a size-reduction method to solve the multi-CCP. Authors in [18] apply integer programming techniques to find the locations of medians and a variable neighborhood search based algorithm for customer assignment phase.

The novelty of this paper is threefold. First, it introduces a method that not only emphasizes clustering optimality, but also explicitly minimizes overlap between clusters. In essence, the proposed approach employs convex polygons to separate clusters from each other as much as possible. Second, after the initial clustering of points, unlike previous methods, the algorithm does not open a new cluster to accommodate unassigned points. Instead, it first applies a boundary perturbation to adjust cluster borders, thereby freeing capacity, and then relocates the unassigned points within the existing clusters. Third, it provides a comparative analysis of two regret functions, demonstrating that different allocation policies are optimal in constrained versus unconstrained clustering stages.

The remainder of this paper is organized as follows. Section 2 formally defines the CPMP and presents its mathematical formulation. Section 3 details our proposed algorithm. The computational complexity is analyzed in Section 4. Section 5 presents numerical results and a parameter analysis. Finally, Section 6 concludes the paper and outlines directions for future research.

2. Problem statement

The Capacitated P -Median Problem is defined on an undirected complete graph $G = (V, A)$, where $V = \{1, \dots, n\}$ is the set of nodes. Each node $i \in V$ is associated with a non-negative integer demand q_i . Integers Q and p are also given as the limited capacity of each cluster and the number of clusters, respectively. The goal is to choose a subset μ of V including p points as medians to attribute all $n - p$ remaining nodes to just one median in a way that the sum of distances of all non-median (customer) points to their median is minimized, while the sum of demands of all points in each cluster do not exceed Q . Figure 1 illustrates a sample of CPMP with its optimal solution.

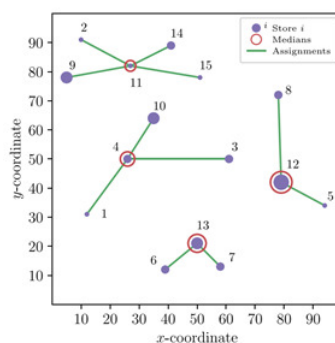


Figure 1. A sample for CPMP, [15]

According to [19], the CPMP can be formulated as follows:

Table 1. Table of Symbols

Notation	Definition
i	index of customer points, $i = 1, 2, \dots, n$.
j	index of medians $j = 1, 2, \dots, p$.
c_j	the median of the j^{th} cluster.
q_i	demand of point i .
Q	the limited capacity of each cluster.
$Cap(j)$	the fulfilled capacity of cluster j .
D	Euclidean distance matrix, D_{ij} is the Euclidean distance between points i and j .
DQ	Distance/demand matrix.
$X \subset V$	Set of customer points.
$\mu \subset V$	Set of medians, ($V = X \cup \mu$).

$$\min \sum_{i \in V} \sum_{j \in V} D_{ij} x_{ij} \quad (1)$$

$$s.t. \sum_{j \in V} x_{ij} = 1, \quad i \in V, \quad (2)$$

$$\sum_{j \in V} x_{jj} = p, \quad (3)$$

$$\sum_{i \in V} x_{ij} q_i \leq Q x_{jj}, \quad j \in V, \quad (4)$$

$$x_{ij} \in \{0, 1\}, \quad i, j \in V, \quad (5)$$

where D_{ij} is the Euclidean distance between points i and j . The binary decision variable x_{ij} equals 1 if point $i \in V$ is assigned to median j , and 0, otherwise. Hence, $x_{jj} = 1$ indicates that point $j \in V$ is selected as the median. The objective function (1) minimizes the total traversed distance. Constraints (2) guarantee that each point is assigned to exactly one median. Constraint (3) enforces the selection of exactly p medians, and constraints (4) control the capacity limitation, preventing the assignment of customers to non-median points. Finally, constraints (5) provide binary condition.

Table 1 introduces the symbols used throughout this paper.

3. Capacitated Clustering by Polygon Perturbation (CCPP)

This section describes the proposed capacitated clustering method called Capacitated Clustering by Polygon Perturbation (CCPP). CCPP is a four-stage heuristic designed to generate high-quality, non-overlapping capacitated clusters. The stages are as follows:

1. Median Initialization (Section 3.1): Initial medians are selected using a classical clustering algorithm applied to a modified distance matrix, which integrates both the spatial distance and the demand of each point.
2. Convex Polygon Clustering (Section 3.2): Points are assigned to the initial medians by constructing

non-overlapping convex polygons. This core step ensures clusters are spatially compact and distinct.

3. Placement of Unassigned Points (Section 3.3): Any points that could not be assigned in the previous step due to capacity limits or polygon constraints (termed left points) are placed into existing clusters by systematically perturbing the cluster boundaries.
4. Solution Refinement (Section 3.4): A final improvement stage is applied to the complete clustering solution to further optimize the objective function value.

Figure 2 provides a schematic overview of the entire CCP algorithm.

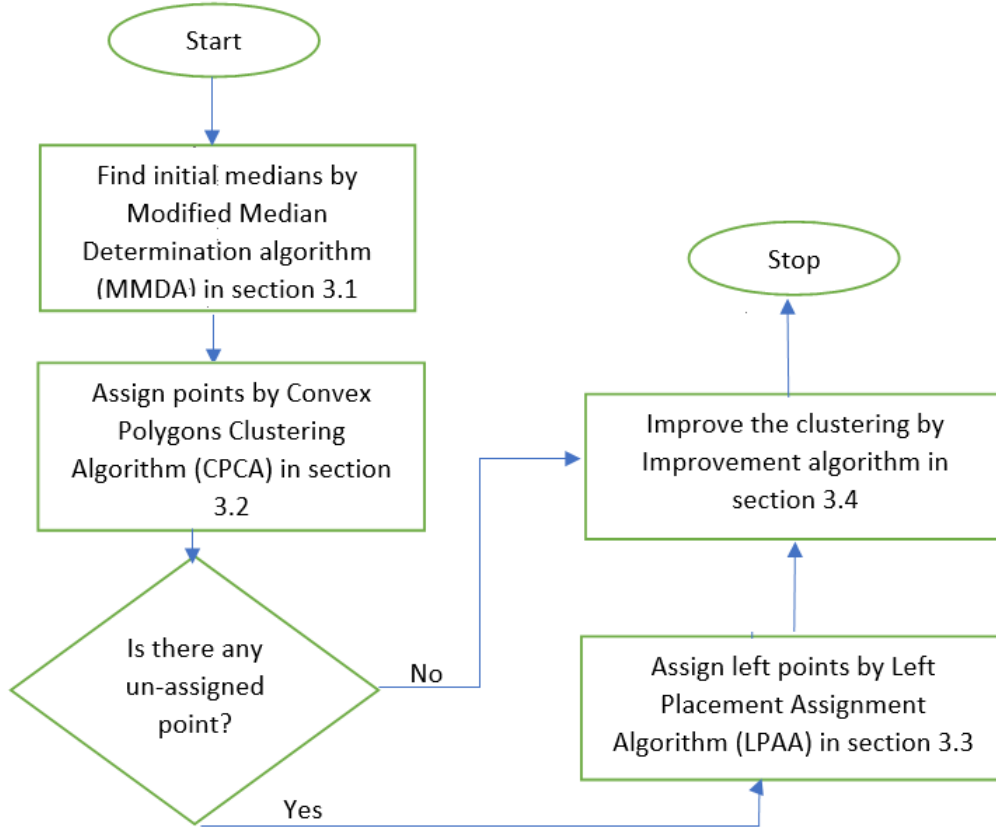


Figure 2. Work flow diagram of the whole algorithm

3.1. Modified Median Determination Algorithm (MMDA)

To ascertain the set of medians μ , a modified classical clustering approach is employed. In this process, the points clustered at this stage are disregarded in subsequent steps, and only medians are applied. Typically, the initial step of clustering involves selecting p points as initial medians. Various methodologies exist for this purpose. Authors in [1] introduce the notion of density as the weighted sum of points close to them. They advocate for the selection of high-density points as initial medians rather than a random initiation. In this paper, the Kmeans++ method in [3] is implemented in step (i), incorporating a blend of randomness and density. In this method, first, a point is randomly selected as the median of the first cluster. The j^{th} median, i.e. $c_j, j = 2, \dots, n$, is selected with the following probability

$$P(c_j = i) = \frac{DN(i)^2}{\sum_{k \in V \setminus \mu} DN(k)^2}, \quad (6)$$

where $DN(i)$ is the distance between point i and its nearest median found so far.

The following steps describe MMDA.

- (i) Choosing initial medians by Kmeans++ method,
- (ii) Assigning customers to medians based on a regret function,
- (iii) Updating medians.

Steps (ii) and (iii) are iteratively repeated until medians remain unchanged over two consecutive steps. This is the classical stopping criteria of K-means based methods, which was also applied in MMDA. In classical methods, the regret function in step (ii) relies on the distance matrix, regardless of cluster capacities, which does not lead to efficient μ . The efficiency of set μ hinges on the placement of medians precisely at the center of points relative to their demands, rather than merely based on distances. To enhance the algorithm's efficacy, the distance matrix (D) is replaced with a distance/demand matrix (DQ), accounting for demand. Incorporating demands imposes the algorithm to determine medians close to high-demand points.

$$DQ_{ij} = \alpha \frac{D_{ij} - \min(D)}{\max(D) - \min(D)} + \beta \frac{q_i - \max(q)}{\min(q) - \max(q)} \quad (7)$$

where $\min(q)$, $(\max(q))$ and $\min(D)$, $(\max(D))$ are the minimum (maximum) values of vector q and matrix D , respectively. In other words, distance/demand matrix DQ is defined as the normalized sum of distance and demand, wherein parameters α and β signify the importance of distance and demand, respectively, contingent on the decision maker and problem conditions. To summarize, Algorithm 1 outlines MMDA.

Input: Set of points $V = \{1, \dots, n\}$, demand vector q and distance/demand matrix DQ .

Output: Set of medians μ .

- 1: Determine the initial median set μ , by the K-means++ method.
- 2: **while** Medians remain unchanged **do**
- 3: Construct distance/demand matrix DQ by (7).
- 4: **while** all points are assigned **do**
- 5: Let $r = \arg \min_{i=1,2,\dots,n} \text{reg}(i)$.
- 6: $c = \arg \min_{j \in \mu} (DQ_{rj})$.
- 7: Assign customer r to cluster c
- 8: Let $DQ_{rj} = \infty$, for all $j \in \mu$, to avoid reassigning point r .
- 9: **end while**
- 10: Update medians.
- 11: **end while**

Algorithm 1. Modified Median Determination Algorithm (MMDA)

In algorithm 1, $\text{reg}(i)$ is a regret function assigning a value to each customer i . Two different regret functions have been applied in this paper. The first, a classic function, wherein $\text{reg}(i)$ denotes the distance

to the closest median to customer i . The second, proposed in [24], is the difference of distance between two closest medians to customer i .

$$reg_{min}(i) = \min_j \{DQ_{ij}\}, \quad i = 1, \dots, n \quad (8)$$

$$reg_{MB}(i) = DQ_{ij'} - DQ_{ij''}, \quad i = 1, \dots, n, \quad (9)$$

where j' is the column number of the lowest value in i^{th} row of matrix DQ , and j'' is the column number of the second lowest value in row i . In Section 5, the choice between these two regret functions is discussed. To improve the quality of the initial medians, we depart from the standard K-means median update rule, which typically selects the point nearest a cluster's geometric centroid. Instead, we update the median by identifying the 1-median of the cluster: the point that minimizes the sum of distances to all other points within that same cluster. This selection criterion is more robust and yields a superior initial set of medians. The final p medians produced by Algorithm 1 serve as the initial medians, provide the foundation for the convex polygon clustering phase detailed in the next section.

3.2. Convex Polygons Clustering Algorithm (CPCA)

This section details the assignment of customer points X to the set of medians μ identified by the MMDA. The same distance/demand matrix represented by DQ is employed to prioritize assigning a distant point from a median with high demand over a low-demand point at the same distance or slightly closer, controlled by parameters α and β . Customer points are assigned sequentially based on this priority until cluster capacities are approached. Consequently, some points may be left unassigned, necessitating placement in the subsequent steps of the algorithm. The lower the demand of an unassigned point, the easier it is to find a suitable location for its assignment.

The clusters are constructed while adhering to capacity constraints. To minimize overlap, the clusters are configured as convex polygons, ensuring that their capacity does not surpass the prescribed limitation and they are not overlapped. Similar to the previous step, the regret function for this stage offers two choices, namely reg_{min} and reg_{MB} (refer to Eq. 8), elaborated further in Section 5. Therefore, row r and column c are selected in the same manner as MMDA, i.e. point r has the lowest regret value and median c is the closest median to it. Point r is assigned to cluster c if two following conditions are met:

1. Assigning point r to cluster c is feasible, i.e. the capacity of cluster c does not exceed its limitation,
2. Assigning point r to cluster c does not result in overlapping clusters.

This process iterates until all points are considered. Therefore, CPCA stops when according to two conditions explained above, all points are considered for assignment. If all points are assigned in this step, the algorithm proceeds directly to the improvement phase in Section 3.4, Otherwise, the algorithm continues to allocate the unassigned points using the method described in Subsection 3.3. At the end of this step, a set of convex polygons representing clusters is established. The clusters generated thus far are guaranteed to be non-overlapping and capacity-compliant. However, this strictness may result in a set of residual points, denoted as *Left*, that could not be assigned without violating these constraints. The

procedure for integrating these remaining points into the existing clusters is detailed in the subsequent section.

3.3. Unassigned Point Placement Algorithm (UPPA)

This subsection describes the Unassigned Point Placement Algorithm (UPPA). At this stage, we have a set of non-overlapping, capacity-compliant clusters in the form of convex polygons. However, there may be points yet to be placed due to capacity or non-overlapping limitations. To accommodate these points, the UPPA makes adjustments to the existing clusters. It is crucial to note that the non-overlapping constraint, which was strictly enforced until now, is relaxed at this stage. This relaxation becomes essential to prevent the allocation of points from becoming infeasible. The initial enforcement of this constraint was vital for establishing a high-quality initial cluster structure.

The UPPA begins by selecting a candidate point from the set *Left* to assign, along with a candidate cluster to receive it. The point with the highest demand, denoted as p , is given the highest priority. To find a candidate cluster for p , the algorithm first investigates if it is located within an existing polygon. If so, an attempt is made to reassign boundary points from that cluster to its neighbors to create sufficient capacity for p . If this is not feasible, or if p is not inside any polygon, the algorithm considers the set of clusters closest to p (denoted as \mathcal{N}). If a cluster in \mathcal{N} has enough capacity, p is assigned. Otherwise, the algorithm initiates the Cluster Perturbation Process (CPP), which is detailed in Algorithm 2. The CPP generates a list of feasible moves from clusters in \mathcal{N} to other clusters, ranking them according to a regret function. A schematic of the UPPA is depicted in Figure 3.

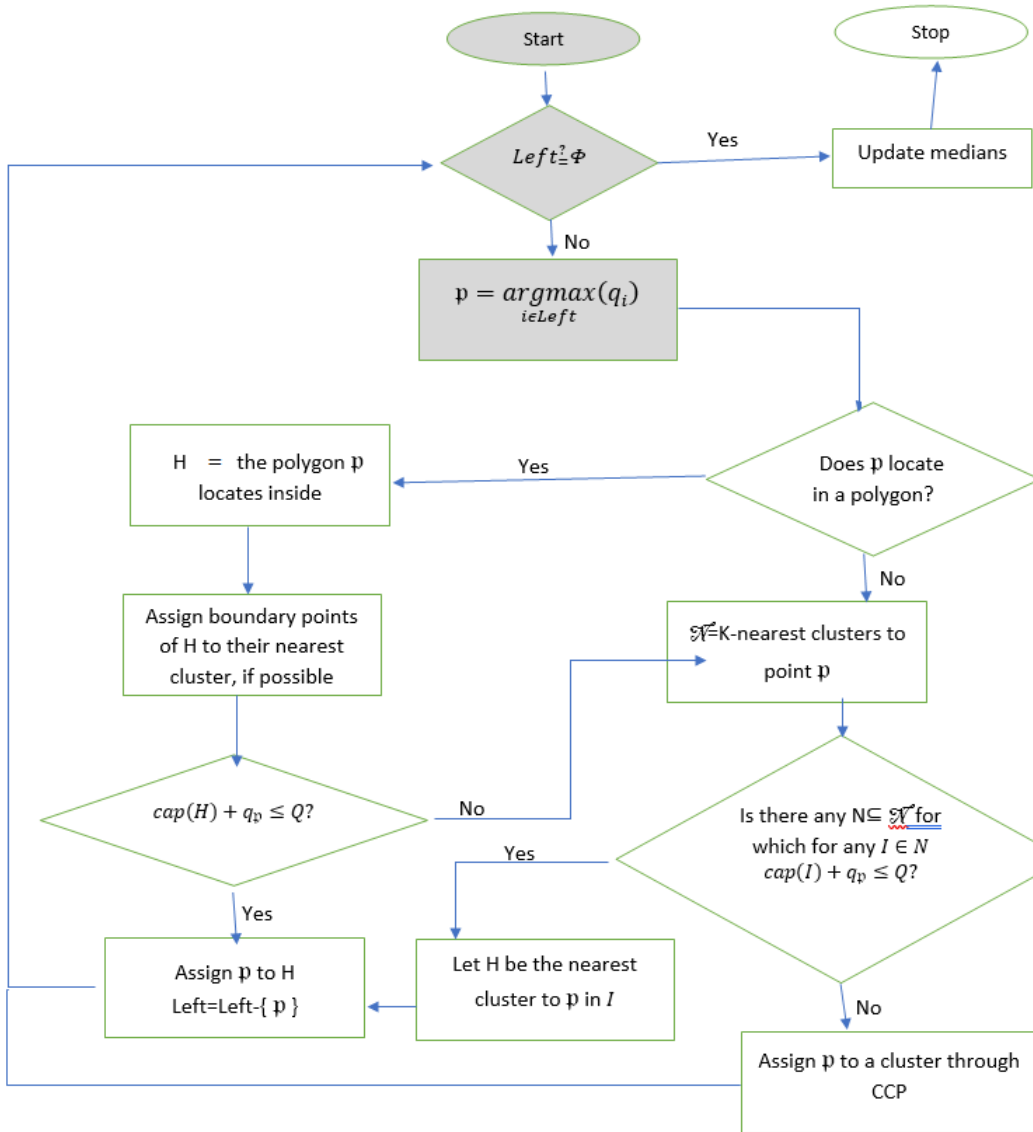


Figure 3. Unassigned Point Placement Algorithm (UPPA)

Note that \mathcal{N} is the set of K nearest clusters to point p . By K closest clusters, we mean K closest medians.

According to UPPA in Figure 3, CCP is recalled when any polygon around the left point p is not empty enough to accept it. In this stage CCP prepares a list of feasible moves in order to create a cluster to include p . The priority for choosing the best move among all feasible moves created in step 15, is determined by its *Regret* value. Performing any move changes three aspects: the capacity of the *Current* and *Neighbor* clusters, and the overall objective function. Figure 4 clarifies this matter.

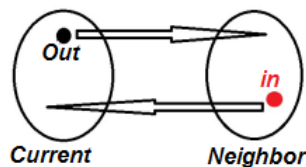


Figure 4. Swapping points *Out* and *in* between clusters *Current* and *Neighbour*

Input: Candidate left point p and set of neighbor clusters \mathcal{N} ,

Output: A cluster containing p without violating capacity constraint.

- 1: Initialize parameters ρ and K ,
- 2: Let $Tabu = \phi$.
- 3: Let $\bar{\mathcal{N}} = \mathcal{N}$
- 4: **while** $\bar{\mathcal{N}} \neq \phi$ **do**
- 5: Let h be the emptiest neighbor in \mathcal{N}
- 6: Let $Tabu = Tabu \cup \{h\}$
- 7: Move any boundary point of cluster h to one of the clusters in $\mathcal{N} \setminus Tabu$. The goal cluster must be vacant enough. If multiple moves are feasible, choose the one which reduces the whole objective function the most.
- 8: $\bar{\mathcal{N}} = \bar{\mathcal{N}} \setminus \{h\}$
- 9: **end while**
- 10: Add point p to the emptiest cluster in \mathcal{N} , say H , neglecting its feasibility.
- 11: **while** there exists an infeasible cluster say H for which $cap(H) \not\leq Q$ **do**
- 12: Let $Tabu = \phi$, $Move = \phi$
- 13: **for** $b \in$ boundary of H **do**
- 14: Let IN be the set of points not further than ρ to b and do not belong to H .
- 15: For any $in \in IN$, if swapping points b and in makes cluster H feasible, add row $\eta = [Current, Out, Neighbor, in, Regret]$ to table $Move$, if this move does not belong to $Tabu$, in which:
 - $Current = H$, $Out = b$, $Neighbor$ = the cluster containing point in ;
 - $Regret$ = value of regret function in this move.
- 16: **if** no new row was added to table $Move$ in the previous step **then**
- 17: Open a new cluster for p .
- 18: Go to step 25.
- 19: **end if**
- 20: **end for**
- 21: Find the row in table $Move$ with lowest $Regret$ and call it η ,
- 22: Add row η to $Tabu$.
- 23: Implement row η , i.e. add point Out to cluster $Neighbor$ and remove it from cluster $Current$, also add point in to cluster $Current$ and remove it from cluster $Neighbor$.
- 24: **end while**
- 25: Let $Left = Left \setminus \{p\}$

Algorithm 2. Cluster Perturbation Process (CPP)

In other words, the move shown in Figure 4, induces the following changes:

$$\begin{aligned} cap(Current) &= cap(Current) - q_{out} + q_{in} \\ cap(Neighbor) &= cap(Neighbor) - q_{in} + q_{out} \\ obj_{new} &= obj_{old} - D_{out,Current} - D_{in,Neighbor} + D_{in,Current} + D_{out,Neighbor} \end{aligned}$$

The first change is not important, since as it was explained in the algorithm, only moves that render cluster H feasible are preserved in the *Move* table. The highest priority is related to the move(s) after which the cluster *Neighbor* is also feasible. The reason is that the process of locating point p stops in this point. But if the algorithm is supposed to be continued, it is clear that we should pick up the move which has the best effect on objective function: reduces it the most or increases it the least. The *Regret* value in Algorithm 2 takes these changes into account. It is defined as follows:

$$Regret = \begin{cases} \frac{obj_{new} - min_{obj}}{max_{obj} - min_{obj}} + \frac{cap_{Neighbor} - min_{infeas}}{max_{infeas} - min_{infeas}}, & r > Q, \\ -\infty, & r \leq Q. \end{cases} \quad (10)$$

where $r = cap_{Neighbor} - q_{in} + q_{out}$ represents the new capacity of cluster *Neighbor* after the intended move. If the move renders cluster *Neighbor* feasible, the *Regret* value takes its lowest value, $-\infty$, so this move has the highest priority. On the other hand, if a move makes the cluster *Neighbor* infeasible, then the value of *Regret* is equal to the normalized sum of the new value of objective function and the new value of filled capacity of cluster *Neighbor*.

It is important to note that the term *swapping* does not necessarily imply that a point is exchanged between clusters. Instead, it includes scenarios where a point, say *out*, transitions to a cluster, say *Neighbor*, but due to its sufficient available capacity, no point exits the cluster. All calculations are the same in this case except that we let $q_{in} = 0$. The algorithm considers all types of moves, including swaps and one-way transfers, when constructing the *Move* table.

It is worth noting at the end of this section that the UPPA terminates when all points in *Leaf*—if any—are assigned to existing clusters or when a new cluster is added at the CPP. However, the condition in Step 16 never occurred in the numerical results section, which serves as evidence of the effectiveness of the CPP.

3.4. Improving clusters

This final subsection assesses the potential improvement of the clusters before implementing them. The process of improving the clusters involves the following steps for all non-median points i :

- (i) Determine K nearest points to i but not in the same cluster.
- (ii) Determine the clusters to which the K nearest neighbors belong, referred to as the set of neighbor clusters NB .
- (iii) List all possible moves where point i is transferred to clusters in NB and calculate the corresponding

change in the objective value.

- (iv) Identify the move in the list that results in the most decrease in the objective value, and implement it if feasible (ensuring the capacity condition is met).
- (v) Update medians.

It is notable that there may be instances where no move leads to a decrease in the objective value. In such cases, no improvement is executed.

4. Computational complexity

The computational complexity of the proposed algorithm (CCPP) is derived from the steps outlined in Figure 2. Initially, the computational complexity of the first step, MMDA (described in Algorithm 1 of Subsection 3.1), is calculated. The determination of initial medians employs Kmeans++, which has a complexity of $O(np \log p)$. Subsequently, a main loop runs for IT iterations until a stopping condition is met. The complexity of each iteration is the sum of three components:

$$O(n^2 \log n^2 + n^2 + (np) \log p + n^2 + n \log n) = O(n^2 \log n)$$

, The term $n^2 \log n^2 + n^2$ accounts for calculating the distance/demand matrix, DQ . The term $(np) \log p$ corresponds to the reg function. The term $n^2 + n \log n$ is associated with the median update step. Finally, the computational complexity of the MMDA algorithm is equal to:

$$O(IT \times n^2 \log n). \quad (11)$$

Now, we consider the second step, CPCA, as described in Subsection 3.2. Initially, For each non-median point i , computing the reg function has a complexity of $O(np \log p)$. Following that, the best value of the reg function is selected in $O(n \log n)$. Subsequently, a polygon is constructed based on this new assignment, ensuring non-overlapping with other polygons and adhering to the capacity constraint.

Convex hull algorithms from standard packages like CGAL and SciPy exhibit a time complexity of $O(h \log h)$ for a given set of h points, where $h \approx \frac{n}{p}$ ¹. Investigating overlap with other polygons costs $p - 1$ steps. The entire procedure continues until all points are assigned or considered as *Left* points, meaning the operations are iterated for all non-median points ($n - p$ times). Finally, medians are updated in $O(n^2 + n \log n)$. In general, the assignment in convex polygons costs:

$$O((n - p) \times (np \log p + n \log n + h \log h + p - 1) + n^2 + n \log n) = O(n^3 \log n). \quad (12)$$

In the third step, the UPPA, outlined in Subsection 3.3, is used to assign left points. Figure 3 illustrates the diagram of UPPA. Initially, the left point with the maximum demand is selected in $O(L \log L)$, where $L = |Left|$. The subsequent steps are carried out for each left point:

¹Available at: <https://www.scipy.org/>

- Does the left point locate in a polygon? Authors in [17] provide an algorithm of $O(\log h)$ to investigate whether a given point locates inside a given polygon with h vertices or not. In the worst case the number of vertices of the polygon is equal to the number of points of that cluster. As it was mentioned earlier $h \approx \frac{n}{p}$. This investigation should be done for any existing p clusters, so this step is of time complexity $O(p \log h)$.
- If the point does not locate inside any polygon, K closets neighbors of it, is found in $O(n \log n)$.
- Transferring boundary points between neighboring clusters to free up capacity costs $O(h \log h)$.
- In the worst-case scenario, if it is not even possible to free up capacity in neighbor cluster, the left point is assigned to the emptiest neighbor, and clusters are made feasible through CPP in Algorithm 2. In this algorithm, the following steps are repeated at most T times:
 1. List all neighbors in ρ radius of all boundary points of the intended cluster (at most h points).
 2. Check the possibility of replacing the intended boundary point with all existing neighbors in the list of the previous step.
 3. Form the table of all possible moves, which takes $O(hn \log n)$.

Hence, the time complexity of the CPP algorithm is $O(T \times hn \log h)$. Moreover, the third step algorithm UPPA costs:

$$o(L \times (p \log h + n \log n + h \log h + THN \log n)) = o(LTn^2 \log n). \quad (13)$$

The clusters are improved in Subsection 3.4. A movement is checked for any $n - p$ non-median point. Firstly, K closest neighbors are found in $O(n \log n)$, then the algorithm finds the cluster to which these K found neighbors belong, which costs $O(K \times n)$. Verifying the feasibility of the related move takes $O(K)$ steps. Finally, medians are updated. As a result, the cluster improvement step is of time complexity:

$$o((n - p) \times (n \log p + pn + p) + n^2 + n \log n) = o(pn^2). \quad (14)$$

In conclusion, based on the relations 11 - 14, the computational complexity of the proposed algorithm is equal to $O(T \times n^3 \log n)$. As mentioned earlier, T represents the required number of moves between clusters to render them feasible. In the computer runs reported in Section 5, the relation $T \ll n$ always holds. Therefore, the computational complexity of the proposed algorithm is equal to $o(n^4 \log n)$.

5. Numerical Results

This section evaluates the performance of the proposed CCP algorithm on two standard benchmark sets for the Capacitated p -Median Problem (CPMP). The first set of testproblems includes 20 instances shown as p1 to p20, in [29]. These benchmark problems are available at <http://people.brunel.ac.uk/mastjjb/jeb>. The second testproblem, introduced in [19], includes 6 samples. To evaluate the algorithm's performance, we employ the relative error as a metric. The relative error is defined as follows:

$$\text{Relative Error} = 100 \times \frac{(\text{obtained objective value}) - (\text{optimal objective value})}{(\text{optimal objective value})} \quad (15)$$

The experiments were implemented in MATLAB 2020 and executed on an Intel Core i7-4810MQ processor under Windows 10.

5.1. Overview of Results

Across both benchmark sets, CCPP demonstrates competitive or superior performance compared to state-of-the-art heuristics and matheuristics. On the first benchmark set [29], CCPP achieves the best-known or optimal solution in 7 out of 20 instances, while remaining within 0.1% of the optimal in all but one case. On the second benchmark set [19], although CCPP does not always reach the best-known objective values, it consistently provides solutions within 6% of the best-known results, often in significantly shorter computation times. These results highlight CCPP's balance between accuracy and efficiency.

To enable a systematic comparison between the proposed CCPP algorithm and other methods across various benchmarks, this paper introduces a novel concept for evaluating any two algorithms on a single problem instance based on specific criteria.

Definition 1. Problem P and instance I of it are given. If algorithms \mathfrak{A} and \mathfrak{B} are solving instance I of problem P and a criteria c (such as time, objective value, etc.) has been reported, then algorithm \mathfrak{A} is said to dominate algorithm \mathfrak{B} in criteria c of sample I , if and only if the reported value for criteria c of \mathfrak{A} is better than the same reported value of \mathfrak{B} in all samples of I . This is shown by $\mathfrak{B} \preceq_I^c \mathfrak{A}$.

5.2. Methodological Considerations

In this section, we outline the methodological choices that underpin our analysis. We explain the rationale behind the selection of key parameters, the normalization strategies employed, and the choice of random seed. All experiments and results presented in the following tables and figures are conducted across all instances of both test sets. Providing these details ensures transparency, reproducibility, and a clear understanding of the results presented in the subsequent sections.

5.2.1. Parameter Sensitivity Analysis

A key feature of CCPP is the use of the distance/demand matrix DQ and the regret functions reg_{min} and reg_{MB} in (8). The first regret function is the classic one, which selects the minimum value of each row (i.e., the closest median to each customer). The second regret function assigns a quantity to each customer, indicating the difference in distances between the customer and its two closest medians. We tested combinations of the weighting parameters (α, β) , with $\alpha = 0, 0.05, 0.1, \dots, 1$ and $\alpha + \beta = 1$, and two regret functions (reg_{min} and reg_{MB}).

Figures 5 and 6 summarizes the impact of these design choices, displaying four piece-wise linear diagrams for four different methods, labeled MB , min , $minMB$, and $MBmin$, corresponding to regret function selection, implemented on the first and second test set, respectively. The methods denoted by min and MB apply the first and second regret functions in (8) for both assignments, respectively. The method labeled $minMB$ applies reg_{min} in the first assignment (Algorithm 1) and reg_{MB} in CPCA (Section 3.2). Conversely, the method $MBmin$ is exactly the opposite.

All four diagrams consist of 20 points, each shows the the mean of the relative error of all samples of the corrpoding sets according to a choice of (α, β) .

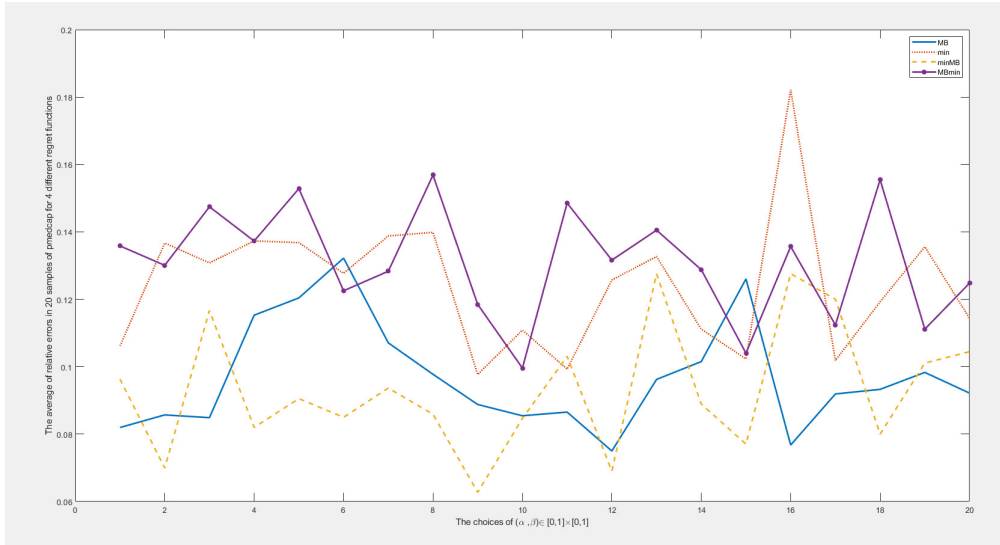


Figure 5. Effect of different regret functions and different coefficients of DQ matrix on the first test set

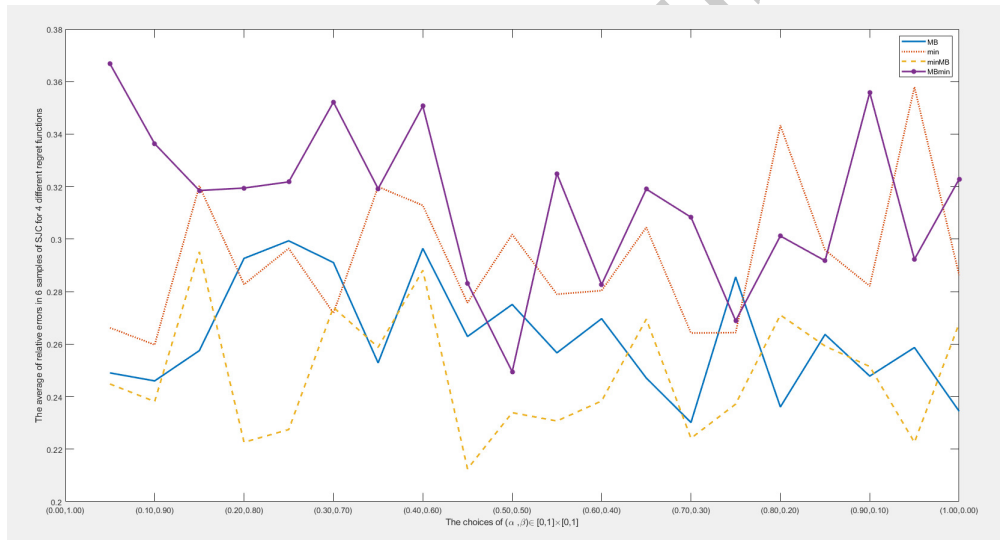


Figure 6. Effect of different regret functions and different coefficients of DQ matrix on the second test set

The figure reveals that the $minMB$ is the top-performing method, with the optimal parameter choice $(\alpha, \beta) = (0.45, 0.55)$ on both test sets. The success of $minMB$ stems from its two-stage assignment strategy. In the first stage, points are assigned solely based on their closest median in the DQ matrix. Since this step has no capacity or overlap constraints, the reg_{min} function is the most effective choice. However, in the second assignment, two constraints come into play: capacity and overlapping. As a result, some points may remain unassigned at the end of this step. In such cases, if there exists a point that is close to a median but far from the next median at hand, it should be given high priority for assignment. This is precisely what the reg_{MB} function takes into account. The flat region around $\alpha \in [0.35, 0.55]$ indicates that CCPP performance is robust to moderate changes in the distance–demand weighting, while extreme values lead to inferior prioritization strategies.

5.2.2. Interpretation of normalization choices

The distance–demand matrix DQ serves as a priority mechanism rather than a direct cost function. Scaling demand in a decreasing manner ensures that high-demand points are considered earlier in the assignment process. In capacitated clustering, postponing the assignment of high-demand points substantially increases the probability of infeasibility and leads to excessive boundary perturbations in later stages. Conversely, prioritizing high-demand points early reduces the difficulty of placing remaining low-demand points and improves overall robustness. This principle explains the observed superiority of decreasing-demand normalization over its increasing counterpart.

Table 2. Effect of demand normalization direction on CCPP performance (instance set 1)

Demand normalization	Avg. relative (%) error (%)	Avg. $Left$ after CPCA	UPPA activations
Decreasing demand (proposed)-First set	0.012	1.4	1.6
Increasing demand-First set	0.035	5.8	4.9
Decreasing demand (proposed)-Second set	0.046	4.2	7.5
Increasing demand-Second set	0.087	8.5	10.4

Table 2, further illustrates the impact of the normalization direction. Increasing-demand normalization refers to scaling demand as $(q_i - \min(q))/(\max(q) - \min(q))$, while decreasing-demand normalization corresponds to the formulation in (7). When demand is scaled in an increasing manner, high-demand points tend to be postponed during the assignment process, resulting in a larger number of unassigned points after the convex polygon clustering stage. Consequently, the algorithm relies more heavily on the perturbation mechanism, leading to inferior objective values. In contrast, decreasing-demand normalization prioritizes high-demand points early, reduces the number of left points, and yields more stable solutions. Results are averaged over all instances in set 1 and all instances in set 2, over 20 runs per instance. Note that all runs are based on the *minMB* method for $(\alpha, \beta) = (0.45, 0.55)$, according to the results of Section 5.2.1 .

5.2.3. Random seed selection

The proposed algorithm includes randomized components, mainly in the initialization stage (e.g., random selection of initial centers). As a result, the quality of the obtained solutions may vary across different executions. To assess the robustness of the algorithm and to select a fixed seed for reproducibility purposes, a dedicated seed sensitivity analysis was conducted. Three different random seeds (2, 75, and 202417) were considered. For each seed, the algorithm was executed 20 independent times on every benchmark instance from both problem sets. For each instance–seed combination, the best and average objective values, as well as the standard deviation over the 20 runs, were recorded. Similar to Section 5.2.2, all runs are based on the *minMB* method for $(\alpha, \beta) = (0.45, 0.55)$, according to the results of Section 5.2.1. To keep the presentation concise, Table 3 reports aggregated results at the benchmark-set level. In particular, the reported standard deviation corresponds to the average of instance-level standard deviations within each benchmark set. Detailed instance-level results are provided in the supplementary material and the public GitHub repository.

Table 3. Aggregated seed sensitivity analysis (20 runs per seed).

Benchmark set	Seed	Best	Average	Std
Set 1 (20 inst.)	2	0	0.015	1.87
	75	0	0.012	2.01
	202417	0	0.021	1.98
Set 2 (6 inst.)	2	0.022	0.41	215
	75	0.028	0.045	220
	202417	0.023	0.043	235

The results indicate that the proposed algorithm exhibits stable performance across different random seeds, with relatively small standard deviations compared to the magnitude of the objective values. Among the tested seeds, seed 75 consistently achieved the best average performance across both benchmark sets. Therefore, seed 75 was fixed and used for all computational results reported in the remainder of this paper, ensuring full reproducibility of the experiments.

5.3. Results for instance set 1

As mentioned earlier, the test problem in this section is CPMP, includes 20 instances shown as p1 to p20, in [29]. These benchmark problems are available at <http://people.brunel.ac.uk/mastjbj/jeb>. The coordinates of the points in this test set have been randomly generated from a uniform distribution [1,100]. The demand values of each point are also random numbers from uniform distribution [1, 20]. The first 10 instances have $n = 50$ and $p = 5$, and the remaining 10 instances have $n = 100$ and $p = 10$.

The proposed algorithm (CCPP) has been compared to four other geometrically-based construction heuristics designed for the CPMP:

1. A heuristic proposed by Mulvey and Beck, 1984, [24] (MB), In this method, the most central points are selected as centres. Initially, p random depots are chosen as the cluster centres. Customer points are ordered according to the second regret function in (8). Customers are assigned to clusters according to the K -means algorithm. Next, the solution is improved by testing all pairwise interchanges. If any points remain unassigned, a new cluster is created for them.
2. A density search constructive method by Ahmadi and Osman, 2004, [1] (DSCM), They define the new regret function based on density, which is the weighted sum of distances. Centers are selected according to their density, instead of being initiated randomly.
3. A model based method by Mai et al., 2018, [21] (EMPR). They propose a constructing heuristic for the CPMP. They apply a Gaussian mixture modelling approach, considering the capacity constraints. Next, they improve the solution by swapping points between clusters.
4. A Tabu search algorithm by Montoya et al., 2019, [23] (Tabu) They apply a Tabu-search based metaheuristic for CPMP.

The reason for presenting the findings from [24] lies in the algorithm's commendable performance in specific instances, despite its age, making it worthwhile to report. In Table 4, the results obtained by four mentioned algorithms, along with the objective value found by the proposed algorithm (CCPP), are reported. The best result from 80 runs has been recorded, adhering to the standard strategy for randomly

initialized clustering methods. Specifically, 80 runs were conducted for the four methods based on the regret functions mentioned earlier, considering 20 pairs of (α, β) . The optimal values, sourced from [21], were obtained using the Gurobi MILP solver for solving the model. The best values in each sample have been highlighted in bold for clarity.

Table 4. Results obtained by CCPP for 80 runs and comparison of best solutions

Sample No.	Proposed (CCPP)	Tabu	EMPR	DSCM	MB	Optimal value	Relative Error
p1	714	717	713	713	713	713	0.001
p2	740	740	740	740	740	740	0
p3	754	752	754	753	764	751	0.003
p4	651	653	651	651	651	651	0
p5	666	664	674	666	666	664	0.003
p6	778	778	778	778	787	778	0
p7	837	796	792	787	788	787	0.063
p8	839	826	822	839	838	820	0.023
p9	718	718	718	724	717	715	0.001
p10	841	844	829	837	838	829	0.014
p11	1026	1054	1009	1006	1015	1006	0.019
p12	966	978	975	970	969	966	0
p13	1035	1032	1026	1056	1026	1026	0.008
p14	1001	1012	983	1009	988	982	0.019
p15	1101	1133	1096	1099	1105	1091	0.009
p16	954	975	956	979	958	954	0
p17	1043	1077	1034	1123	1048	1034	0.008
p18	1048	1086	1058	1062	1053	1043	0.004
p19	1053	1057	1045	1055	1037	1031	0.021
p20	1029	1133	1018	1051	1059	1005	0.023

As evident from the results, the proposed method, CCPP, outperforms all other methods in samples p12, p16, and p18. In samples p2, p4 and p6 the proposed method achieves the best solution along with some other methods. Furthermore, in samples p1 and p9, the objective value of CCPP is only one unit higher than the best-found solution. According to Definition 1, the proposed method is not dominated in objective value by any other method in our list.

Figure 7 depicts two pie charts illustrating the relative error of CCPP and the percentage of instances where it achieved the best solution.

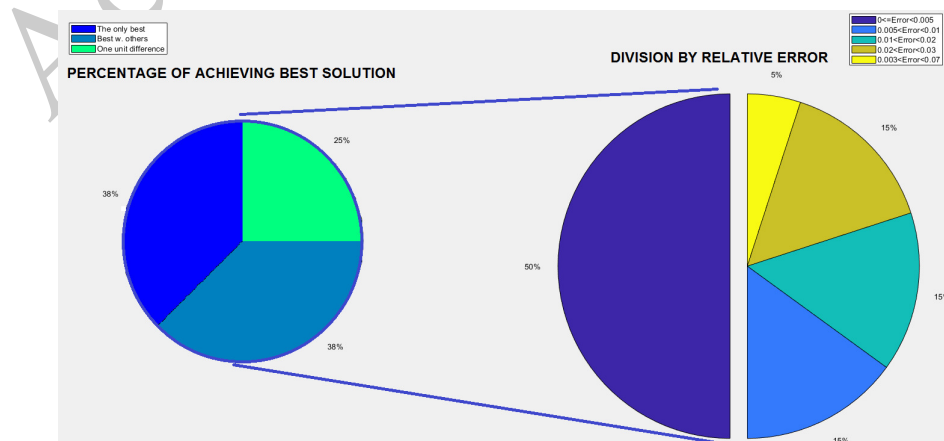


Figure 7. The pie chart of results

As illustrated in the pie chart of Figure 7, in half of the samples, the relative error is lower than 0.005. In 38% of the cases, the proposed method (CCPP) stands as the sole approach in finding the solution closest to the optimal one. In other instances, it either matches the best-known solution or produces a result that is only one unit suboptimal. Furthermore, only 5% of the samples exhibit a relative error greater than 0.003, while lower than 0.07.

5.4. Results for instance set 2

In this subsection the proposed algorithm CCAA is implemented on the test set in [19], including 6 samples. This test set is mainly applied by matheuristic methods, therefore satisfactory results have been reported for them by now. Although the CCPP did not surpass the existing methods in terms of objective value for any of the samples, but it could obtain acceptable results in shorter time for some samples. The reason that we report the results of CCPP on this test set, is to show its capability to find approximated solutions even more quickly than other methods. To be more precise, CCPP devise solutions whose quality is within 6% of the best solution ever known in a shorter time. We further compare the results of CCPP with three methods from the literature.

- A local search based matheuristic by Stefanello et al., 2015, [32] (IRMA),
- A local search based random-key genetic algorithm by Chaves et al., 2018, [9] (A-BRKGA+CS),
- A decomposition based matheuristic by Gnägi and Baumann, 2021, [15] (DBM).

Table 5 show the objective value and CPU time of CCPP and three other methods. The column "relative error" shows the relative error of the obtained objective value by CCPP relative to the best value in the table, according to (15).

It should be noted that just the method *minMB* were run for this test set, due to its superior performance in compare to other regret choices. Moreover all samples were run 20 times for different choices of (α, β) , as mentioned earlier. Although the proposed method achieve better solution compare to 4 other methods in no sample, but it can be seen that it successes to obtain a solution within at most 0.06 of the best found solution in less time than the best-performing algorithm in half of the test cases. Specifically, in samples SJC3a and SJC4a the proposed algorithm delivers a solution with relative error less than 0.06 in nearly half the time required to achieve the best solution.

5.5. Discussion

The results confirm that CCPP is both accurate and efficient:

- On smaller instances (Set 1), CCPP is highly competitive, achieving or matching the best-known solutions in over one-third of the cases.
- On larger, matheuristic-oriented instances (Set 2), CCPP trades a small increase in objective value (at most 6%) for significant reductions in computation time.
- Sensitivity analysis shows that the algorithm is robust across parameter choices, with the *minMB* regret configuration consistently dominating alternatives.

Table 5. Performance comparison of CCPP and approaches from literature

ID	n	K	CCPP		IRMA		A-BRKGGA+CS		DBM		Relative Error	
			Z*	Time	Z	Time	Z	Time	Z	Time	Error	
SJC1	100	10	17789.62	4.63	17288.99	1.9	17359.75	10.20	17363.47	3.94	0.028	
SJC2	200	15	33924.74	3.13	33270.94	3.25	33181.65	38.48	33425.61	4.78	0.022	
SJC3a	300	25	47781.12	13.86	45335.16	23.96	45354.29	104.33	45470.41	30.57	0.053	
SJC3b	300	30	43043.95	14.84	40635.90	2.47	40660.55	120.73	40839.40	31.04	0.059	
SJC4a	300	30	65485.23	21.86	61925.51	56.67	61937.94	172.73	62030.00	64.22	0.057	
SJC4b	402	40	55215.56	22.49	52458.02	6.26	52202.48	175.91	52551.74	41.10	0.057	

- Findings suggest that the proposed algorithm does not depend on finely tuned parameters, and that its performance is driven primarily by the structural prioritization of demand rather than by precise normalization constants.

Overall, CCPP provides high-quality solutions with strong robustness and efficiency, making it a practical choice for capacitated clustering problems.

6. Conclusion

The inherent complexity of the Capacitated Clustering Problem (CCP) necessitates efficient heuristics that can produce high-quality solutions for medium to large-scale instances within a reasonable time. This paper introduces CCPP, a novel four-stage algorithm designed to address this challenge by transforming an un-capacitated clustering into a capacitated one—a significant concern in many DNPs. The core of our method uses convex polygons to guide point allocation, minimizing cluster overlap while maintaining visual coherence. However, during the placement process for left points in the capacitated clustering transformation, we relax the overlap constraint to achieve a feasible clustering.

Our main contributions are threefold:

1. A polygon-based allocation strategy that explicitly addresses spatial overlap—an often overlooked yet critical aspect in capacitated geographical clustering,
2. A strategic disturbance mechanism for polygon boundaries to place leftover points without opening new clusters, enhancing practicality in fixed-cluster applications, and
3. A comparative analysis of two regret functions, demonstrating that different allocation policies are optimal in constrained versus unconstrained clustering stages.

The CCPP method proceeds in four stages: 1) initial median selection using a modified distance matrix, 2) primary point allocation under non-overlapping and capacity constraints, 3) strategic placement of any remaining points by disturbing the polygon boundaries, and 4) final cluster refinement.

In the first and second stages, we utilize a *reg* function, evaluating two different functions in this paper. Our analysis shows that a standard *reg* function, which prioritizes the the minimum element of the DQ matrix, is the most effective in the unconstrained first stage. However, in the second stage, where capacity and overlap constraints apply, the second *reg* function demonstrates superior performance. The rationale behind this lies in the fact that if a point is to remain unassigned, it is advantageous for that point to be the one that, if not assigned to its nearest median, does not have a significant distance from its second nearest median.

Furthermore, our results indicate that weighting demand slightly higher than distance in the distance-demand matrix DQ improves outcomes, as it prioritizes high-demand points that are easier to place. This is due to the fact that, as the demand of remaining points decreases, placing them in the third stage becomes easier. Therefore, the priority for allocation is with a point that is not only close but also has high demand.

Numerical experiments demonstrate the effectiveness of CCP. On the first benchmark set, it outperformed all competing methods in 3 of 20 instances and matched the best-known solution in 4 others, with a maximum relative error never exceeding 0.07. On the second set, CCP consistently achieved solutions within 6% of the best-known results, often in less time. While CCP converged in all 80 executions performed, a formal proof of convergence remains an open direction for future research, as does developing modifications to guarantee it.

References

- [1] AHMADI, S., AND OSMAN, I. H. Density based problem space search for the capacitated clustering p-median problem. *Annals of Operations Research* 131, 1-4 (2004), 21–43.
- [2] ARAÚJO, K., GUEDES, J., AND PRATA, B. Hybrid metaheuristics for the multi-capacitated clustering problem. *RAIRO Operations Research* 56 (2022), 1167–1185.
- [3] ARTHUR, D., AND VASSILVITSKII, S. K-means++ the advantages of careful seeding. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (January, 2007), 1027–1035.
- [4] BARRETO, S., FERREIRA, C., PAIXÃO, J., AND SANTOS, B. Using clustering analysis in a capacitated location-routing problem. *European Journal of Operational Research* 179 (2007), 968–977.
- [5] BAUMANN, P. A binary linear programming-based k-means approach for the capacitated centered clustering problem. In *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (2019, December), 335–339.
- [6] BOCCIA, M., SFORZA, A., STERLE, C., AND VASILYEV, I. A cut and branch approach for the capacitated p-median problem based on fenchel cutting planes. *Journal of mathematical modelling and algorithms* 7 (2008), 43–58.
- [7] BUHRMANN, J. The effects of clustering on the medium and large-scale capacitated location-routing problem. phd dissertation.
- [8] BUHRMANN, J., CAMPBELL, I., AND ALI, M. A capacitated clustering heuristic for large datasets. *Proceedings of the International Conference on Industrial Engineering and Operations Management* (2018).
- [9] CHAVES, A., GONÇALVES, J., AND LORENA, L. Adaptive biased random-key genetic algorithm with local search for the capacitated centered clustering problem. *Computers and Industrial Engineering* 124 (2018), 331–346.
- [10] CHIYOSHI, F., AND GALVÃO, R. A statistical analysis of simulated annealing applied to the p-median problem. *Annals of Operations Research* 96, 1-4 (2000), 61–74.
- [11] FALKNER, J., AND SCHMIDT-THIEME, L. Neural capacitated clustering. *arXiv preprint arXiv:2302.05134* (2023).
- [12] GARCÍA, S., LANDETE, M., AND MARÍN, A. New formulation and a branch-and-cut algorithm for the multiple allocation p-hub median problem. *European Journal of Operational Research* 220, 1 (2012), 48–57.
- [13] GEETHA, S., POONTHALIR, G., AND VANATHI, P. Improved k-means algorithm for capacitated clustering problem. *INFO-COMP Journal of Computer Science* 8, 4 (2009), 52–59.
- [14] GHOSEIRI, K., AND GHANNADPOUR, S. Solving capacitated p-median problem using genetic algorithm. In: *IEEE International Conference on Industrial Engineering and Engineering Management* (2007, December), 885–889.
- [15] GNÁGI, M., AND BAUMANN, P. A metaheuristic for large-scale capacitated clustering. *Computers and operations research* 132 (2021), 105304.
- [16] HOCKING, T. D., JOULIN, A., BACH, F., AND VERT, J. Clusterpath an algorithm for clustering using convex fusion penalties. In *Proceedings of the twenty-eighth international conference on machine learning* (2011), 1.
- [17] HORMANN, K., AND AGATHOS, A. The point in polygon problem for arbitrary polygons. *Computational geometry* 20, 3 (2001), 131–144.
- [18] IRAWAN, C., IMRAN, A., AND LUIS, M. Solving the bi-objective capacitated p-median problem with multilevel capacities using compromise programming and vns. *International Transactions in Operational Research* 27, 1 (2020), 361–380.
- [19] LORENA, L., AND SENNE, E. Local search heuristics for capacitated p-median problems. *Networks and Spatial Economics* 3 (2003), 407–419.
- [20] LORENA, L., AND SENNE, E. A column generation approach to capacitated p-median problems. *Computers and Operations Research* 31, 6 (2004), 863–876.
- [21] MAI, F., FRY, M., AND OHLMANN, J. Model-based capacitated clustering with posterior regularization. *European journal of operational research* 271, 2 (2018), 594–605.
- [22] MLADENVIĆ, N., BRIMBERG, J., HANSEN, P., AND MORENO-PÉREZ, J. The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research* 179, 3 (2007), 927–939.
- [23] MONTROYA, M., VELÁZQUEZ, R., ANALCO, M., FLORES, J., AND LORANCA, M. Solution search for the capacitated p-median problem using tabu search. *International Journal of Combinatorial Optimization Problems and Informatics* 10, 2 (2019), 17–2.
- [24] MULVEY, J., AND BECK, M. Solving capacitated clustering problems. *European Journal of Operational Research* 18, 3 (1984), 339–348.
- [25] NADIZADEH, A., RASHED, S., ALI, S., AND SEYED, M. Using greedy clustering method to solve capacitated location-routing problem. *African Journal of Business Management* 17, 5 (2011).

- [26] NANANUKUL, N. Clustering model and algorithm for production inventory and distribution problem. *Applied Mathematical Modelling* 37, 24 (2013), 9846–9857.
- [27] NEGREIROS, M., MACULAN, N., PALHANO, A., MURITIBA, A., AND BATISTA, P. Capacitated clustering models to real-life applications. In: *Operations Management and Management Science* (2022).
- [28] NEGREIROS, M., AND PALHANO, A. The capacitated centred clustering problem. *Computers and operations research* 33, 6 (2006), 1639–1663.
- [29] OSMAN, I., AND CHRISTOFIDES, N. Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operational Research* 1, 3 (1994), 317–336.
- [30] PI, J., WANG, H., AND PARDALOS, P. A dual reformulation and solution framework for regularized convex clustering problems. *European Journal of Operational Research* 290, 3 (2021), 844–856.
- [31] QUY, T., ROY, A., FRIEGE, G., AND NTOUTSI, E. Fair-capacitated clustering. *arXiv preprint arXiv:2104.12116* (2021).
- [32] STEFANELLO, F., DE ARAÚJO, O., AND MÜLLER, F. Matheuristics for the capacitated p-median problem. *International Transactions in Operational Research* 22, 1 (2015), 149–167.
- [33] WARD, J. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association* 58, 301 (1963), 236–244.
- [34] YAGHINI, M., KARIMI, M., AND RAHBAR, M. A hybrid metaheuristic approach for the capacitated p-median problem. *Applied soft computing* 13, 9 (2013), 3922–3930.

Accepted manuscript