Neural Computing and Applications
https://doi.org/10.1007/s00521-022-08010-w

ORIGINAL ARTICLE

A genetic algorithm integrated with the initial solution procedure and parameter tuning for capacitated P-median problem

Mehmet Kursat Oksuz[1] · Kadir Buyukozkan[2] · Alperen Bal[3] · Sule Itir Satoglu[4]

Received: 6 April 2022 / Accepted: 26 October 2022

Abstract
The capacitated p-median problem is a well-known location-allocation problem that is NP-hard. We proposed an advanced Genetic Algorithm (GA) integrated with an Initial Solution Procedure for this problem to solve the medium and large-size instances. A 3^3 Full Factorial Design was performed where three levels were selected for the probability of mutation, population size, and the number of iterations. Parameter tuning was performed to reach better performance at each instance. MANOVA and Post-Hoc tests were performed to identify significant parameter levels, considering both computational time and optimality gap percentage. Real data of Lorena and Senne (2003) and the data set presented by Stefanello et al. (2015) were used to test the proposed algorithm, and the results were compared with those of the other heuristics existing in the literature. The proposed GA was able to reach the optimal solution for some of the instances in contrast to other metaheuristics and the Mat-heuristic, and it reached a solution better than the best known for the largest instance and found near-optimal solutions for the other cases. The results show that the proposed GA has the potential to enhance the solutions for large-scale instances. Besides, it was also shown that the parameter tuning process might improve the solution quality in terms of the objective function and the CPU time of the proposed GA, but the magnitude of improvement may vary among different instances.

Keyword Location-allocation · Capacitated p-median problem · Facility location · Genetic algorithm · Initial solution algorithm · Parameter tuning

1 Introduction

The p-median is one of the most popular facility location problems in the Operations Research literature [1]. It aims at deciding the location of the facilities and allocating the demand points to single or multiple facilities [2]. Those facilities might be of various types, such as bus terminals [3], mobile communication switches [49], emergency facilities, such as fire stations, hospitals, shelters [4], product distribution centers and so on. The objective of the p-median problem is the minimization of the sum of the total demand weighted distance to be traveled between the medians and the demand points. The main idea is that serving customers from several dispersed facilities instead of a centralized one is more effective [5]. Kariv and Hakimi [6] showed that this problem is NP-hard. However, when the number of medians is fixed, the problem can be solved in polynomial time [2]. When the facilities are capacitated, the problem is called the capacitated p-median problem (CPMP). Since the capacitated p-median problem can be relaxed into the uncapacitated p-median version, it can be regarded as a special case of the p-median. Hence, it can be concluded that the capacitated problem is also NP-hard.

There is extensive literature on the p-median problem. Reese [2] reviewed the past studies according to the

✉ Sule Itir Satoglu
onbaslis@itu.edu.tr

[1] Computer Engineering Department, Erzincan Binali Yildirim University, Erzincan, Turkey

[2] Industrial Engineering Department, Karadeniz Technical University, Trabzon, Turkey

[3] College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait

[4] Industrial Engineering Department, Istanbul Technical University, ITU Ayazaga Kampusu, Rektorluk Binasi, Sariyer, 34467 Istanbul, Turkey

problem type and the solution methods employed. Later, [7] assessed the metaheuristic studies intended to solve the p-median problem. More recent studies are summarized in the Literature Review section of this study. Based on the survey, the metaheuristic studies that intend to solve the medium and large instances of the capacitated p-median problem are limited. Authors believe that the problems are usually large-scale in real settings and solving these instances to optimality is difficult. Therefore, further heuristic studies for solving the large instances of the capacitated version of the problem are strongly needed.

The purpose of this study is to develop an advanced Genetic Algorithm (GA) to solve the medium and large sizes of the capacitated p-median problems. In the past, the large and medium cases were rarely studied. However, in a real-life setting, the problems are usually large-sized, and these cannot be solved optimally. The challenge starts with the metaheuristics where the solvers are not useful. Therefore, this study strengthens the literature by solving medium to large instances of the capacitated p-median problems.

In this study, an Initial Solution Algorithm is first developed and employed to reach good initial solutions. Besides, a 3^3 Full Factorial Design is made where three levels are selected for the factors of the probability of mutation, the population size, and the number of iterations, and parameter tuning is performed to reach a better performance. The objective values of the capacitated p-median problem and the CPU times are considered as response variables. For each parameter level, the proposed GA is run ten times. By using the GA solution results, MANOVA and Post-Hoc tests are performed to identify whether the performance difference between selected parameter levels are statistically significant for each problem. Hence, significant parameter levels are determined, and the GA algorithm could reach better results.

Although there are past studies that suggested the use of the initialization functions integrated into the Genetic Algorithms, such as for the Travelling Salesman Problem [8], none of them employed this for obtaining a better initial population of the GA for solving the capacitated p-median problem. So, the motivation of our study is to enhance the performance of the GAs in terms of solution quality and computational time, by integrating the proposed Initial Solution Algorithm. Besides, as many past studies emphasized the importance of parameter tuning for the metaheuristic algorithms [9], we intended to improve the GAs performance by searching and using the right parameter values of the proposed GA for specific instances.

The unique aspect of this study is that an Initial Solution Algorithm is adapted for the capacitated p-median problem for the first time in the literature, to reach high-quality initial solutions. Hence, the Genetic Algorithm may find better solutions based on these initial ones in shorter computational times. Another contribution of our study is that suitable parameter values of the GA are determined by MANOVA and Post-Hoc tests where both the CPU times and the fitness (objective) function are considered simultaneously. This stage is called “Parameter Tuning” in our study which enhanced the quality of the solutions and computational times of our proposed GA, in most of the instances.

To test the performance of our GA, it was run for the significant parameter levels of the real data set presented by Lorena and Senne [10], and the new data set presented by Stefanello et al. [11]. To enhance the performance of the GA in terms of CPU time and to avoid the trial-and-error process for determining the suitable parameter levels, parameter tuning is applied. The proposed GA was employed for solving the medium and large-size problems, and its performance was found to be better than that of the other metaheuristics in solving several instances, which is a distinguishing aspect of this study.

The paper is organized as follows: The capacitated p-median studies and those that used GA for the p-median problem are reviewed in the Literature Review section. In the Materials and Methods section, the mathematical model of the CPMP, the proposed GA that is integrated with the Initial Solution Algorithm, as well as the Parameter Tuning and Experimental Design stages are explained. Then, for the selected data sets, the Computational Study is explained, and the results are presented and discussed in the fourth section. Finally, the conclusion and future research are presented.

2 Literature review

The facility location or location analysis problem aims at the optimal placement of facilities by minimizing the transportation costs and considering the factors and constraints associated with the specific type of facility. From these facilities, including the central, regional depots, transshipment depots, or a combination of them, commodities are distributed. The CPMP is a facility location problem where the number and locations of facilities are decided by minimizing the total distance traveled [12].

Many exact and heuristic algorithms have been developed to solve the CPMP in the literature. There are also limited studies that proposed robust and stochastic optimization models for the CPMP under uncertainty [13, 14]. The heuristic algorithms may be divided into two categories, namely classical heuristics, and metaheuristics. The classical heuristics for the CPMP may be constructive, local search, and those based on mathematical programming formulations. In this class of heuristics, Lorena and

Senne [10, 15] proposed a local search heuristic based on the Lagrangean/surrogate relaxation techniques and a column generation approach. Under the class of exact heuristics, Baldacci et al. [16] proposed a new exact solution algorithm for solving the CPMP based on a set partitioning of the problem. Besides, the branch-and-bound [17], branch-and-price [18] [19], and cutting plane algorithms [20] were employed. In addition to these, Avella et al. [21] presented an approach based on the generation of general cutting planes.

In the field of metaheuristics, Maniezzo et al. [22] proposed a bionomic approach that integrated the main steps of the bionomic algorithm with a Lagrangean-based lower bound to the CPMP. This method is similar to GA but differs from the use of multiple parents. Ahmadi and Osman [23] proposed a merger of Greedy Random Adaptive Search Procedure (GRASP) and Adaptive Memory Programming (AMP) into a new framework for the CPMP. Diaz and Fernandez [24] proposed Scatter-search and Path-relinking algorithms and applied the combination of both algorithms to the CPMP. [25] proposed a scatter-based heuristic approach for solving CPMP, containing a few user-controlled parameters. Besides, Fleszar and Hindi [26] proposed a variable neighborhood search heuristic that uses lower bounds to assess whether each of the moves during the neighborhood search is useful or not. Shamsipoor et al. [27] developed a new dynamic assignment method based on the urgency function and proposed a new neural network structure. Besides, Hong et al. [28] developed a multi-objective mathematical model for deciding the locations of the emergency relief facilities and explained a real-life example. Gnägi and Baumann [29] proposed a Mat-heuristic for the CPMP. They observed that the algorithm consistently performed better than the state-of-the-art approach for the CPMP on medium and large-size instances.

Moreover, hybrid metaheuristics are employed for the CPMP. In an early study, Osman and Christofides [30] proposed a hybrid metaheuristic that combines simulated annealing and tabu-search algorithms by using a new non-monotonic cooling schedule. Chaves et al. [31] proposed a new hybrid heuristic called Clustering Search to solve CPMP. This heuristic consists of detecting promising search areas based on clustering. In recent years, Landa-Torres et al. [32] proposed a grouping genetic algorithm and a grouping harmony search algorithm to handle complex problems. Both algorithms are hybridized with a customized local search procedure for improving the solution performance. In addition, Yaghini et al. [33] proposed a hybrid metaheuristic which is composed of a cutting-plane neighborhood structure and a tabu-search algorithm. In a recent study, Stefanello et al. [11] proposed a Mat-heuristic approach that combines a local search-based metaheuristic and mathematical programming techniques for the CPMP.

As for genetic algorithms, there has been an increasing interest ever since it was invented by Holland (1992). On the other hand, genetic algorithms were rarely used for solving the CPMP in the literature. However, many researchers proposed the GA for the uncapacitated p-median problem [34–37]. Fathali [36] used 40 test problems from OR-Library and compared the proposed GA with a variable neighborhood search algorithm. Oksuz et al. [37] proposed a GA for the uncapacitated p-median problem and used 15 test problems from OR-Library to test the algorithm. The first study that used GA for the CPMP was performed by Correa et al. [38]. In this study, a new operator called “hyper-mutation” is applied right after the random generation of the initial population and compared with a tabu-search algorithm. The algorithm is used for a specific real word problem which assigns 19,710 candidate students to 26 among a set of 43 available facilities for a university’s admission examination. However, the demand value of the nodes (students) is fixed and equal to 1, which is not suitable for applying the algorithm for the other types of the p-median problem. The second study using the GA for the CPMP was conducted by Ghoseiri and Ghannadpour [39], who proposed the classical assignment method and the assignment through urgencies. Besides, Yang et al. [40] proposed a hybrid bi-level optimization method for the capacitated p-median problem. The proposed method decomposes the problem into sub-problems via a mixed-integer programming model and generates better sub-problems by using a genetic algorithm. It was shown that the hybrid method has good performance for large-scale problems concerning solution time and quality. In addition to the CPMP, the genetic algorithms have been used for other facility location problems, such as bus terminal location [41], e-commerce distribution center location [42, 43], and waste recycling plant location problems [44]

The efficient use of GA is investigated in some studies. Osaba et al. [8] studied the influence of using heuristic initialization functions in GA and implied the efficiency of using heuristic initialization functions. However, they also emphasized that the excessive use of them can decrease the exploration capacity of GA. In another study, Osaba et al. [45] investigated the influence of using blind crossover operators in GA. Performed experimentation showed that the use of blind crossover operators in GA for solving combinatorial optimization problems increases execution time substantially and provides no significant improvement in the results. Paul et al. [46] pointed out the importance of deciding on problem-specific population initialization in GA and studied different population seeding techniques for permutation-coded genetic algorithms. A bio-inspired crossover and mutation are used by Ravichandran et al.

[47] to achieve a desirable amount of protection for real-time medical image security applications. Doerr et al. [48] discussed using different mutation rates, observing that larger mutation rates give significantly better runtimes. In this study, the algorithm was tested by using 10 problem instances presented in the OR-Library. Salcedo-Sanz et al. [49] developed a hybrid-genetic algorithm for optimal capacitated terminal assignment problems associated with the telecommunication networks and attempted to solve instances with up to 100 nodes and 12 medians.

The summary of the capacitated p-median studies is presented in Table 1. For each of the studies, the test data used and the dimensions of the problems are indicated to get a better insight into previous works for the CPMP. The test data are represented by letters, and related references are given below the table. The dimension of the largest size problem in the data set is presented in the 'problem dimension' column as NxP, which are the number of demand points and the medians, respectively.

For further investigation, [7] presented a survey of metaheuristic approaches for solving the classical p-median problems. In addition, Reese [2] summarized the literature on solution methods for the uncapacitated and capacitated p-median problems and presented a glossed bibliography of different solution methods. As a result of this comprehensive literature review, except for the three studies mentioned above, researchers did not use the GA to solve the medium and large-size instances of the CPMP. This is the first study that attempts to solve medium and large-size instances of the CPMP using the proposed advanced GA. The real data set of Lorena and Senne [10] and fifteen test problems of Stefanello et al. [11] were solved very close to the best-known solutions.

3 Materials and methods

In this section, first, the mathematical model of the capacitated p-median problem is presented and shortly explained. Then the proposed Genetic Algorithm integrated with the Initial Solution Algorithm and Parameter Tuning were explained in detail.

3.1 Mathematical model

The standard mathematical formulation of the CPMP is presented in this section. Let i represent the nodes for each site from 1 to *n,* and j represents the potential location sites from 1 to *m*. The potential sites form a subset of the nodes. An integer coefficient d_{ij} describes the distance or cost between two sites i and j. It is also assumed that $d_{ij} \geq 0$ for all i, j. q_i represents the demand of the corresponding node. Q_j represents the capacity of the potential facility location sites. To satisfy the demand of the node, p facilities must be located in p different sites. The objective here is to minimize the total distance while assigning each node to a facility. In the meantime, it must be assured that each facility has enough production capacity to satisfy the

Table 1 Summary of the capacitated p-median studies

Study	Method	Test data set	Problem dimension N×P)
[30]	Hybrid metaheuristic	A	100×10
[22]	Bionomic algorithm	A and 3 gen. data set	100×10, 50×5
[10]	Local search heuristic	A and B	100×10, 402×40
[17]	Branch and bound and price alg	A	100×10
[38]	Genetic algorithm	a real-world app	$19{,}710 \times 46$
[15]	Column generation	B and C	402×40, 3038×1000
[18]	Branch and price algorithm	A	100×10
[24]	Scatter search and path relinking	A, B and D	100×10, 402×40, 737×148
[19]	Branch and Price algorithm	5 generated test prob	500×200
[39]	Genetic algorithm	A	100×10
[26]	Variable neighborhood search	A, B and E	100×10, 402×40, 200×20
[20]	Cutting plane algorithm	A, B and E	100×10, 402×40, 200×20
[27]	Neural network	A and B	100×10, 402×40
[32]	Hybrid metaheuristic	15 gen. test prob	500×50
[33]	Hybrid metaheuristic	A and E	100×10, 200×20
[11]	Mat-heuristics	A, B, C and D	100×10, 402×40, 3038×1000, 737×148

*A: [30], B: [10], C: [15], D: (Stefenallo et al., 2015), E: [16]

demand of all its nodes. The model formulation is presented below [18]:

$$\min \sum_{i}^{n} \sum_{j}^{p} d_{ij} x_{ij}$$

$$s.t. \sum_{j}^{p} x_{ij} = 1 \quad \forall i \in n \tag{1}$$

$$\sum_{i}^{n} q_i x_{ij} \le Q_j y_j \quad \forall j \in m \tag{2}$$

$$\sum_{j}^{p} y_j = p \tag{3}$$

$$x_{ij}, y_j \in \{0, 1\} \quad \forall i \in n, \forall j \in m \tag{4}$$

There are two binary variables in the model where x_{ij} are assignment variables and y_j are location variables. Constraint (1) ensures that each node is allocated to one and only one facility. Constraint (2) ensures that the sum of the demands of the assigned nodes to each facility does not exceed the capacity of the facility that they are assigned. In addition to this, constraint (2) also prevents the assignment of nodes to inactive facilities. Constraint (3) is satisfied with the p number of facilities. Constraint (4) imposes that x_{ij} and y_j assignment variables can take the value of 1 if and only if they are assigned and, 0 otherwise.

3.2 Genetic algorithm

Genetic algorithms (GA) are well-known evolutionary algorithms that mimic the behavior of the genetic processes. Although there are several well-known metaheuristic algorithms, some of them start from a single solution, such as simulated annealing, tabu-search, guided local search [50]. The advantage of the GA is that it is a population-based algorithm and starts the solution process from multiple initial solutions that enhances the diversification in the population. This prevents from being stuck to a local optimum [50].

Through all population-based algorithms, populations must be sufficiently diverse to explore multiple regions of the solution space [51]. Alp and Erkut (2003) mentioned in their pioneer p-median study that Genetic Algorithms work well for the complex optimization problems, as they retain good solutions (individuals) with high fitness function, eliminate poor solutions, and keep diversifying the solutions by the mutation and crossover operators to reach better ones. Corus and Oliveto [51] provided evidence that employing both crossover and mutation operators to evolve the populations helps find better solutions and may improve the optimization time. Therefore, application of both mutation and crossover operators distinguishes the GAs from other population-based algorithms, in achieving the diversity through the solutions. Katoch et al. [50] also mentioned that GAs can be easily hybridized with other optimization methods. Several facility layout problems were solved by using GAs [50]. For these reasons, we employed the GA for this NP-hard optimization problem, in this study.

The Genetic Algorithm associates each chromosome with a solution to the problem and starts with an initial solution set or population that contains a certain number of individuals, i.e., chromosomes. The chromosomes can also be considered individuals. After that, it employs selection, crossover, and mutation operators iteratively to achieve better individuals. Then, it calculates the fitness value of each individual. Thus, it generates neighbor solutions and intends to reach better solutions. The reader should refer to (Holland, 1992) and [52] for further information about the GA. The notation used in the proposed GA is described below:

APL: Assignment priority list
Fitness_1, Fitness_2: The Fitness values calculated for the considered individuals.
L_{i1}: The nearest median point with sufficient capacity for node-i
L_{i2}: The second nearest median point with sufficient capacity for node-i
D_i: The difference between the distance from i to L_{i1} and the distance from node-i to L_{i2}.
CL_{i1}: Unused capacity for median point L_{i1}.
CL_{i2}: Unused capacity for median point Li2.
$C_{assigned}$: The number of assigned demand points.
$C_{assigned}$: The number of demand points that could not be assigned to the closest median.
M: A big penalty value.

The proposed GA starts with the definition of the values of the necessary parameters, namely, the population size (*p*_size), mutation probability (mp), and maximum iterations (max_iter). Then, the algorithm calls the Initial Solution Algorithm which intends to provide better initial solutions and helps the GA reach near-optimal solutions in short computational times. The pseudocode of the proposed GA is described in Fig. 1.

3.2.1 The initial solution algorithm

Within GAs, the selection of the parents in other words former solutions to produce new solutions is not a trivial stage and affects the performance of the algorithm (Alp and Erkut, 2003). Therefore, we enhanced this stage by proposing an Initial Solution Algorithm. The proposed GA starts with the Initial Solution Algorithm that is illustrated in Fig. 2. It is adapted from Mulvey and Beck [53]. For the

```
Proposed GA
{
    Apply Initial Solution Algorithm
    Evaluate population
    Store best individual as "Best_Solution"
    Iter=0
    Repeat
    {
        Iter=Iter +1
        Apply Selection procedure on population_Iter
        Apply Crossover procedure on population_Iter
        Apply Mutation procedure on population_Iter
        Evaluate population_Iter
        Store "best_individual" of population_Iter
        If Best_individual < Best_Solution then
            Best_Solution = Best_individual
        end if
        population_Iter+1 = population_Iter
    }
    Until( Iter < Max_Iter )
    Bring the "Best_Solution" as a final solution
}
```

Fig. 1 Pseudocode of the proposed GA

first individual, p-medians are randomly selected, and the initial values of *Fitness_1* and *Fitness_2* are set. Later, the *Fitness Evaluation Stage-1 is* run. This stage is illustrated in Fig. 3.

The Fitness Evaluation Stage-1 will be explained at first. The fitness value of an individual is obtained by calculating the value of the objective function given in the mathematical model. To perform this calculation, it is needed to know which node will be re-assigned to which median, indicated by the x_{ij} binary variable. The procedure to assign the nodes to medians that is proposed by Correa et al. [38] is adapted to this study. First, all distances from nodes to median points are calculated (d_{ij} (i = 1,..,n; j = 1,..,p. Then median points are sorted in ascending order according to d_{ij} for each node and recorded in the M_{ij} vector. For example, given a problem of five medians, if the distances of the first node to the five medians are d_{1j} = 15–22-18–35-7 then M_{ij} = [5, 1, 3, 2, 4]. After that the distance value D_i is obtained by calculating the difference between the distance from the ith node to L_{i1} and that distance from i to L_{i2}, for each node. Here, L_{i1} is the nearest median point with sufficient capacity, while L_{i2} is the second nearest median point that has sufficient capacity. These nodes are sorted according to the D_i values in descending order and recorded in APL. According to the order of this list, nodes are attempted to be assigned to the nearest median point. If the demand value (q_i) of node-i is greater than the capacity (CL_{i1}) of the L_{i1} median point, it then proceeds to other nodes without assigning the ith node. After these operations are made for all the nodes, if all nodes are not assigned, M_{ij} and D_i values are recalculated for the unassigned nodes, and APL has generated again. This process continues until all nodes are assigned.

During the first stage of the fitness evaluation procedure, if there is only one median with sufficient capacity for an unassigned node, then the D_i value will be equal to the distance from *i*th node to L_{i1}. In addition, if there is no median with sufficient capacity for an individual, then this individual will be punished with a major penalty, and the rest of the transaction will be skipped for this individual. The procedure will continue for the next individuals. Thus, the GA eventually ensures that the punished individuals will be eliminated from the population for the next generations.

As a result of the *Fitness Evaluation Stage-1*, an individual comprising the selected medians is returned. Then, within the *Initial Solution Algorithm*, for each individual returned, the center point is calculated again for each median, and nodes are assigned to this median. If a new median point is found, which can improve the fitness value, then the median point of this individual is changed. If any of the medians of the individual change, the evaluation procedure is restarted. After that, if the solution obtained from the first stage of the Fitness Evaluation Procedure is better, the central points are recalculated. Otherwise, this individual is recorded into the population, and this process is repeated as much as the population size. The initial solution procedure is terminated after all the individuals are determined. Thus, the *Initial Solution Algorithm* returns the (initial) population. Among the initial solutions obtained through the *Initial Solution Algorithm*, the best individual is determined as the *Best Solution*. The number of iterations is increased by one. Then, the *Selection operator is* run to select the individuals of the next generation.

3.2.2 Selection

The ranking-based selection method [38] is applied using Eq. (5) to select the individuals for the next generation. In this equation, *R* is the list of individuals ranked in ascending order according to the fitness value, and the length of R is equal to *p_size*. *Rnd* is a random number generated between 0 and 1. $\lfloor b \rfloor$ symbol used in Eq. (5) represents the largest integer which is smaller or equal to *b*. Equation (5) gives the sequence number (*j*) of the individual which will be selected from the list R. For the selected individuals, the crossover operator is applied.

$$\text{Select}(R) = \left\{ \begin{array}{l} r_j R | j = p\_\text{size} \\ \qquad - \left\lfloor \dfrac{-1 + \sqrt{1 + 4\text{rnd}\left(p\_\text{size}^2 + p\_\text{size}\right)}}{2} \right\rfloor \end{array} \right\} \tag{5}$$

Fig. 2 Flowchart of the Initial Solution Algorithm

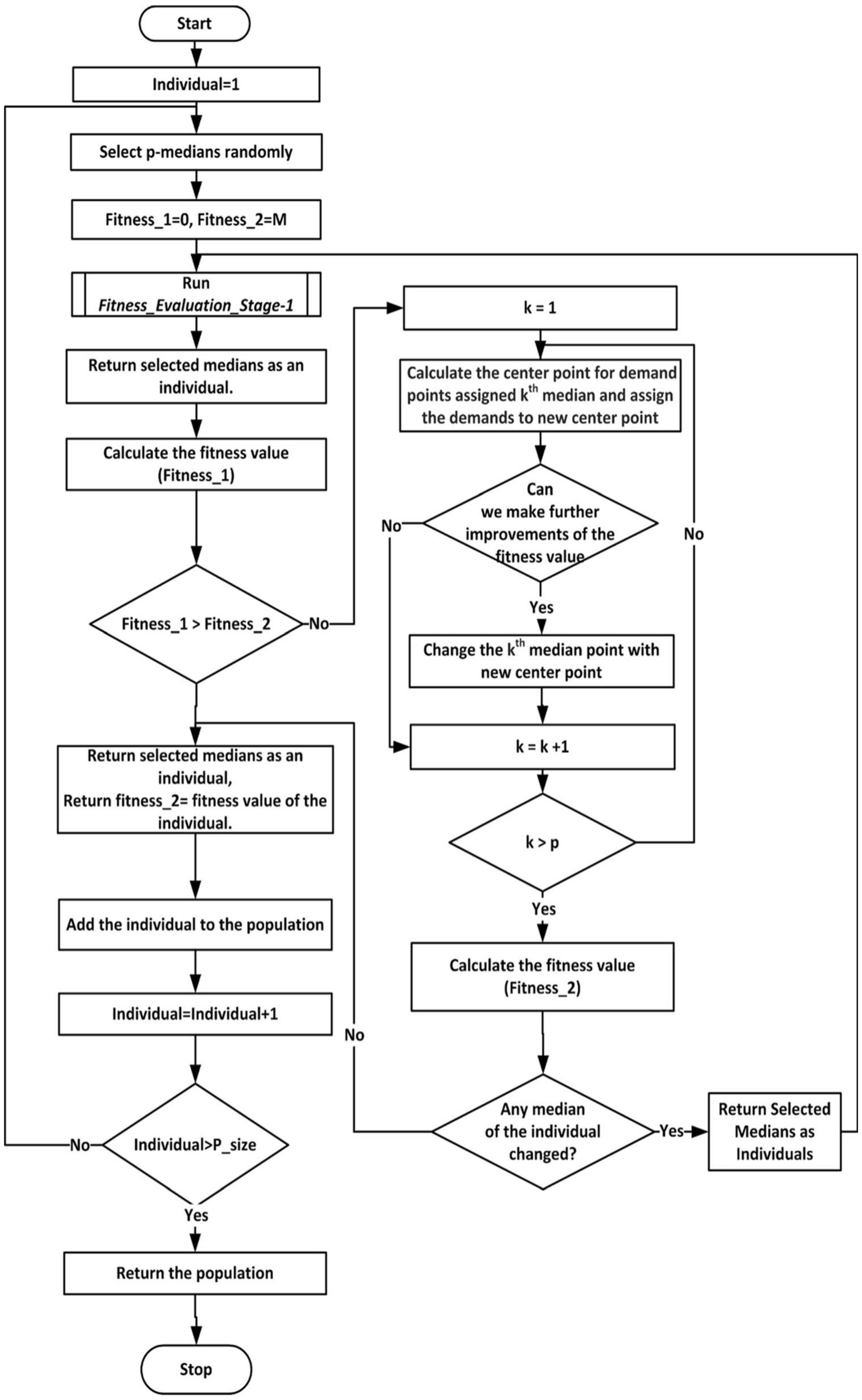
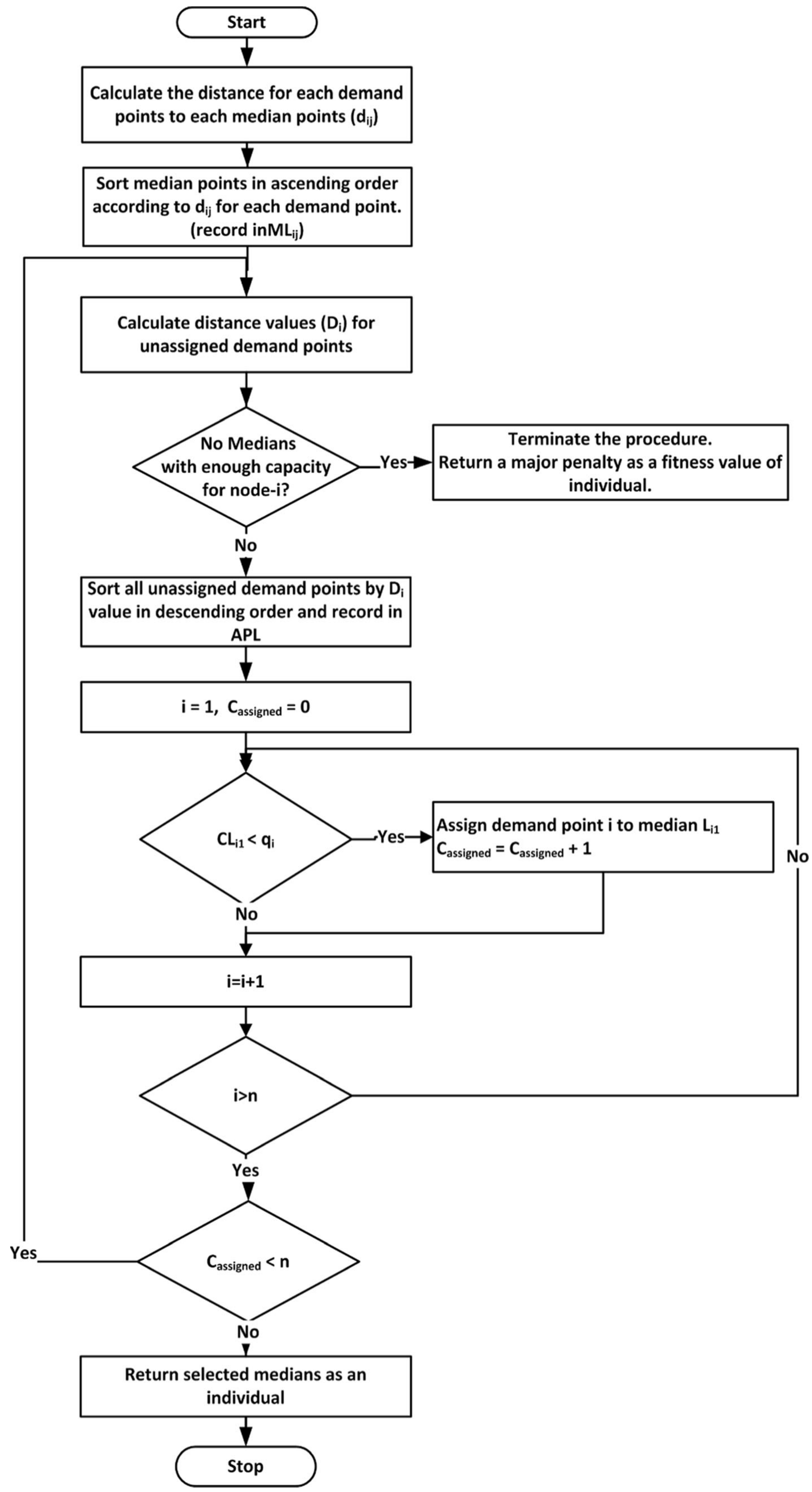

Start
Individual=1
Select p-medians randomly
Fitness_1=0, Fitness_2=M
Run Fitness_Evaluation_Stage-1
Return selected medians as an individual.
Calculate the fitness value (Fitness_1)
Fitness_1 > Fitness_2
No
k = 1
Calculate the center point for demand points assigned kth median and assign the demands to new center point
Can we make further improvements of the fitness value
No
Yes
Change the kth median point with new center point
k = k +1
k > p
No
Yes
Calculate the fitness value (Fitness_2)
Any median of the individual changed?
Yes
Return Selected Medians as Individuals
No
Return selected medians as an individual, Return fitness_2= fitness value of the individual.
Add the individual to the population
Individual=Individual+1
Individual>P_size
No
Yes
Return the population
Stop

Fig. 3 Flowchart of the Fitness Evaluation Stage-1


Start
Calculate the distance for each demand points to each median points (d_{ij})
Sort median points in ascending order according to d_{ij} for each demand point. (record inML_{ij})
Calculate distance values (D_i) for unassigned demand points
No Medians with enough capacity for node-i?
Yes
Terminate the procedure. Return a major penalty as a fitness value of individual.
No
Sort all unassigned demand points by D_i value in descending order and record in APL
$i = 1$, $C_{assigned} = 0$
$CL_{i1} < q_i$
Yes
Assign demand point i to median L_{i1}
$C_{assigned} = C_{assigned} + 1$
No
No
$i=i+1$
$i>n$
Yes
Yes
$C_{assigned} < n$
No
Return selected medians as an individual
Stop

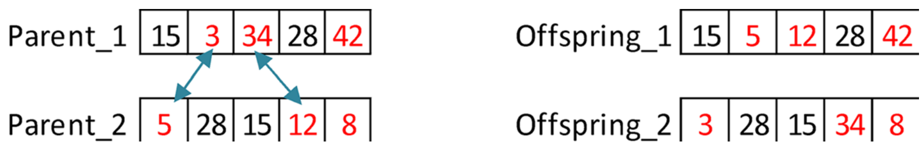


Fig. 4 An instance of the crossover operation

3.2.3 Crossover

Crossover is performed after pairing the selected individuals. The crossover operator in Correa et al. [38] is adapted to the proposed algorithm. In this process, the crossover is applied to all individuals but not the same ones. In this operation, some genes that are not the same are exchanged with each other. The number of genes that will be replaced for each pair is determined using a '*k*' parameter obtained by a value generated randomly. '*k*' takes a value between 1 and the non-identical number of genes. The crossover operator is illustrated by an instance in Fig. 4. In this example, *k* is assumed as two while the non-identical gene number is three. After the crossover operator, the mutation operator is employed by considering the mutation probability to reach better solutions.

3.2.4 Mutation

An individual's probability of mutation is represented by the '*mp*' parameter. A random number is generated between 0 and 1 for each individual after the crossover phase. If the generated random number is smaller than the *mp* value, the mutation process is performed for the concerned individual. In this process, a randomly selected median is replaced with a randomly selected node.

3.2.5 Fitness evaluation

After both crossover and mutation operations are applied, the Fitness value is recalculated and evaluated for the new individuals through the *Fitness Evaluation* step of the proposed GA. To be accepted into the population, new individuals must have a better fitness value than the worst individual in the population. This step consists of Fitness Evaluation Stage-1 and Stage-2. The Fitness Evaluation Stage-1 is the same as the stage of the Initial Solution Algorithm explained in Sect. 4.

Later, the Fitness Evaluation Stage-2 is applied. Once the assignments are made in the Fitness Evaluation Stage-1, the algorithm proceeds to the second stage. All nodes that could not be assigned to their closest median are switched one by one with the other nodes that are already assigned to the other medians. Thus, it is intended to reach a better solution. For this operation, only the medians whose remaining capacity is lower than the demand of the considered nodes is taken into account.

As a result of the Fitness Evaluation stages, the individual with the best fitness value is returned, which is compared with the current Best Solution. If it is better than the current Best Solution, the Best Solution is updated. Otherwise, the Best Solution stays the same. These main steps explained above are repeated as much as the Maximum Iterations. As soon as the iterations are completed, the Best Solution is returned, and the GA is terminated.

3.3 Design of experiments and parameter tuning

The parameters of a heuristic algorithm may have a great influence on the desired output. Moreover, the time required for the parameter setting of an algorithm sometimes far exceeds the development time [54]. Some researchers stated that parameter tuning increases the algorithm's performance [9, 33, 55]. Despite this fact, parameter tuning is usually neglected in most heuristic studies.

In this study, a statistical design of experiments (DOE) was conducted to determine the parameter levels of the proposed GA for specific data sets and thus obtain better results. For this purpose, a 3^3 Full Factorial Design is performed where three levels are selected for the probability of mutation, the population size, and the number of iterations. The objective (fitness) value and the CPU time are considered response variables. To determine the parameter levels, firstly, each problem was solved ten times with the different levels of parameters. This process was continued as long as a better result was achieved. Then, three levels with the best result were determined for each parameter. As a result of experiments, the population sizes are taken as 20, 30, 40; the probability of mutations is taken as 0.1, 0.3, 0.5 and the maximum number of iterations is taken as 200, 250, 300 for the problem size up to 402 demand points and 40 medians. For the other larger-size problems, the same process was conducted, and the population sizes are taken as 40, 60, 80; the probability of mutations are taken as 0.3, 0.5, 0.8, and the maximum number of iterations is taken as 300, 400, 500. It is noticed that if the problem size increases, it gets harder to obtain reasonable solutions. Therefore, higher parameter levels for the algorithm are needed to get better results.

Ten runs were conducted for each combination of the factor levels. MANOVA and Post-hoc tests which are Duncan [56] and Tukey [57] are performed through the SPSS software by using the GA solution results to identify whether the performance difference between selected parameter levels are statistically significant for each problem set. The confidence interval is set to 95%, and hypotheses are tested. Hence, significant parameter levels are determined and input into the algorithm. The selected parameters according to the DOE are shown in Table A1. The proposed GA is tested by using the significant parameter levels of the real data set and the new data set. To show the impact of the parameter tuning on the performance of the proposed GA, the results of real data set instances obtained before and after parameter tuning were compared, in the following section. The MANOVA and Post Hoc tests and the interpretation of the test results were explained in detail the following section for a problem instance, where the best parameter values were determined.

4 Computational study

Two data sets were used to test the performance of the proposed algorithm. The first one is a real data set taken from [58] and presented by Lorena and Senne [10]. It consists of six instances with up to 40 medians and 402 demand points. The set of problems is based on the data collected through the Geographical Information Systems Software for deciding the facility locations within the central area of Sao Jose dos Campos City (SJC). The second data set was taken from [59] and presented by Stefanello et al. [11]. It consists of fifteen test problems with up to 200 medians and 724 demand points. The reason for choosing these data sets is to compare the proposed algorithm performance with other algorithms using these data sets.

The proposed GA was coded and implemented in MATLAB®, and the computational tests were made on i7-4500U CPU 2.0 GHz personal computer. The results of the GA for the two data sets are presented in Tables 2 and 3, respectively. Moreover, the CPU times and the gap between the best solutions and solutions obtained by the GA are reported.

The real data set of Lorena and Senne [10] was used to test the performance of our algorithm in medium-size problems, and the results are shown in Table 2. According to the results, the optimum solution for the problem SJC2 was obtained by the proposed GA, which could not be reported previously by any heuristic algorithm in the literature. Besides, near-optimal solutions were obtained for other problems at reasonable CPU times. The average gap

Table 2 Results for the real data set instances

Problem	NxP	Optimum	GA		VNS		SS + PR		NN		GAP %			
			Obj. Val	Time (sec.)	Obj. val	Time (sec.)	Obj. val	Time (sec.)	Obj. val	Time (sec.)	GA	VNS	PR + SS	NN
SJC1	100 × 10	17,252	17,579	31	17,288,99	50,5	17,288,99	13,57	17,289,9	20,3	1,895	0.214	0.214	0.220
SJC2	200 × 15	33,187	33,187	155	33,270,94	44,8	33,287,98	95,17	33,452	85,27	0.000	0.253	0.304	0.799
SJC3a	300 × 25	45,203	45,638	248	45,335,16	8580,3	45,349,34	425,27	46,145,1	460,12	0.962	0.292	0.324	2,084
SJC3b	300 × 30	40,505	40,853	621	40,635,9	2292,9	40,684,14	756,44	40,839	227	0.859	0.323	0.442	0.825
SJC4a	402 × 30	61,751	62,642	792	61,925,51	4221,5	62,030,3	2393,7	62,606,12	1169,9	1,443	0.283	0.452	1,385
SJC4b	402 × 40	52,286	53,180	921	52,469,96	3471,4	52,594,08	2712	53,178,13	906	1,710	0.352	0.589	1,706
Average											1,145	0.286	0.388	1,170

GA Genetic Algorithm, *NN* Neural Network, *SS* + *PR* Hybrid Scatter Search and Path Relinking, *VNS* Variable Neighborhood Search

Table 3 Results for the new data set instances

Problem	NxP	Best-Known	GA		MatHeu		GAP %	
		Solution	Obj, val	Time (s)	Obj, val	Time (s)	GA	MatHeu
P1	318 × 5	180,281	180,281	5,2	180,281	9,15	0.000	0.000
P2	318 × 15	88,901	89,142	256	88,901	26,35	0.271	0.000
P3	318 × 40	47,988,38	48,285	724	48,040,24	222,41	0.606	0.108
P4	318 × 70	32,198,64	33,085	1750	32,290,39	127,45	2,753	0.285
P5	318 × 100	22,942,69	23,958	2432	23,639,7	222,65	4,425	3,038
P6	535 × 5	9956,77	9956,77	6,3	10,337,57	7,08	0.000	3,825
P7	535 × 25	3695,15	3716,3	416	3770,45	311,36	0.572	2,038
P8	535 × 50	2461,41	2509,9	2358	2497,05	377,28	1,970	1,448
P9	535 × 100	1438,42	1501	5466	1454,48	362,75	4,351	1,117
P10	535 × 150	1032,28	1108	9078	1044,6	366,54	7,916	1,193
P11	724 × 10	181,783	181,783	410	184,031,2	6,64	0.000	1,237
P12	724 × 30	95,034,01	95,676	814	96,513,51	158,05	0.883	1,557
P13	724 × 75	54,735,05	56,363	3546	54,742,43	507,56	2,974	0.013
P14	724 × 125	38,976,76	40,749	10,196	38,992,44	509,01	4,547	0.040
P15	724 × 200	28,079,97	27,203	12,187	28,117,06	508,81	−3,123	0.132
Average							1,824	1,069

between the optimum solutions and the solutions obtained by the GA is about 1.145%, for this data set.

Moreover, the results of the proposed GA were compared to those obtained by the three different metaheuristics, namely Variable Neighborhood Search (VNS) [26], hybrid scatter search, and path relinking (SS + PR) [24], and neural network (NN) [27]. In terms of the solution quality (average gap), the proposed GA is better than the NN and could compete with the other algorithms. For the problem SJC2, the proposed GA reached the optimum, in contrast to the other heuristics. Besides, in terms of the computation time, it seems that our algorithm performed much better than the others. However, it must also be noted that the computational experiments of these techniques were carried out on different computers that had different CPU speeds (VNS-3.2 GHz, SS + PR- Sun Blade 1000/750, NN-2.1 GHz, the proposed GA-2.0 GHz).

As seen in Table 2, although the VNS has a lower average gap percentage of 0.286% compared with our GA’s average gap percentage (1.145%), our proposed GA has a much lower CPU time in 5 out of 6 instances. VNS needed thousands of seconds to reach its best solution in four larger-size instances. This shows the computational time performance of our proposed GA. Similarly, SS + PR Algorithm needed a much longer CPU time than our GA to find its best solution in 4 out of 6 instances. Besides, the NN algorithm had a higher gap percentage, but its CPU time was shorter than that of our GA in most of the cases. So, in summary, our proposed GA has a much better computational time performance compared to the VNS and SS + PR and can compete with the NN in terms of computational time performance. So, our GA could decrease the computational time requirements for several CPMP data instances.

Furthermore, the new data set of Stefanello et al. [11] was used to test the performance of the proposed GA for larger-size problem instances. Fifteen problems with up to 200 medians and 724 demand points are considered from this data set. The problems were solved by using the proposed GA, and the results are reported in Table 3. Besides, the problems were run in the CPLEX solver, but no solution could be found for any of the instances. In Table 3, the best-known solutions, the solutions of Mat-heuristic (Matheu) presented by Stefanello et al. [11], and our results are compared. According to the results, the average gap between the solutions of the GA and the best-known solutions is 1.824%. As seen in Table 3, the proposed GA improved the gap percentage in three instances (P1, P6, P11). Furthermore, the best-known solution mentioned in [11] was improved for the largest instance (P15), and hence a negative gap was achieved. Better results are obtained in five problems compared with those of the Mat-heuristic of Stefanello et al. [11]. So, our GA has the potential to enhance the solutions for comparatively large instances. However, in terms of CPU times, the performance of the Mat-heuristic is better than that of the proposed GA. It should also be noted that the proposed GA was run by a

CPU of 2.0 GHz, and the Mat-heuristic was run by a CPU of 2.8 GHz.

Computational results indicated that if the problem size increases, getting optimum solutions by the mathematical model of the problem is not possible, but good solutions could be found by the proposed GA with reasonable CPU times. In addition, it is observed that if the N*P value increases, it gets harder to solve the problem with similar characteristics (such as demand/capacity ratio and distances). This has also affected the determination of the algorithm parameters (see Table A2 in the Appendix). These results show that effective metaheuristics are still needed for large-size problems that cannot be solved to optimality and achieve good solutions.

In order to show the impact of the Initial Solution Algorithm that improves the overall performance of the GA, we conducted a study such that we have run the GA without the Initial Solution Algorithm for the instances SJC1, SJC2, SJC3a, SJC3b, SJC4a, SJC4b and found some solutions, and presented them in Table 4. We compared them with those solutions that were already found. Both of them are shown in Table 4. Hence, we were able to reveal the effect of the Initial Solution Algorithm. According to the results in Table 4, it is clear that without the Initial Solution Algorithm, randomly generated solutions are found that are worse than all of those solutions found by applying the Initial Solution Algorithm, in terms of the objective (fitness) function.

To show the impact of the parameter tuning on the proposed GA, the results both before and after parameter tuning are shown in Table A3, in the Appendix. In some cases, parameter tuning has enabled the same results to be achieved faster (SJC1), and in some cases, it has enabled the solution to be improved in terms of the objective function by using more CPU time (SJC2, SJC3a, SJC3b, SJC4a). However, in one instance, the solution could not be improved, but a very close result was obtained after parameter tuning in a slightly shorter computational time (SJC4b). So, this shows that the parameter tuning process may improve the solution quality of the proposed GA, in terms of the objective function and the CPU time. However, the magnitude of improvement may vary among different instances.

Specifically, we would like to explain how statistical analysis techniques were applied for parameter tuning. MANOVA was applied to analyze the effects of population size, mutation probability, and maximum iteration number parameters on objective function and CPU time. Three values were taken for each parameter, which were found to give the best results after sufficient trials. Therefore, a total of 135 runs were made with 3^3 Full Factorial Design. According to the results, whether the effect of each parameter on the dependent variables was statistically significant or not was determined by MANOVA. Afterward, Post-hoc tests were performed to determine whether there was a significant difference between parameter levels. Thus, the pairwise comparative significance of the parameters was tested. According to the results obtained, parameter optimization was made by combining the parameter values that give the best objective function value and the shortest CPU time.

In order to explain the MANOVA and Post-hoc tests, the first problem in the real dataset (SCJ1) is discussed. For this problem, as seen in Table A4, values of 20, 40, and 60 for population size, 0.10, 0.30, and 0.50 for mutation probability, and values of 200, 250, and 300 for maximum iteration number were considered. According to the MANOVA Multivariate Test results in Table A5, the effect of all three parameters is statistically significant.

Whether the parameters are significant on the dependent variable is determined by the Tests of Between-Subjects Effects results in Table A6. Accordingly, it is clear that the number of iterations is significant for both objective function and CPU time. On the other hand, population size and mutation probability are not significant for the objective function, but they are significant for the CPU time. According to this result, it can be deduced that the Initial Solution Algorithm and the crossover operator provide sufficient diversity and the effect of population size and mutation probability for this problem is limited.

In the second step, pairwise comparisons were made with the Post-hoc test to determine which parameter groups differed. According to the Tukey HSD results given in

Table 4 Impact of the initial solution procedure

GA objective function value			
Problem	With initial solution procedure	Without initial solution procedure	Difference
SJC1	1779	17755	176
SJC2	33251	33438	187
SJC3a	45638	45834	196
SJC3b	40853	41074	221
SJC4a	62642	62942	300
SJC4b	53180	53368	188

Table A7, while there is no significant difference in terms of the objective function for population size (p_size) values, there is a significant difference between 20–60 and 40–60 in terms of CPU time. According to the Tukey HSD results given in Table A8, for the mutation probability (mp), there is a significant difference between the 0.1–0.3 and 0.1–0.5 values in terms of the objective function, while there is no significant difference between the parameter values in terms of CPU time.

Lastly, according to the Tukey HSD results given in Table A9, there was a significant difference between 200–300 and 250–300 values in terms of both objective function and CPU time for the number of iterations (max_iter). As a result, it is clear that the key factor in terms of CPU time for this problem is the population size and the number of iterations. In this way, statistical analyses were carried out for each problem, and optimum parameter values were determined.

5 Conclusion

In this study, an improved GA is developed for the capacitated p-median problem that employs an Initial Solution Algorithm. Hence, good initial solutions are obtained, and the computational time of the GA is reduced considerably. This is a significant contribution of this study to the literature.

Moreover, a 3^3 Full Factorial Design is performed where three levels are selected for the factors of the probability of mutation, the population size, and the number of iterations, and parameter tuning is performed to reach a better performance. The objective values and the CPU times are considered as response variables. For each parameter level, the proposed GA was run ten times. By using the GA solution results, MANOVA and Post-Hoc Tests are performed to identify whether the performance difference between selected parameter levels is statistically significant for each problem. Hence, significant parameter levels are determined and input into the algorithm. The proposed GA is solved for the significant parameter levels of the real data set presented by Lorena and Senne [10] and the new data set presented by Stefanello et al. [11]. The parameter tuning based on statistical analysis for the GA is a significant contribution to this study.

Besides, the results of the proposed GA are compared to those of the three metaheuristics for the real data set of Lorena and Senne [10]. For one of the instances, our algorithm reached the best solution in contrast to the others. For the other instances, our GA could reach very close results to those of other algorithms but in considerably less computational time. As a result, the proposed GA can solve medium and large-size cases of the capacitated p-median problems very close to the best-known solution in reasonable computational times that show its utility.

In future studies, the proposed GA may be improved by hybridizing it with other heuristics to obtain better results. Besides, more up-to-date applications of the capacitated p-median problem, such as electric vehicle charging station location, and renewable energy generation facility location problems can be studied in future.

Appendix A–Results of parameter tuning

See Tables 5, 6, 7, 8, 9, 10, 11, 12 and 13.

Table 5 Fine-tuned parameter values for the test problems according to the DOE

Problem	Parameters			
	N*P	*p*_size	mp	max_iter
SJC1	1000	20	0.1	300
SJC2	3000	40	0.3	300
SJC3a	7500	30	0.5	250
SJC3b	9000	40	0.5	300
SJC4a	12.060	40	0.3	300
SJC4b	16.080	40	0.5	300
P1	1590	30	0.3	200
P2	4770	30	0.3	250
P3	12.720	40	0.5	300
P4	22.260	40	0.5	300
P5	31.800	40	0.5	300
P6	2675	40	0.3	300
P7	13.375	40	0.5	300
P8	26.750	60	0.5	400
P9	53.500	60	0.8	500
P10	80.250	80	0.8	500
P11	7240	40	0.5	300
P12	21.720	60	0.5	400
P13	54.300	80	0.8	500
P14	90.500	80	0.8	500
P15	144.800	80	0.8	500

Table 6 Parameters for different N*P values

N*P	*p*_size (max)	mp (max)	max_iter
< 20.000	40	0.5	300
< 50.000	60	0.5	400
Others	80	0.8	500

Table 7 Impact of the parameter tuning

	GA (before parameter tuning)					GA (after parameter tuning)				
Problem	Objective	CPU time (s)	*p*_size	mp	max_iter	Objective	CPU time (s)	p_size	mp	max_iter
SJC1	17579	44	60	0.8	300	17579	31	20	0.1	300
SJC2	33251	130	60	0.9	250	33187	105	40	0.3	300
SJC3a	46138	177	40	0.8	200	45638	168	30	0.5	250
SJC3b	41107	426	60	0.8	250	40853	221	40	0.5	300
SJC4a	62898	778	60	0.8	300	62642	642	40	0.3	300
SJC4b	53176	1023	60	0.9	350	53176	921	40	0.5	300

Table 8 Analyzed parameter values for the real data problem SJC1

Between-subjects factors		
		N
Population size (p_size)	20	45
	40	45
	60	45
Mutation Probability (mp)	0.10	45
	0.30	45
	0.50	45
Max Number of iterations (max_iter)	200	45
	250	45
	300	45

Table 9 MANOVA multivariate test results

Multivariate tests						
Effect	Value		*F*	Hypothesis df	Error df	Sig
Intercept	Pillai's Trace	1,000	185,589,832	2,000	112,000	0,000
	Wilks' Lambda	,000	185,589,832	2,000	112,000	0,000
	Hotelling's Trace	3314,104	185,589,832	2,000	112,000	,000
	Roy's Largest Root	3314,104	185,589,832	2,000	112,000	,000
*p*_size	Pillai's Trace	,813	38,676	4,000	226,000	,000
	Wilks' Lambda	,193	71,352	4,000	224,000	,000
	Hotelling's Trace	4,140	114,893	4,000	222,000	,000
	Roy's Largest Root	4,133	233,496	2,000	113,000	,000
mp	Pillai's Trace	,146	4,437	4,000	226,000	,002
	Wilks' Lambda	,855	4,554	4,000	224,000	,001
	Hotelling's Trace	,168	4,669	4,000	222,000	,001
	Roy's Largest Root	,162	9,146	2,000	113,000	,000
max_iter	Pillai's Trace	,139	4,219	4,000	226,000	,003
	Wilks' Lambda	,862	4,305	4,000	224,000	,002
	Hotelling's Trace	,158	4,388	4,000	222,000	,002
	Roy's Largest Root	,148	8,349	2,000	113,000	,000

Table 10 MANOVA tests of between-subjects effects

Tests of between-subjects effects						
Source	Dependen*t* variable	Type III sum of squares	df	Mean square	*F*	Sig.
Corrected model	Objective	9,070,489,871[a]	21	431,928,089	8,092	,000
	CPU_time	33,791,259[b]	21	1609,108	50,226	,000
Intercept	Objective	19,066,944,736,186	1	19,066,944,736,186	357,202,082	,000
	CPU_time	43,770,891	1	43,770,891	1366,257	,000
p_size	Objective	126,812,995	2	63,406,497	1,188	,309
	CPU_time	14,292,625	2	7146,313	223,064	,000
mp	Objective	944,420,210	2	472,210,105	8,846	,000
	CPU_time	23,490	2	11,745	,367	,694
max_iter	Objective	770,295,299	2	385,147,650	7,215	,001
	CPU_time	172,583	2	86,292	2,693	,002

[a]. R Squared = ,601 (Adjusted R Squared = ,526)

[b]. R Squared = ,903 (Adjusted R Squared = ,885)

Table 11 Result of Post-hoc test for the population size (*p*_size)

Multiple comparisons								
Dependent variable		(I) *p*_zsize	(J) *p*_size	Mean difference (I–J)	Std. Error	Sig	95% Confidence interval	
							Lower bound	Upper bound
Objective	Tukey HSD	20	40	19,47	48,707	,916	− 96,21	135,15
			60	− 76,53	48,707	,262	− 192,21	39,15
		40	20	− 19,47	48,707	,916	− 135,15	96,21
			60	− 96,00	48,707	,124	− 211,68	19,68
		60	20	76,53	48,707	,262	− 39,15	192,21
			40	96,00	48,707	,124	− 19,68	211,68
CPU_time	Tukey HSD	20	40	− 1,4867	1,19,326	,429	− 4,3207	1,3473
			60	− 33,2267*	1,19,326	,000	− 36,0607	− 30,3927
		40	20	1,4867	1,19,326	0,429	− 1,3473	4,3207
			60	− 31,7400*	1,19,326	,000	− 34,5740	− 28,9060
		60	20	33,2267*	1,19,326	,000	30,3927	36,0607
			40	31,7400*	1,19,326	,000	28,9060	34,5740

*Significant

Table 12 Result of post-hoc test for the mutation probability (mp)

Multiple comparisons

Dependent variable		(I) m*p*	(J) m*p*	Mean difference (I–J)	Std, Error	Sig	95% Confidence interval	
							Lower bound	Upper bound
Objective	Tukey HSD	,10	0,30	406,47*	48,707	0,000	290,79	522,15
			0,50	406,33*	48,707	0,000	290,65	522,01
		,30	,10	− 406,47*	48,707	,000	− 522,15	− 290,79
			,50	− ,13	48,707	1,000	− 115,81	115,55
		,50	,10	− 406,33*	48,707	,000	− 522,01	− 290,65
			,30	0,13	48,707	1,000	− 115,55	115,81
CPU_time	Tukey HSD	,10	,30	4,4222*	1,19,326	,001	1,5882	7,2562
			,50	− 9,1356*	1,19,326	,000	− 11,9695	− 6,3016
		,30	,10	− 4,4222*	1,19,326	,001	− 7,2562	− 1,5882
			,50	− 13,5578*	1,19,326	,000	− 16,3918	− 10,7238
		,50	,10	9,1356*	1,19,326	,000	6,3016	11,9695
			,30	13,5578*	1,19,326	,000	10,7238	16,3918

*Significant

Table 13 Result of post-hoc test for the iteration number (max_iter)

Multiple comparisons

Dependent variable		(I) max_iter	(J) max_iter	Mean difference (I–J)	Std, Error	Sig	95% Confidence interval	
							Lower bound	Upper bound
Objective	Tukey HSD	200	250	− 20,29	48,707	,909	− 135,97	95,39
			300	240,69*	48,707	,000	125,01	356,37
		250	200	20,29	48,707	,909	− 95,39	135,97
			300	260,98*	48,707	,000	145,30	376,66
		300	200	− 240,69*	48,707	,000	− 356,37	− 125,01
			250	− 260,98*	48,707	,000	− 376,66	− 145,30
CPU_time	Tukey HSD	200	250	− 2,0400	1,19,326	,206	− 4,8740	,7940
			300	− 15,3667*	1,19,326	,000	− 18,2007	− 12,5327
		250	200	2,0400	1,19,326	,206	− ,7940	4,8740
			300	− 13,3267*	1,19,326	,000	− 16,1607	− 10,4927
		300	200	15,3667*	1,19,326	,000	12,5327	18,2007
			250	13,3267*	1,19,326	,000	10,4927	16,1607

*Significant

Acknowledgements This study has been financially supported by the Turkish National Science Foundation (TUBITAK) with project number 215M143.

Data availability The authors acknowledge that the data sets used in this study were obtained from [58] and [59]. The addresses of these websites are presented in the References.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Cruz C, Pelta D (2009) Soft computing and cooperative strategies for optimization. Appl Soft Comput 9(1):30–38
2. Reese J (2006) Solution methods for the p-median problem: an annotated bibliography. Networks 48(3):125–142
3. Djenić A, Radojičić N, Marić M, Mladenović M (2016) Parallel VNS for bus terminal location problem. Appl Soft Comput 42:448–458

4. Zheng YJ, Chen SY, Ling HF (2015) Evolutionary optimization for disaster relief operations: a survey. Appl Soft Comput 27:553–566
5. Satoglu SI, Durmusoglu MB, Dogan I (2006) Evaluation of the conversion from central storage to decentralized storages in cellular manufacturing environments using activity-based costing. Int J Prod Econ 103(2):616–632
6. Kariv O, Hakimi SL (1979) An algorithmic approach to network location problems. II: The p-medians. SIAM J Appl Math 37(3):539–560
7. Mladenović N, Brimberg J, Hansen P, Moreno-Pérez JA (2007) The p-median problem: a survey of metaheuristic approaches. Eur J Oper Res 179(3):927–939
8. Osaba E, Carballedo R, Diaz F, Onieva E, Lopez P, Perallos A (2014a) On the influence of using initialization functions on genetic algorithms solving combinatorial optimization problems: a first study on the TSP. In: IEEE conference on evolving and adaptive intelligent systems (EAIS) (pp 1–6). IEEE
9. Veček N, Mernik M, Filipič B, Črepinšek M (2016) Parameter tuning with chess rating system (CRS-Tuning) for meta-heuristic algorithms. Inf Sci 372:446–469
10. Lorena LAN, Senne ELF (2003) Local search heuristics for capacitated p-median problems. Netw Spat Econ 3(4):407–419
11. Stefanello F, de Araújo OC, Müller FM (2015) Matheuristics for the capacitated p-median problem. Int Trans Oper Res 22(1):149–167
12. Jamshidi M (2009) Median location problem. In: Hekmatfar M (ed) Zanjirani Farahani, Reza. Physica-Verlag, Facility Location, pp 177–191
13. Baldacci R, Caserta M, Traversi E, Wolfler Calvo R (2022) Robustness of solutions to the capacitated facility location problem with uncertain demand. Optim Lett 16:2711–2727
14. Ryu J, Park S (2022) A branch-and-price algorithm for the robust single-source capacitated facility location problem under demand uncertainty. EURO J Trans Logistics 11:100069
15. Lorena LAN, Senne ELF (2004) A column generation approach to capacitated p-median problems. Comput Oper Res 31(6):863–876
16. 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
17. Ceselli A (2003) Two exact algorithms for the capacitated p-median problem. Q J Belg Fr Ital Oper Res Soc 1(4):319–340
18. Ceselli A, Righini G (2005) A branch-and-price algorithm for the capacitated p-median problem. Networks 45:125–142
19. Klose A, Görtz S (2007) A branch-and-price algorithm for the capacitated facility location problem. Eur J Oper Res 179(3):1109–1125
20. 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 Modell Algorithms 7(1):43–58
21. Avella P, Boccia M, Mattia S (2013) A branch-and-cut algorithm for the single-source capacitated facility location problem. In: Advanced logistics and transport (ICALT), 2013 international conference on (pp 181–186). IEEE
22. Maniezzo V, Mingozzi A, Baldacci R (1998) A bionomic approach to the capacitated p-medianproblem. J Heuristics 4(3):263–280
23. Ahmadi S, Osman IH (2005) Greedy random adaptive memory programming search for the capacitated clustering problem. Eur J Oper Res 162(1):30–44
24. Diaz JA, Fernandez E (2006) Hybrid scatter search and path relinking for the capacitated p-median problem. Eur J Oper Res 169(2):570–585
25. Scheuerer S, Wendolsky R (2006) A scatter search heuristic for the capacitated clustering problem. Eur J Oper Res 169(2):533–547
26. Fleszar K, Hindi KS (2008) An effective VNS for the capacitated p-median problem. Eur J Oper Res 191(3):612–622
27. Shamsipoor H, Sandidzadeh MA, Yaghini M (2012) Solving capacitated p-median problem by a new structure of neural network. Int J Ind EngTheory Appl Pract 19(8):305–319
28. Hong JD, Jeong KY, Xie Y (2015) A multi-objective approach to planning in emergency logistics network design. Int J Indus Eng Eng Theory Appl Practice 22(4):412–425
29. Gnägi M, Baumann P (2021) A matheuristic for large-scale capacitated clustering. Comput Oper Res 132:105304
30. Osman IH, Christofides N (1994) Capacitated clustering problems by hybrid simulated annealing and tabu search. Int Trans Oper Res 1(3):317–336
31. Chaves AA, de Assis Correa F, Lorena LAN (2007) Clustering search heuristic for the capacitated p-median problem. In: Emilio CR, Manuel J, Abraham A (eds) Innovations in hybrid intelligent systems, Corchado. Springer, Berlin Heidelberg, pp 136–143
32. Landa-Torres I, Del Ser J, Salcedo-Sanz S, Gil-Lopez S, Portilla-Figueras JA, 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
33. Yaghini M, Karimi M, Rahbar M (2013) A hybrid metaheuristic approach for the capacitated p-median problem. Appl Soft Comput 13(9):3922–3930
34. Alp O, Erkut E, Drezner Z (2003) An efficient genetic algorithm for the p-median problem. Ann Oper Res 122(1–4):21–42
35. Bozkaya B, Zhang J, Erkut E (2002) An efficient genetic algorithm for the p-median problem. In: Drezner Z, Hamacher HW (eds) Facility location: applications and theory. Springer Verlag, Berlin, New York, pp 179–205
36. Fathali J (2006) A genetic algorithm for the p-median problem with pos/neg weights. Appl Math Comput 183(2):1071–1083
37. Oksuz MK, Satoglu SI, Kayakutlu G, Buyukozkan K (2016) A genetic algorithm for p-median facility location problem. In: Global joint conference on industrial engineering and its application areas (GCJIE 2016), Istanbul, Turkey, July 14–15
38. Correa ES, Steiner MTA, Freitas AA, Carnieri C (2004) A genetic algorithm for solving a capacitated p-median problem. Numer Algorithms 35(2–4):373–388
39. Ghoseiri K, Ghannadpour SF (2007) Solving a capacitated p-median problem using genetic algorithm. In: IEEE international conference on industrial engineering and engineering management, (pp 885–889). IEEE
40. Yang K, Wang R, He H, Yang X, Zhang G (2021) Multi-supply multi-capacitated p-median location optimization via a hybrid bi-level intelligent algorithm. Comput Ind Eng 160:107584
41. Taghavi A, Ghanbari R, Ghorbani-Moghadam K, Davoodi A, Emrouznejad A (2022) A genetic algorithm for solving bus terminal location problem using data envelopment analysis with multi-objective programming. Ann Oper Res 309:259–276
42. Liu D (2014) Network site optimization of reverse logistics for E-commerce based on genetic algorithm. Neural Comput Appl 25(1):67–71
43. Guo K (2020) Research on location selection model of distribution network with constrained line constraints based on genetic algorithm. Neural Comput Appl 32(6):1679–1689
44. Liu J, Xiao Y, Wang D, Pang Y (2019) Optimization of site selection for construction and demolition waste recycling plant using genetic algorithm. Neural Comput Appl 31(1):233–245
45. Osaba E, Carballedo R, Diaz F, Onieva E, De La Iglesia I, Perallos A (2014) Crossover versus mutation: a comparative analysis of the evolutionary strategy of genetic algorithms applied to combinatorial optimization problems. Sci World J 2014:1–22

46. Paul PV, Moganarangan N, Kumar SS, Raju R, Vengattaraman T, Dhavachelvan P (2015) Performance analyses overpopulation seeding techniques of the permutation-coded genetic algorithm: an empirical study based on traveling salesman problems. Appl Soft Comput 32:383–402
47. Ravichandran D, Praveenkumar P, Rayappan JBB, Amirtharajan R (2016) Chaos-based crossover and mutation for securing DICOM image. Comput Biol Med 72:170–184
48. Doerr B, Le HP, Makhmara R, Nguyen, TD (2017) Fast genetic algorithms. In: Proceedings of the genetic and evolutionary computation conference (pp 777–784). ACM
49. Salcedo-Sanz S, Portilla-Figueras JA, Ortiz-García EG, Pérez-Bellido AM, Thraves C, Fernández-Anta A, Yao X (2008) Optimal switch location in mobile communication networks using hybrid genetic algorithms. Appl Soft Comput 8(4):1486–1497
50. Katoch S, Chauhan SS, Kumar V (2021) A review on genetic algorithm: past, present, and future. Multimed Tools Appl 80(5):8091–8126
51. Corus D, Oliveto PS (2020) On the benefits of populations for the exploitation speed of standard steady-state genetic algorithms. Algorithmica 82:3676–3706
52. Reeves C (2003) Genetic algorithms. Springer, USA
53. Mulvey JM, Beck MP (1984) Solving capacitated clustering problems. Eur J Oper Res 18(3):339–348
54. Adenso-Diaz B, Laguna M (2006) Fine-tuning of algorithms using fractional experimental designs and local search. Oper Res 54(1):99–114
55. Buyukozkan K, Kucukkoc I, Satoglu SI, Zhang DZ (2016) Lexicographic bottleneck mixed-model assembly line balancing problem: artificial bee colony and tabu search approaches with optimized parameters. Expert Syst Appl 50:151–166
56. Duncan DB (1955) Multiple ranges and multiple F tests. Biometrics 11:1–42
57. Tukey J (1949) Comparing individual means in the analysis of variance. Biometrics 5(2):99–114
58. Url-1. http://www.lac.inpe.br/~lorena/instancias.html. Last access date: June 1st, 2021
59. Url-2. http://www-usr.inf.ufsm.br/~stefanello/instances/CPMP/group2/. Last access date: July 12st, 2021