Journal of Engg. Research Vol. 6 No. (4) December 2018 pp *1-16*

Multiple-server facility location problem with stochastic demands along the network edges

Mahmoud Golabi*, Gokhan Izbirak** and Jamal Arkat***

** Industrial Engineering Department, Girne American University, Girne, via Mersin 10, Turkey*
*** Industrial Engineering Department, Eastern Mediterranean University, Famagusta, via Mersin 10, Turkey*
**** Industrial Engineering Department, University of Kurdistan, Sanandaj, Iran*
Corresponding author's email: mahmoudgolabi@gau.edu.tr

ABSTRACT

This paper investigates the network location problem for multiple-server facilities,which are subject to congestion. A number of facilities are to be selected among several candidate locations in order to satisfy customers' demands. For each network edge, the corresponding customers are uniformly distributed along the edge, and their demands are generated according to the Poisson process. Furthermore, the number of servers in each established facility is considered as a decision variable, and the service time for each server follows an exponential distribution. Using queuing system analysis, a mathematical model is developed to minimize the customers' aggregate expected traveling times and the aggregate expected waiting times. Since network location problems are NP-hard, three metaheuristic algorithms including genetic algorithm, memetic algorithm, and simulated annealing are then investigated and developed to solve the proposed problem. The resultsof implementing the algorithms on some test problems demonstrate that the proposed memetic algorithm outperforms theother two algorithms in terms of objective values.

Keywords: Logistics, facility location, distributed demand, queuing theory, metaheuristics.

INTRODUCTION

Since the seventeenth century, facility location problem (FLP) has been studied by a myriad number of researchers. Nowadays, it is one of the most prominent branches of operation research, which is applied in different fields such as determining the location of warehouses, hazardous materials sites, automated teller machines (ATMs), coastal search, and rescue stations. Also, the application of FLP in the emergency service location has become rampant recently(Moeini et al., 2015). As reviewed byKlose & Drexl (2005),Hale & Moberg (2003), andBoloori Arabani & Farahani (2012), FLP could be subdivided into various categories.

FLP can be deterministic or stochastic. In stochastic problems, as opposed to deterministic ones, some parameters like demand or cost are uncertain(Bieniek 2015; Rahmaniani et al. 2013);for more details, refer toSnyder (2006). For the first time, Larson (1974 and 1975)challenged the deterministic essence of real-life FLPs by presenting the idea of congestion. In congested facility location problem (CFLP), customers need to wait in a queue to receive the service. Boffey et al. (2007) scrutinized different versions of CFLPs.

Based on the objective function, location problems are classified into three categories: center, covering, and median problems.

The objective function of center problems minimizes the maximum distance or service times (Aboolian et al., 2009). These problems are generally applied in emergency location problems such as locating ambulance stations and fire stations.Since center problems are NP-hard(Garey & Johnson, 1979; Kariv & Hakimi, 1979), many metaheuristic algorithms have been applied in order to solve these problems.

The objective function of covering models is to maximize the number of covered clients. Using the M/G/1 queuing systems,Hamaguchi & Nakadeh (2010) formulated a model in order to maximize the total number of covered demands. By considering the budget constraint, Hu et al. (2013)presented a model according to M/M/C queuing system with the objective function of maximizing the covered demands.Farahani et al. (2012)compiled different versions of covering problems.

Median problems refer to those problems wherein the objective function is to minimize the total traveling or waiting times. The single-server (M/M/1) model proposed by Wang et al. (2002) and the multiple-server (M/M/C) model presented by Berman & Drezner(2007)are some of the examples of median CFLPs. The application of bi-objective models is a sought approach in median CFLPs(Marianov & Serra, 2011).Some of these bi-objective models consider customer and server aspects both simultaneously(Pasandideh & Niaki, 2012). Some studies have expanded these bi-objective models by adding one extra objective function to account for pecuniary aspects(Pasandideh et al., 2013). Kariv & Hakimi(1979)proved that the median problem is NP-hard on a general graph. Since the exact methods may not be able to solve the median problems, the application of metaheuristic algorithmsbecomes inevitable. Differentmetaheuristic algorithms such as genetic algorithm(Alp et al., 2003; Pasandideh et al., 2013), simulated annealing (Berman & Drezner, 2007), Tabu search(Brimberg & Drezner, 2013), and variable neighborhood search(Rahmaniani & Ghaderi, 2015)have been developed for solving the median problems. Mladenović et al. (2007) reviewed many metaheuristics and exact methods to solve this problem.

Flow capturing location problems (FCLPs) are another type of problems thatare similar to FLPs. In contrast withstudies mentioned before in which it is considered that the customers are located at the network nodes, a network of paths each directed towards a specific flow is considered in FCLPs(Hodgson, 1990). Flow refueling location problems (FRLP) could be considered as one of the branches of FCLP. Kuby & Lim (2007)andKuby et al. (2009) applied FRLP models in order to find the best location of fuel stations.

Inspired by FCLPs, Arkat & Jafari (2016)proposed a more realistic p-median facility location problem according to M/M/1 queuing system, in which customers are uniformly distributed along the network edges. Since, in the majority of real-life problems, more than one server is required to satisfy customer demands at each facility, this study expands the model proposed by Arkat & Jafari (2016)by incorporating the M/M/C queuing framework in order to reflect a more realistic image of the problem. This expansion transforms the developed mixed integer linear model to a mixed integer nonlinear model, which intensely heightens the complexity of the problem and solution methods. After determining the location of open facilities, a predefined number of servers are distributed among the open facilities in order to satisfy customer demands. The objective function minimizes the aggregate expected transportation time between the customers and the open facilities, and the aggregate waiting times for the customers in the system. Applications of such a model include the location of bank branches, post offices, and healthcare centers, where the number of staff or tellersat each location should be determined.

The paper is organized as follows. In Section 2, the problem is described, and a mathematical model is presented. Further, in order to check the model validation, a small problem is solved. In Section 3, the characteristics of three applied metaheuristic algorithms including GA, MA, and combined SA are described. In order to examine the applicability of solution algorithms, some numerical examples are generated in Section 4.In Section 5, the numerical results are reported, and the conclusion is drawn in Section 6.

PROBLEM DEFINITION AND ASSUMPTIONS

The problem studied in this paper is a facility location problem for immobile facilities where several servers are settled to serve the customers. In this paper, it is assumed that demands are generated along the network edges with pre-defined geographical positions. The demands are uniformly distributed along the edges, and the time interval between any two consecutive demands follows an exponential distribution with a known parameter. Customers who arrive at a busy server will wait in a queue with innings system. The service time of each server follows an exponential distribution with a defined parameter. In this problem, homological(1) demands have different distances to their closest open facility. Therefore, in the queue, based on traveling distance, the demands that are generated sooner may stand after the demandsthat are generated later. As a result, the time interval between demand generations follows a different distribution with the service time interval, and this difference makes the related model more complex. This problem is a discrete location problem, which means that there are a set of potential locations to set up facilities and from them, a certain number is selected to cover customers' demands and minimize the aggregate customers' expectedtraveling and waiting times. The total number of available servers, which are distributed among the open facilities, is known in advance. Due to the fact that the distance between any demand point and the end points ofthe corresponding edge follows the uniform distribution, customer's movement along each network edge could be assumed as$M/U/\infty$ queuing system. Since the output process of $M/G/\infty$ is Poisson (Gross et al., 2008),and these outputs enter the facilities in order to receive the service, the queuing system for each facility will be according to M/M/C. The remaining assumptions are summarized as follows:

- The customers' moving speed along all the network edges is identical.
- Potential locations for setting up the facilities are known.
- The servicing system of queue follows FIFO.
- In order to receive the service, each customer goes to the closest open facility.
- Based on the budget constraint, the number of open facilities is known in advance.
- A customer, who arrivesat a facility thatis too crowded, could not cancel receiving the service or skip the facility to go to another one.

(1) Customers who are settled on the same edge are called homological customers.

For more clarification, Fig. 1 presents the problem schematically.

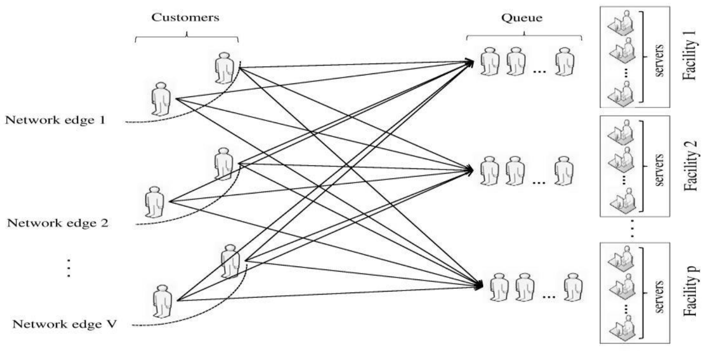


Fig. 1. facility location problem scheme.

Model formulation

In order to develop the mathematical model, the sets, parameters, and decision variables are defined as follows:

Sets:

- $G\ (V, A) : A$ network comprised of the set of nodes (V) and the set of edges (A)
- V: The set of network nodes ($v, v' \in V$)
- A : The set of network edges ($A \subseteq V \times V, (v, v') \in A$)
- J: The set of candidate locations for establishing facilities ($j, j' \in J$)

Parameters:

- $l_{vv'}$: The length of edge (v, v')
- $\lambda_{vv'}$: The rate of demand occurrence for edge (v, v')
- t_{vj}: The minimum distance between node v and facility j obtained from the Dijkstra algorithm

Decision variables:

- c_j: Number of servers at facility j
- w_j: The expected waiting time at facility j
- $y_j: \begin{cases} 1 & if\ facility\ j\ is\ open \\ 0 & otherwise \end{cases}$
- $x_{vj}: \begin{cases} 1 & if\ the\ closest\ open\ facility\ to\ node\ v\ is\ facility\ j \\ 0 & otherwise \end{cases}$
- γ_j: The demand entrance rate at facility j
- π_{0_j}: The probability that opened facility j contains no customers (idle probability)
- $b_{vv'jj'}$: The distance between node v and decomposing point of edge (v, v'), if nodes v and v' are, respectively, assigned to open facilities j and j'

Scalars:

- p: Number of open facilities
- M: A large positive value whose lower bound is the biggest amount of t_{vj}
- M' : A large positive value whose lower bound is c_{max}
- c_{total} : The total number of available servers
- w_{max}: The maximum time that customer could wait in the system
- μ: Common service rate of each server
- ε: An infinitesimal positive number

As shown in Fig. 2, suppose that j and j' are the closest opened facilities to the nodes v and v', respectively $(x_{vj} = x_{v'j'} = 1)$. The customers who are located between node and decomposing point of the edge (v, v') are assigned to facility. The remaining customers of edge (v, v') are assigned to facility j'.

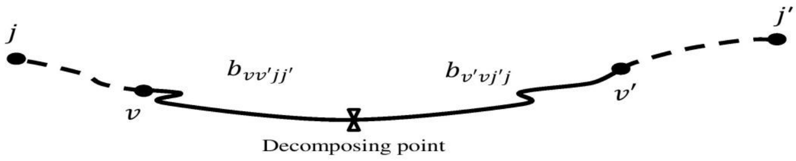


Fig. 2. A network edge.

According to the assumptions of Fig. 2, the edge (v, v') is decomposed as

$$t_{vj} + b_{vv'jj'} = t_{v'j'} + b_{v'vj'j} \tag{1}$$

$$\text{Since} b_{vv'jj'} + b_{v'vj'j} = l_{vv'}, \text{ so } b_{vv'jj'} = \frac{t_{v'j'} + l_{vv'} - t_{vj}}{2} \tag{2}$$

Lemma 1) If facilities j and j' are the closest opened facilities to the nodes and v', respectively, then $0 \leq b_{vv'jj'} \leq l_{vv'}$

Proof : Suppose that $_{vv'jj'} < 0$, so

$$t_{v'j'} + l_{v'v} - t_{vj} < 0 \qquad \rightarrow \qquad t_{vj} > (t_{v'j'} + l_{v'v} = t_{vj'}) \tag{3}$$

It means that, instead of facility j, now j' is the closest facility to the node and itis in contradiction with our assumptions. The same result could be obtained for the case in which $b_{vv'jj'} > l_{v'v}$.

Since customers' moving speed along all of the network edges is identical, instead of time needed to traverse the edges, the length of edges is used. The number of generated demands along each network edge is related to the length of the edge, sofor each edge, the aggregate expected customers' traveling time for reaching one of the end points of the edges could be calculated as

$$T = \lambda \int_0^L \frac{x}{L} dx = \frac{\lambda}{L} \int_0^L x dx = \lambda \frac{L}{2} \tag{4}$$

By applying these assumptions, the proposed mathematical model is as follows:

$$Min\sum_{j\in J}\sum_{j'\in J}\sum_{(v,v')\in V}\frac{\lambda_{vv'}b_{vv'jj'}}{l_{vv'}}\left(t_{vj}+\frac{b_{vv'jj'}}{2}\right)x_{vj}x_{v'j'}+\sum_{j\in J}\gamma_j w_j \quad (5)$$

subject to

$$\sum_{j\in J} y_j = p \quad (6)$$

$$\sum_{j\in J} x_{vj} = 1 \qquad \forall\, v\in V \quad (7)$$

$$x_{vj} \le y_j \qquad \forall\, v\in V, \forall j\in J \quad (8)$$

$$\sum_{j'\in J} x_{vj'}t_{vj'} \le t_{vj} + M(1-y_j) \qquad \forall\, v\in V\ , \forall j\in J \quad (9)$$

$$2b_{vv'jj'} = (t_{v'j'} + l_{vv'} - t_{vj})x_{vj}x_{v'j'} \qquad \forall (v,v')\in A\ ,\ \forall j\in J\ ,\ \forall\, j'\in J \quad (10)$$

$$\gamma_j = \sum_{j'\in J}\sum_{(v,v')\in V}\frac{\lambda_{vv'}b_{vv'jj'}}{l_{vv'}}x_{vj}x_{v'j'} \qquad \forall j\in J \quad (11)$$

$$\sum_{j\in J} c_j = c_{total} \qquad \forall j\in J \quad (12)$$

$$c_j \le M'y_j \qquad \forall j\in J \quad (13)$$

$$c_j \ge \frac{\gamma_j}{\mu} + \varepsilon \qquad \forall j\in J \quad (14)$$

$$\pi_{0_j} = \left(\left(\sum_{n=0}^{c_j-1}\frac{\gamma_j^n}{\mu^n n!}\right) + \frac{\gamma_j^{c_j}}{\mu^{c_j}c_j!\left(1-\frac{\gamma_j}{\mu c_j}\right)}\right)^{-1} \qquad \forall j\in J \quad (15)$$

$$w_j = \frac{1}{\mu} + \left(\frac{\gamma_j^{c_j}}{\mu^{c_j}(c_j!)(c_j\mu)(1-\frac{\gamma_j}{\mu c_j})^2}\right)\pi_{0_j} \qquad \forall j\in J \quad (16)$$

$$w_j \leq w_{max} \qquad \forall j \in J \tag{17}$$

$$x_{vj}, y_j \in \{0,1\} \qquad \forall v \in V, \forall j \in J \tag{18}$$

$$\gamma_j \geq 0, \; c_j \geq 0, \quad b_{vv'jj'} \geq 0 \qquad \forall (v,v') \in A, \forall j \in J, \forall j' \in J \tag{19}$$

InEq. (5), the customers' aggregate expected traveling timesand waiting timesare minimized. Constraint (6) ensures that, among all candidates, plocationsare selected to establish the facilities. Constraint (7) guarantees that a unique facility is assigned to each node. Constraint (8) shows that no customer is assigned to closed facilities. Constraint (9) assures that each customer will be assigned to the closest open facility. Constraint (10) calculates the decomposing point for each network edge. Constraint (11) calculates the demand entrance rate for each open facility. Constraint (12) ensures that the total number of assigned servers is. Constraint (13) guarantees that servers are not assigned to closed facilities. Constraint (14) assures that the queuing system will achieve a steady state mode. According to the characteristics of M/M/C queuing systems(Gross et al., 2008), constraints (15) and (16) calculate the idle probabilities and expected waiting times at the open facilities.Constraint (17) considers an upper bound for customers' expected waiting time at each facility. Constraints (18) and (19) preserve the binary and nonnegative restrictions on decision variables.

An illustrative example

In order to examine the model, according to Fig. 3, a small network consisting of7 nodes, 11edges, and 4 candidate locations is presented. The length of each edge is written above it,and the numbers written in parentheses are related tothe correspondingrate of demand generation. It is assumed thatthe common service rate is 8 customers per hour, a totalof15 servers are available, maximum allowed waiting time is 25 minutes, and two facilities could be opened. This problem has been coded in general algebraic modeling system (GAMS) and solved by the Bonmin solver. According to the obtained results, 9 and 6 servers should be assigned to facilities established at nodes 3 and 4, respectively.

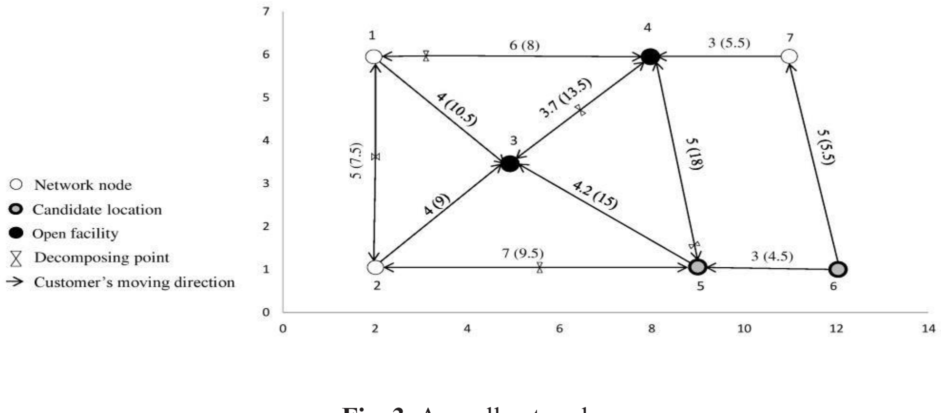


Fig. 3. A small network.

SOLUTION METHODS

As mentioned earlier, the median problems on a general graph are NP-hard problems. Further, a constrained non-linear mixed integer programming model, such as the proposed herein,is considered to be NP-hard (Pasandideh et al., 2013; Hajipour et al., 2014). Due to these reasons, finding the exact solution for the proposed model is hard (if not impossible). Therefore, in order to solve the problem, in addition to coding the model in GAMS for finding the best solution, three metaheuristic algorithms are developed. The developed algorithms are the genetic algorithm (GA), the memetic algorithm (MA), and the combined simulated annealing algorithm (SA). While describing the applied algorithms, it should be noted that, due to some technical restrictions, the proposed model could not be solved by GAMS in its basic form. Coding summation operator with a variable upper bound as used in constraint (15) is not allowed in GAMS. Also, the factorial function with variable inputs (constraints (15) and (16)) could not be coded in GAMS. In order to get rid of these restrictions, after defining a new binary variable (z_{kj}), the following constraint manipulation is used.

$$z_{kj}: \begin{cases} 1 \ \ if\ k\ servers\ are\ assigned\ to\ facility\ j \\ 0 \qquad otherwise \end{cases}$$

Constraints (12), (13), (14), (15), and (16) are replaced with the following constraints, respectively.

$$\sum_{j}\sum_{k=1}^{c_{total}} kz_{kj} = c_{total} \qquad \forall j \in J \tag{20}$$

$$\sum_{k=1}^{c_{total}} z_{kj} = y_j \qquad \forall j \in J \tag{21}$$

$$\sum_{k=1}^{c_{total}} kz_{kj} \geq \frac{\gamma_j}{\mu} + \varepsilon \qquad \forall j \in J \tag{22}$$

$$\pi_{0_j} = \left(\sum_{k=1}^{c_{total}} z_{kj}\left(\sum_{n=0}^{k-1}\frac{\gamma_j^n}{\mu^n n!} + \frac{\gamma_j^k}{\mu^k k!\left(1-\frac{\gamma_j}{\mu k}\right)}\right)\right)^{-1} \qquad \forall j \in J \tag{23}$$

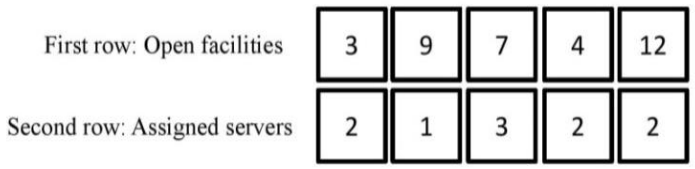
$$w_j = \frac{1}{\mu} + \left(\sum_{k=1}^{c_{total}} z_{kj}\frac{\gamma_j^k}{\mu^k(k!)(k\mu)(1-\frac{\gamma_j}{\mu k})^2}\right)\pi_{0_j} \qquad \forall j \in J \tag{24}$$

Genetic algorithm (GA)

Genetic algorithms, which are derived from observed processes in natural evolution, were first introduced by John Holland in the 1970s. GA is a search technique thatstarts with a population of random solutions(Khodaparasti et al., 2016). Each solution is called a chromosome. Through successive iterations, called generations, the chromosomes are evolved and the algorithm is converged to the best chromosome (see Eiben & Smith (2003)for more information). During each generation, the genetic operators such as selection, crossover, and mutation are implemented in order to generate new chromosomes.The main steps of GA applied in this paper are explained in the next subsections.

Step 1*(Initialization)*: In this step, the parameters of GA are initialized. The parameters applied here are population size (Npop), maximum number of iterations (MaxIt), crossover probability (Pc), and mutation probability (Pm).

Step 2 *(Representation)*: Encoding is one of the most important steps in designing GA algorithms to create an appropriate definition of solutions. In this paper, each solution (chromosome) is shown by a matrix with two rows. The length of each row is equal to the number of open facilities (p), which means that each row has exactly p genes. The allele of each gene in the first row represents the index of locations chosen for establishing the facilities. In the second row, the allele of each gene determines the number of servers assigned to the corresponding facility. Fig. 4 is an illustration of a chromosome for a problem with 5 facilities and 10 servers. According to the first row of Fig. 4, facilities are located in locations 3, 9, 7, 4, and 12. Furthermore, according to the second row, there are 2, 1, 3, 2, and 2 servers in the aforementioned facilities, respectively.



First row: Open facilities	3	9	7	4	12
Second row: Assigned servers	2	1	3	2	2

Fig.4. Chromosome encoding.

Step 3 *(Initial population)*: The initial population is randomly generated. In order to generate each random solution, the first row of the chromosome is randomly filled by non-repetitive indices of candidate locations. In order to fill the second row, firstly one server is assigned to each facility, and then the remaining servers are randomly distributed among the facilities.

Step 4 *(Fitness evaluation)*: The fitness value of each chromosome is computed by Eq. (5). Since the considered structure for chromosomes does not guarantee the satisfaction of constraints (14) and (17), some generated chromosomes could be infeasible. One of the most prominent ways to handle the infeasible solutions is to apply penalty functions (Yeniay 2005). In the case of infeasibility, the penalty function is added to the fitness value of the solution. If constraint (14) is violated, the applied penalty function is as follows(Hajipour et al. 2014):

$$p(x) = \alpha \times max\left\{\left(\frac{\Upsilon_j}{\mu c_j} - 1\right), 0\right\} \tag{25}$$

The penalty function, which is considered for violation of constraint (17), is defined as

$$p(x) = \alpha \times w_j \times \Upsilon_j \tag{26}$$

In above equations, α is a big positive number.

Step 5 *(Parent selection)*: The total number of parental chromosomes for carrying out the crossover operator is calculated by $(Pc \times Npop)$. The selection process among the parental chromosomes is based on the roulette wheel procedure. In this method, the parents with better fitness values have a greater chance of being selected. In other words, according to the fitness value, a cumulative probability, which shows the chance of each parent for being selected, is calculated (seeKumar (2012) for more information).

Step 6 *(The crossover operator)*: In this section according to one-point-cut crossover operator, two offspring chromosomes are reproduced by mating two parental chromosomes. At the beginning, a crossover point is selected randomly along the length of the mating chromosomes. This point breaks each parent chromosome into two segments. Up to the crossover point, the first segmentgenes of the first parent chromosome are copied to the first offspring. The remaining genes of the first offspring are taken from the second segment of the second parent chromosome. If the first row alleles of the second parent are present in the offspring chromosome, the first segment genes are copied. In a similar process, the second offspring is produced by exchanging the role of the first and second parents.Fig. 5 shows a graphical representation of the crossover operation.

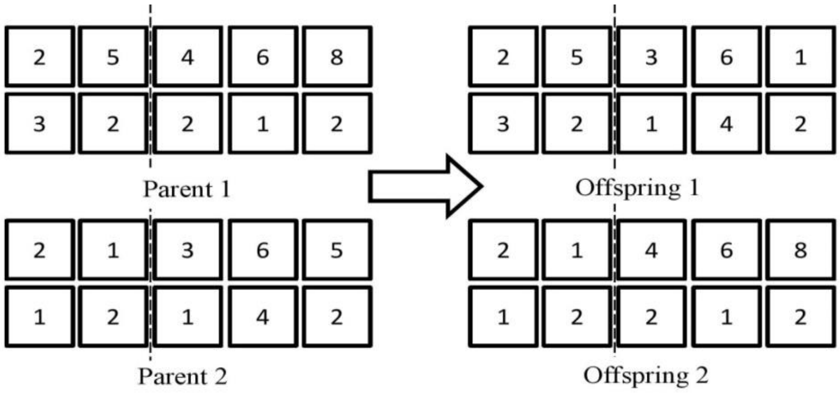


Fig. 5. An example of crossover operation.

Step 7 *(Repairing operator)*: Since, in the crossover operation, each offspring inherits the servers directly from its parents, occasionally constraint (12) can be violated. While constraint (12) is satisfied, the following process is repeated. If the total number of assigned servers (summation of the second-row alleles) is greater than the total number of available servers (, in each repetition, a second-row gene containing more than one server is randomly selected and its allele is decreased by one. Otherwise, in each repetition, the number of assigned servers of a randomly selected second-row gene is increased by one.

Step 8 *(Mutation operator)*: The mutation operator is applied to the second row of each chromosome. According to the mutation probability (Pm), two randomly selected second-row alleles are swapped with each other. Fig. 6 represents the mutation operator.

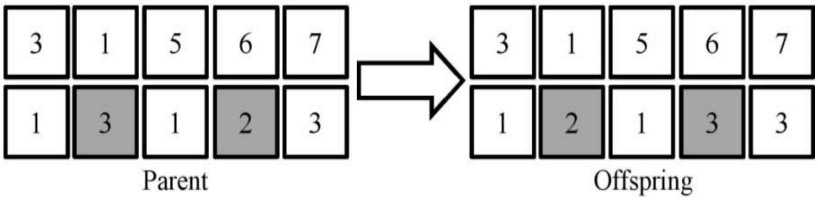


Fig. 6. An example of mutation operation.

Step 9 *(Replacement and stopping criteria)*: In each iteration, according to the steady state strategy (Lozano et al., 2008), the best offspring generated through crossover and mutation operations is compared with the worst individual of the current population. If the fitness value of the offspring is better, it replaces it. When the algorithm reaches a predetermined number of iterations, the GA is stopped.Algorithm 1 shows the pseudo code of the proposed genetic algorithm.

Memetic algorithm (MA)

Similar to GA, MA is also a population-based metaheuristic search method. MA combines the biological evolution of GA with the individual learning procedures in order to mimic the cultural evolution (Tavakkoli-Moghaddam et al., 2009). These individual learning procedures could be implemented by local search techniques (Moscato & Norman, 1992). Therefore, a genetic local search algorithm could be considered as an MA(Moré et al., 1981).

The MA proposed in this paper differs from the applied GA in the application of a local search technique. In order to improve the quality of the generated offspring, after applying the mutation operator to each generation, a local search method is implemented in the MA. In this method, through successive iterations, a predetermined number of neighborhood solutions (localit) are generated for each chromosome. At each iteration of the local search algorithm, the current solution is replaced with a generated neighborhood solution, which has a better fitness value. In order to generate neighborhood solutions, at first, a random integer number (R) in the $[1,\left\lceil\frac{p}{4}\right\rceil]$ interval is generated. Then, in the first row of the current solution, the alleles of R randomly selected genes are replaced with the candidate location indices, which are not present in this solution. The second-row alleles corresponding to exchanged genes are replaced with randomly generated integer numbers in the $[\left\lceil\frac{c_{total}}{p}\right\rceil,\left\lceil\frac{c_{total}}{0.5\times p}\right\rceil]$ interval. At the end, the generated neighborhood solution is repaired in a manner explained in step 7 of the developed GA. Fig. 7 shows the manner of generating neighborhood solutions.

```
0: Initialize the GA parameters (Npop, Pc, Pm, Maxit)
1: Generate the initial population
2: Calculate the fitness value of each individual
3: $S'$ ← worst individual
4: $\bar{S}$ ← best individual
5: iterations ← 0
6: while (iterations < Maxit) do
7:     use Rollete Wheel to select two parents (P1, P2)
8:     child1, child2 ← Crossover (P1, P2)
9:     Repair (child1, child2)
10:    child3 ← Mutate (child1)
11:    child4 ← Mutate (child2)
12:    Calculate the fitness value of each child
13:    $C^*$ ← best child
14:    if fitness ($C^*$) < fitness ($S'$)
15:        $C^*$ replaces $S'$
16:    end if
17:    order the new population and update $S'$ and $\bar{S}$
18:    iterations ← iterations + 1
19: end while
20: return $\bar{S}$
```

Algorithm 1. The pseudo code of the GA.

Algorithm 2 shows the pseudo code of local search procedure. In order to construct the MA, this procedure should be added between steps 11 and 12 of Algorithm 1.

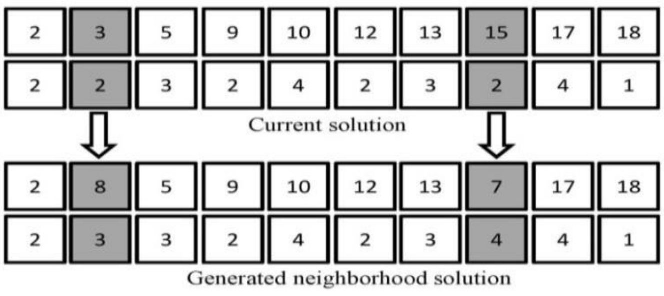


Fig. 7. An example of local search operation.

```
1: j ← 3
2: while (j < 5) do
3:      child(j+2) ← child(j)
4:      local iteration ← 0
5:      while (local iteration < Localit) do
6:          childlocal ← local Search (child(j+2))
7:          repair (childlocal)
8:          if fitness(childlocal) < fitness (child(j+2))
9:              child(j+2) ← childlocal
10:         end if
11:         local iteration ← local iteration + 1
12:     end while
13:     return child(j+2)
14:     j ← j + 1
15: end while
```

Algorithms 2. The pseudo code of local search procedure.

Simulated annealing (SA)

Simulated annealing is a metaheuristic method based on the local search techniques in order to approximate the global optimum solution. SA refrains from being trapped in local minima by accepting worse solutions according to a certain probability(Falah & Khorshid, 2014). Due to the good quality of the solutions found by SA, this algorithm is applied to solve complicated combinatorial optimization problems in a wide variety of areas. SA was independently introduced by Kirkpatrick et al. (1983)and Černý (1985).

In this paper, a combined SA algorithm with an inner layout algorithm (ILA) and an outer layout algorithm (OLA) is developed(Qin et al., 2015). OLA optimizes the location of open facilities, and ILA optimizes the number of assigned servers. The temperature (T) is set to be in the initial level (T_0) in the first step of the proposed SA. The algorithm starts with an initial solution (S). The representation of solution, generating the initial solution, and computing the fitness values are carried out according to steps 2, 3, and 4 of the developed GA,respectively. The global optimum solution ($\bar{S}$) is set to be the initial solution. At each temperature level, through N1 successive iterations, the OLA generates neighborhood solutions ($\bar{S}$) from the current solution (S). At each iteration of OLA, in order to generate neighborhood solutions ($\bar{S}$), a random integer number (R1) in $[1,\left\lceil\frac{p}{4}\right\rceil]$ interval is generated. Then, R1 number of randomly selected facilities in the first row of current solution matrix is replaced with the candidate location indices, which are not present in the current solution. If the objective value of S' is less than $\bar{S}$, S' replaces. In the next step, S' is compared with. Let Δ be the difference between the objective values of S' and, S, that is, $\Delta = \text{obj}(S') - obj(S)$. If $\Delta \leq 0$, S' replaces S; otherwise S' replaces S according to probability ($p = \exp(\frac{-\Delta}{T})$). At each iteration of OLA, ILA is repeated N2 times. At each repetition of ILA, S' is set to be the current solution and S″ is the generated neighborhood solution from the current solution. In order to generate S'', a random integer number (R2) in the [1,$\left\lceil\frac{p}{4}\right\rceil$] interval is generated. Then the number of servers assigned to R2 randomly selected facilities in the second row of current solution matrix is altered to random integer numbers in the [$\left\lceil\frac{c_{total}}{p}\right\rceil$,$\left\lceil\frac{c_{total}}{0.5\times p}\right\rceil$] interval. Since the number of servers has been changed, occasionally constraint (12) can be violated. Thus, as described instep 7 of the developed GA,the repairing operator is applied. The same replacing procedure described in OLA is applied for S' and S'' in the next step. After reaching N1, the temperature is reduced and this process is stopped when the final temperature (T_f) is reached. Algorithm 3 shows the pseudo code of the proposed SA.

```
0:  Initialize the SA parameters ($t_0, \alpha, t_f, N1, N2$)
1:  Generate the initial individual solution ($S$)
2:  Best individual solution ($\bar{S}$) ← $S$
3:  $T$ ← $t_0$
4:  while ($T < t_f$) do
5:      $C1$ ← 1
6:      while ($C1 < N1$) do
7:          $S'$ ← local Search $S$
8:          $C2$ ← 1
9:          while ($C2 < N2$) do
10:             $S''$ ← local Search $S'$
11:             repair ($S''$)
12:             if obj ($S''$)< obj ($S'$)
13:                 $S'$ ← $S''$
14:             else if rand (0,1) < $e^{-(\frac{obj(S'')-obj(S')}{T})}$
15:                 $S'$ ← $S''$
16:             end if
17:             $C2$ ← $C2 + 1$
18:         end while
19:         if obj ($S'$) < obj ($\bar{S}$)
20:             $\bar{S}$ ← $S'$
21:         end if
22:         if obj ($S'$) < obj ($S$)
23:             $S$ ← $S'$
24:         else if rand (0,1) < $e^{-(\frac{obj(S')-obj(S)}{T})}$
25:             $S$ ← $S'$
26:         end if
27:         $C1$ ← $C1 + 1$
28:     end while
29:     $T = t_0 \times \alpha$
30: end while
31: return ($\bar{S}$)
```

Algorithm 3. The pseudo code of SA.

Parameter tuning

Since the parameters influence the efficiency and effectiveness of metaheuristic algorithms, it is necessary to adjust them in advance to implement the algorithms. Different parameters used for proposed algorithms with their relative ranges are given in Table 1. In order to calibrate the parameters, the Taguchi method is utilized in this study. Taguchi uses orthogonal arrays in order to investigate a large number of controllable factors with a small number of experiments (Mousavi et al., 2013). This method finds the optimal level of controllable factors by minimizing the effect of noise. In order to evaluate the variation of the response, the signal to noise ratio (*S/N*) is calculated according to Eq. (27) in which denotes the response value and shows the number of orthogonal arrays.

$$S/_{N} = -10 \times \log(S(Y^2)/n) \tag{27}$$

Table 1. Parameter levels.

Algorithms	Algorithm parameters	Parameter range	Low(1)	Medium(2)	High(3)
GA	Npop (A)	100 - 300	100	200	300
	Pc (B)	0.7 - 0.9	0.7	0.8	0.9
	Pm (C)	0.3 - 0.5	0.3	0.4	0.5
	Maxit (D)	100 - 300	100	200	300
MA	Npop (A)	50 - 150	50	100	150
	Pc (B)	0.7 - 0.9	0.7	0.8	0.9
	Pm (C)	0.1 - 0.3	0.1	0.2	0.3
	Maxit (D)	50 - 150	50	100	150
	Localit (E)	10 - 30	10	20	30
SA	T_O (A)	50 - 70	50	60	70
	T_f (B)	0.05 - 1	0.05	0.1	1
	A (C)	0.85 – 0.95	0.85	0.9	0.95
	N1 (D)	10 - 30	10	20	30
	N2 (E)	10 - 30	10	20	30

In this study, the variation is modeled by applying the smaller-is-better response.According to the proposed parameter combinations for each algorithm shown in Table 2, ten test problems of different sizes and specifications are solved five times in order to find the average objective values. Fig. 8shows the S/N ratios obtained for GA, MA, and SA, respectively. According to these results, the best parameter combination for each algorithm is found. For example, Fig. 8(A) shows that, for GA, parameters A (Npop), B (Pc), C (Pm), and D (Maxit) are better to be at second, third, third, and second levels, respectively.

The Taguchi method is performed by Minitab 16, and the parameter-tuned algorithms are coded in C# programming language and implemented on Intel Xeon E5-2660v2@2.5 GHz computers with 8 GB RAM and 25 MB Cache.

Table 2. Computational results for tuning GA, MA, and SA.

#	Algorithm Parameters					Response		Algorithm Parameters				Response
						MA	SA					GA
	A	B	C	D	E	Objective	Objective	A	B	C	D	Objective
1	1	1	1	1	1	2261.27	2537.88	1	1	1	1	2721.53
2	1	1	1	1	2	2176.18	2594.22	1	2	2	2	2311.90
3	1	1	1	1	3	2036.01	2547.36	1	3	3	3	2197.74
4	1	2	2	2	1	2186.32	2265.63	2	1	2	3	2407.95
5	1	2	2	2	2	2055.95	2233.44	2	2	3	1	2229.35
6	1	2	2	2	3	1903.39	2174.11	2	3	1	2	2217.82
7	1	3	3	3	1	2069.74	2378.25	3	1	3	2	2229.42
8	1	3	3	3	2	2035.24	2217.35	3	2	1	3	2423.62
9	1	3	3	3	3	1921.65	2182.28	3	3	2	1	2401.99
10	2	1	2	3	1	1940.55	2005.78	-	-	-	-	-
11	2	1	2	3	2	1998.20	2065.47	-	-	-	-	-
12	2	1	2	3	3	1946.91	2009.64	-	-	-	-	-
13	2	2	3	1	1	2178.23	2230.65	-	-	-	-	-
14	2	2	3	1	2	2014.53	2403.78	-	-	-	-	-
15	2	2	3	1	3	1896.36	2191.38	-	-	-	-	-
16	2	3	1	2	1	1994.45	2253.48	-	-	-	-	-
17	2	3	1	2	2	1961.22	1993.21	-	-	-	-	-
18	2	3	1	2	3	1949.61	2045.04	-	-	-	-	-
19	3	1	3	2	1	1914.17	2136.25	-	-	-	-	-
20	3	1	3	2	2	1980.70	1992.48	-	-	-	-	-
21	3	1	3	2	3	1829.30	2048.07	-	-	-	-	-
22	3	2	1	3	1	1820.30	1969.01	-	-	-	-	-
23	3	2	1	3	2	1731.87	1998.91	-	-	-	-	-
24	3	2	1	3	3	1667.94	2006.97	-	-	-	-	-
25	3	3	2	1	1	1919.07	2138.01	-	-	-	-	-
26	3	3	2	1	2	1877.49	2303.22	-	-	-	-	-
27	3	3	2	1	3	1888.12	2179.23	-	-	-	-	-

INSTANCE GENERATION

In order to generate random networks, which are close to the reality, network edges are classified into three levels based on the crowdedness criteria: crowded, semi-crowded, and less crowded edges. For each level, a specific rate is considered. For each network edge, multiplying the length by corresponding crowdedness rate determines the demand generation rate. According to the crowdedness criteria, the majority of candidate locations are selected among those network nodes, which are located in crowded areas. For entire generated instances, it is assumed that the length of edges follows a uniform distribution in [1, 10], crowdedness rates are 0.4, 0.8, and 1.2 for less crowded, semi-crowded, and crowded edges respectively, common service rate is 20, and maximum allowed waiting time is 0.35. Fig. 9 represents a network of 550 nodes and 65 candidate locations generated by the mentioned process.

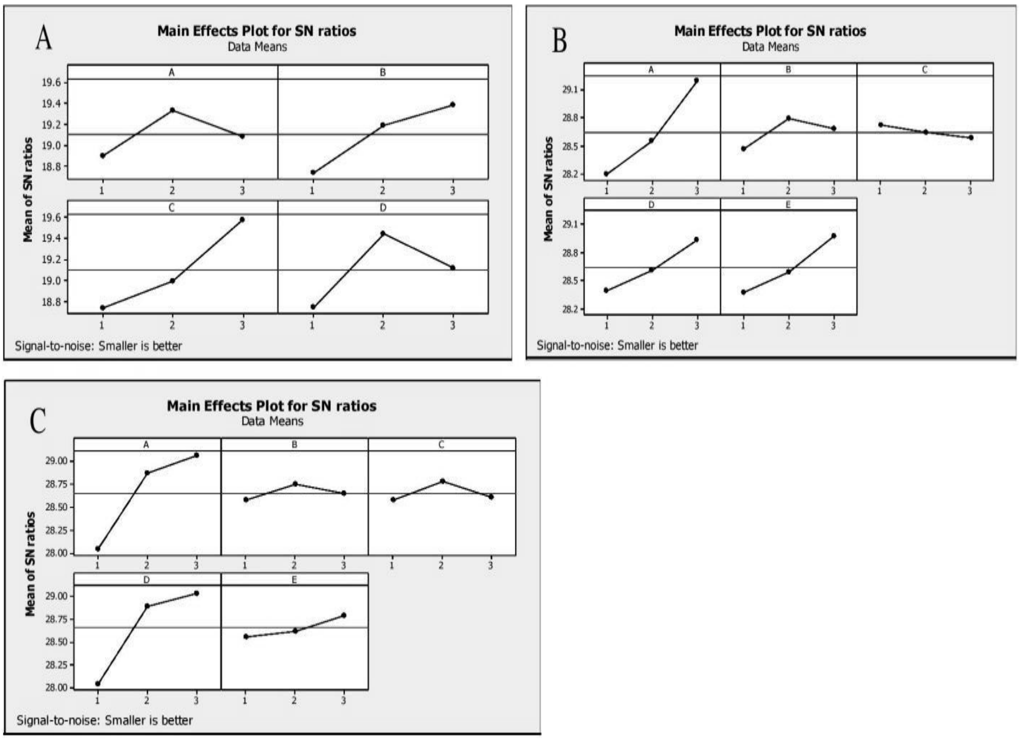


Fig. 8 .S/N ratio plot for (A) GA parameters; (B) MA parameters; (C) SA parameters.

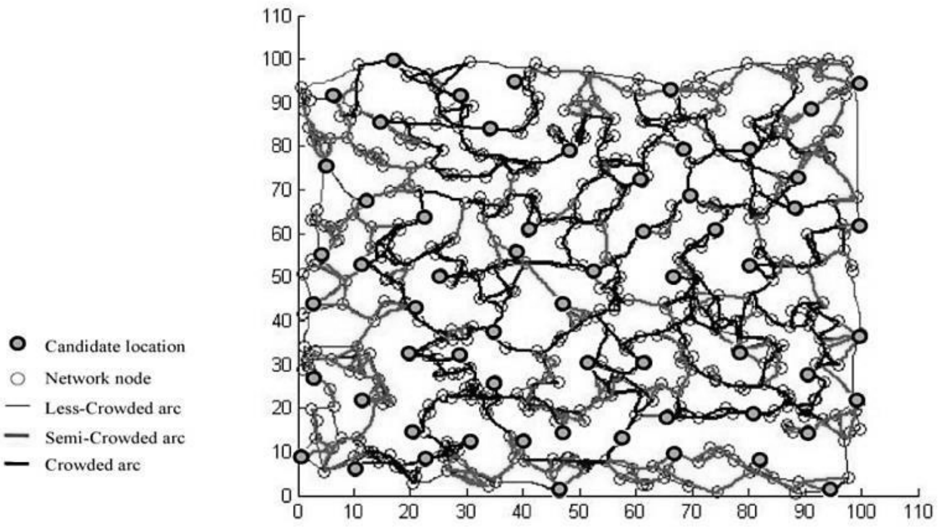


Fig. 9. A sample random network.

RESULTS AND DISCUSSION

The computational result of implementing the proposed solution methods on 24 problems of different sizes and specifications is shown in Table 3. In order to mitigate the effects of uncertainty, each algorithm is implemented ten times on each problem, and the average and best objective values along with the required CPU times of these runs are shown in this table.

As shown in Table 3, GAMS mathematical programming package failed to solve the majority of developed instances and even for very small size instancesthathave been solved by GAMS, the required CPU time was not reasonable. Since it is not possible to obtain the global optimum solution for the developed MINLP model, the performances of the proposed algorithms are compared together. Fig. 10compares the proposed metaheuristic algorithms according tothe average, best, and worst objective values along withthe required CPU times.As illustrated in thisfigure,MA outperforms GA and SAon the basis of objective function values, while, according to the required CPU times, GA outperforms the other two algorithms.

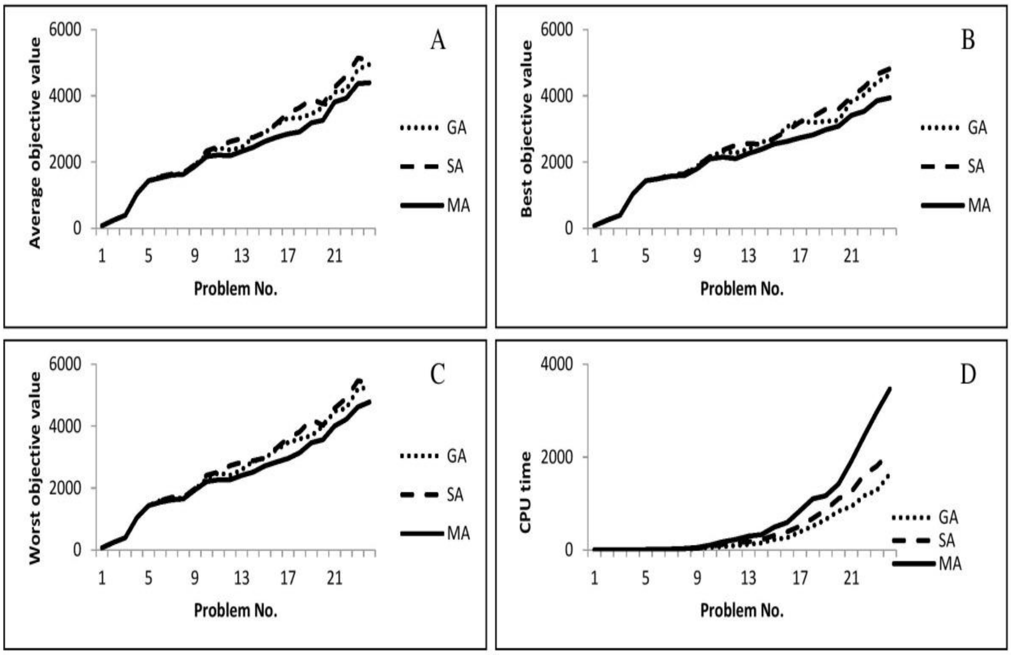


Fig. 10. (A) Average objective values; (B) best objective values; (C) worst objective values; (D) required CPU time of algorithms for different test problems.

Table 3. Computational results of solving methodologies.

#	V	c_{total}	J	P	Proposed GA			Proposed SA			Proposed MA			GAMS	
					Average	Best	Time	Average	Best	Time	Average	Best	Time	Objective	Time
1	6	10	4	2	71.6	71.6	1	71.6	71.6	1	71.6	71.6	1	71.6	2
2	10	12	5	2	243.6	243.6	1	243.6	243.6	2	243.6	243.6	3	243.6	32
3	15	15	10	3	392.6	392.6	1	392.6	392.6	3	392.6	392.6	4	392.6	252
4	20	30	12	3	1041	1041	2	1041	1041	5	1041	1041	6	1041.0	683
5	30	35	15	4	1436.7	1436.7	3	1436.7	1436.7	7	1436.7	1436.7	8	1436.7	1467
6	40	45	20	5	1548.4	1498.4	7	1559.2	1498.4	11	1498.4	1498.4	12	1498.4	2535
7	50	50	25	6	1621.1	1574.2	10	1637.6	1614.7	13	1605.8	1574.2	18	1605.8	3360
8	60	55	27	7	1655.5	1621.3	16	1663.7	1642.3	22	1626.9	1592.7	29	***	***
9	80	65	30	8	1912.6	1873.8	29	1922.2	1882.6	38	1874.3	1803.3	51	***	***
10	100	70	35	10	2217.7	2162.1	58	2338.1	2165.6	74	2163.8	2091.8	105	***	***
11	140	75	40	12	2429.9	2318.4	74	2462.1	2357.3	118	2211.5	2147.3	176	***	***
12	180	85	50	14	2358.3	2275.1	95	2612.9	2506.5	152	2187.6	2102.5	227	***	***
13	220	95	60	16	2447.1	2391.6	119	2712.2	2562.2	182	2320.4	2261.9	294	***	***
14	260	100	65	18	2739.3	2602.7	155	2741.5	2533.3	218	2443.6	2384.4	331	***	***
15	300	110	70	20	2889.4	2695.5	226	2894.2	2714.3	313	2622.8	2549.2	490	***	***
16	330	120	75	21	3139.8	3068.8	266	3162.3	2943.7	393	2748.7	2627.1	592	***	***
17	370	130	80	22	3319.7	3244.1	392	3473.9	3203.9	510	2846.6	2734.2	843	***	***
18	400	140	85	24	3328.2	3192.3	515	3638.8	3361.9	678	2912.4	2821.1	1099	***	***
19	430	150	90	24	3450.6	3227.1	656	3901.7	3592.4	863	3185.3	2793.6	1164	***	***
20	470	160	95	26	3651.7	3254.2	836	3759.2	3602.6	1104	3263.1	3084.7	1423	***	***
21	500	170	100	28	4107.3	3836.8	934	4256.4	3963.9	1252	3814.2	3411.4	1906	***	***
22	550	180	105	30	4176.8	4031.8	1164	4594.1	4264.5	1620	3932.1	3529.6	2455	***	***
23	600	190	110	30	4796.2	4408.9	1288	5138.2	4650.1	1808	4365.5	3848.7	2976	***	***
24	650	200	115	30	4944.7	4629.1	1638	5072.6	4813.5	2100	4392.2	3935.9	3464	***	***

Sign (***) denotes that GAMS(Bonmin solver) cannot find integer solution.

In order to compare the performance of the proposed algorithms statistically, one-way analysis of variance (ANOVA) for the average, best, and worstobtained objective values along withthe required CPU times was performed by using SPSS software. According to Table 4 obtained from ANOVA, algorithms significantly differ in the objective values and therequired CPU times. Table 5 shows the results of Tukey's test for multiple comparisons of the proposed algorithms. As can be seen from this table, while GA outperforms SA, and SA outperforms MA in terms of the required CPU times, MA outperforms GA, and GA outperforms SA in terms of objective values.

CONCLUSION AND FUTURE WORKS

In this paper, an MINLP model is developed for multiple-server facility location problem with stochastic and uniformly distributed demands within M/M/C queuing framework. It has been assumed that each customer is assigned to the closest open facility. Considering distributed demands along the network edges increases the complexity of the developed mathematical model. Since the proposed model is NP-hard, in addition to the utilization of GAMS optimization compiler, three metaheuristic algorithms including GA, SA, and MA were proposed to solve the model. In order to tune the parameters of these algorithms, the Taguchi method was used, and then the parameter-tuned algorithms were implemented on 24 test problems of different sizes and characteristics. Finally, in order to compare the performance of the proposed algorithms, one-way ANOVA method and Tukey's test were applied. The obtained results demonstrate that although GA requires less CPU times, MA finds better solutions according to the objective function values.

For future research, the problem can be modeled as a multi-objective problem in order to consider both customer and server aspects simultaneously. Furthermore, one can utilize other queuing frameworks by considering different distributions for demand generation, different service time distributions, or the capacity restrictionson facilities. In this case, the application of simulation optimization algorithms would be of great interest.Developing other heuristic or metaheuristic algorithms may also result in better solutions in shorter times.

Table 4. ANOVA for performance comparisons.

		Sum of Squares	**df**	**Mean Square**	**F**	**Sig.**
Average objective value	Between Groups	0.012	2	0.006	50.724	0.000
	Within Groups	0.008	69	0.000		
	Total	0.021	71			
Best objective value	Between Groups	0.111	2	0.006	43.070	0.000
	Within Groups	0.009	69	0.000		
	Total	0.020	71			
Worst objective value	Between Groups	0.013	2	0.006	55.623	0.000
	Within Groups	0.008	69	0.000		
	Total	0.021	71			
CPU Time	Between Groups	0.69	2	0.345	247.010	0.000
	Within Groups	0.087	69	0.001		
	Total	0.777	71			

Table 5. Tukey's test for multiple comparisons.

Tukey's Test	Algorithm Method	N	Subset for alpha = 0.05		
			1	2	3
Average objective value	MA	24	0.3161		
	GA	24		0.3360	
	SA	24			0.3478
Best objective value	MA	24	0.3163		
	GA	24		0.3375	
	SA	24			0.3461
Worst objective value	MA	24	0.3154		
	GA	24		0.3368	
	SA	24			0.3476
CPU time	GA	24	0.2182		
	SA	24		0.3241	
	MA	24			0.4579

REFERENCES

Aboolian, R., Berman, O. & Drezner, Z. 2009. The multiple server center location problem. Annals of Operations Research, **167**(1):337–352.

Alp, O., Erkut, E. & Drezner, Z. 2003. An Efficient Genetic Algorithm for the p-Median Problem. Annals of Operations Research, **122**(1):21–42.

Arkat, J. & Jafari, R. 2016. Network Location Problem with Stochastic and Uniformly Distributed Demands. International Journal of Engineering (IJE), TRANSACTIONS B: Applications, **29**(5):654–662.

Berman, O. & Drezner, Z. 2007. The Multiple Server Location Problem. The Journal of the Operational Research Society, **58**(1):91–99.

Bieniek, M. 2015. A note on the facility location problem with stochastic demands. Omega, 55:53–60.

Boffey, B., Galvão, R. & Espejo, L. 2007. A review of congestion models in the location of facilities with immobile servers. European Journal of Operational Research, **178**(3):643–662.

Boloori Arabani, A. & Farahani, R.Z. 2012. Facility location dynamics: An overview of classifications and applications. Computers & Industrial Engineering, **62**(1):408–420.

Brimberg, J. & Drezner, Z. 2013. A new heuristic for solving the p-median problem in the plane. Computers & Operations Research, **40**(1):427–437.

Černý, V. 1985. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications, **45**(1):41–51.

Eiben, A.E. & Smith, J.E. 2003. Introduction to Evolutionary Computing, Berlin, Heidelberg: Springer Berlin Heidelberg.

Falah, A.H. & Khorshid, E.A. 2014. Optimum modeling of a flexible multi-bearing rotor system. Journal of Engineering Research, **2**(2):155–181.

Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M.& Goh, M. 2012. Covering problems in facility location: A review. Computers & Industrial Engineering, **62**(1):368–407.

Garey, M.R. & Johnson, D.S. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness, New York, NY, USA: W. H. Freeman &amp; Co.

Gross, D., Shortle, J. F., Thompson, J. M.& Harris, C. M. 2008. Fundamentals of Queueing Theory 4th ed., New York, NY, USA: Wiley-Interscience.

Hajipour, V., Rahmati, S. H. A., Pasandideh, S. H. R.& Niaki, S. T. A. 2014. A multi-objective harmony search algorithm to optimize multi-server location–allocation problem in congested systems. Computers & Industrial Engineering, 72:187–197.

Hale, T.S. & Moberg, C.R. 2003. Location Science Research: A Review. Annals of Operations Research, **123**(1):21–35.

Hamaguchi, T. & Nakadeh, K. 2010. Optimal Location of Facilities on a Network in Which Each Facility is Operating as an M/G/1 Queue. Journal of Service Science and Management, **3**(3):287–297.

Hodgson, M.J. 1990. A Flow-Capturing Location-Allocation Model. Geographical Analysis, **22**(3):270–279.

Hu, D., Liu, Z.W. & Hu, W. 2013. Congestion service facilities location problem with promise of response time. Mathematical Problems in Engineering, 2013(3):1–11.

Kariv, O. & Hakimi, S.L. 1979. An Algorithmic Approach to Network Location Problems. I: The p-Centers. SIAM Journal on Applied Mathematics, **37**(3):513–538.

Khodaparasti, M., Ganji, M., Amirgholipour, S.& Sharifi, A. M. 2016. A new multi-objective cluster ensemble based on modularity maximization. Journal of Engineering Research, **4**(2):1–16.

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

Klose, A. & Drexl, A. 2005. Facility location models for distribution system design. European Journal of Operational Research, **162**(1):4–29.

Kuby, M., Lines, L., Schultz, R., Xie, Z., Kim, J.-G.& Lim, S. 2009. Optimization of hydrogen stations in Florida using the Flow-Refueling Location Model. International Journal of Hydrogen Energy, **34**(15):6045–6064.

Kuby, M. & Lim, S. 2007. Location of Alternative-Fuel Stations Using the Flow-Refueling Location Model and Dispersion of Candidate Sites on Arcs. Networks and Spatial Economics, **7**(2):129–152.

Kumar, R. 2012. Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms. International Journal of Machine Learning and Computing, **2**(4):365–370.

Larson, R.C. 1974. A hypercube queuing model for facility location and redistricting in urban emergency services. Computers & Operations Research, **1**(1):67–95.

Larson, R.C. 1975. Approximating the Performance of Urban Emergency Service Systems. Operations Research, **23**(5):845–868.

Lozano, M., Herrera, F. & Cano, J.R. 2008. Replacement strategies to preserve useful diversity in steady-state genetic algorithms. Information Sciences, **178**(23):4421–4433.

Marianov, V. & Serra, D. 2011. Location of Multiple-Server Common Service Centers or Facilities, for Minimizing General Congestion and Travel Cost Functions. International Regional Science Review, **34**(3):323–338.

Mladenović, N., Brimberg, J., Hansen, P.& Moreno-Pérez, J. A. 2007. The p-median problem: A survey of metaheuristic approaches. European Journal of Operational Research, **179**(3):927–939.

Moeini, M., Jemai, Z. & Sahin, E. 2015. Location and relocation problems in the context of the emergency medical service systems: a case study. Central European Journal of Operations Research, **23**(3):641–658.

Moré, J.J., Garbow, B.S. & Hillstrom, K.E. 1981. Testing Unconstrained Optimization Software. ACM Trans. Math. Softw., **7**(1):17–41.

Moscato, P. & Norman, M. 1992. A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. Parallel computing and transputer applications, **1**(1):177–186.

Mousavi, S. M., Niaki, S. T. A., Mehdizadeh, E.& Tavarroth, M. R. 2013. The capacitated multi-facility location–allocation problem with probabilistic customer location and demand: two hybrid meta-heuristic algorithms. International Journal of Systems Science, **44**(10):1897–1912.

Pasandideh, S.H.R. & Niaki, S.T.A. 2012. Genetic application in a facility location problem with random demand within queuing framework. Journal of Intelligent Manufacturing, **23**(3):651–659.

Pasandideh, S.H.R., Niaki, S.T.A. & Hajipour, V. 2013. A multi-objective facility location model with batch arrivals: two parameter-tuned meta-heuristic algorithms. Journal of Intelligent Manufacturing, **24**(2):331–348.

Qin, J., Xiang, H., Ye, Y.& Ni, L. 2015. A Simulated Annealing Methodology to Multiproduct Capacitated Facility Location with Stochastic Demand. The Scientific World Journal, 2015(1):1–9.

Rahmaniani, R. & Ghaderi, A. 2015. An algorithm with different exploration mechanisms: Experimental results to capacitated facility location/network design problem. Expert Systems with Applications, **42**(7):3790–3800.

Rahmaniani, R., Saidi-Mehrabad, M. & Ashouri, H. 2013. Robust capacitated facility location problem: Optimization model and solution algorithms. Journal of Uncertain Systems, **7**(1):22–35.

Snyder, L. V. 2006. Facility location under uncertainty: a review. IIE Transactions, **38**(7):547–564.

Tavakkoli-Moghaddam, R., Safaei, N. &Sassani, F. 2009. A memetic algorithm for the flexible flow line scheduling problem with processor blocking. Computers & Operations Research, **36**(2):402–414.

Wang, Q., Batta, R. & Rump, C.M. 2002. Algorithms for a Facility Location Problem with Stochastic Customer Demand and Immobile Servers. Annals of Operations Research, **111**(1):17–34.

Yeniay, Ö. 2005. Penalty function methods for constrained optimization with genetic algorithms. Mathematical and Computational Applications, **10**(1):45–56.

Submitted: 14/02/2017
Revised : 05/05/2017
Accepted : 27/06/2018

حل مسألة موقع منشأة ذات خوادم متعددة مع مطالب عشوائية على طول حواف الشبكة

***محمود غولابي، ** غوخان إزبيراك و ***جمال أركات**

*قسم الهندسة الصناعية، جامعة غرنة الأمريكية، غرنة، عبر مرسين 10، تركيا

** قسم الهندسة الصناعية، جامعة الشرق المتوسط، فاماغوستا، عبر مرسين 10، تركيا

***قسم الهندسة الصناعية، جامعة كردستان، سنندج، إيران

الخـلاصة

يناقش هذاالبحث مسألة موقع الشبكة للمنشآت ذات خوادم متعددة المرافق والمُعرضة للازدحام. ويتم اختيار عدد من المنشآت من بين العديد من المواقع المُقترحةلتلبية متطلبات العملاء. ولكل حافة شبكة، يتم توزيع العملاء المطابقين بشكل متساوي على طول الحافة وفقاً لعملية توزيع بواسون (Poisson). وبالإضافة إلى ذلك، يُعتبر عدد الخوادم في كل منشأة كمتغير قرار وفترة الخدمة لكل خادم تتبع توزيع أسي. تم تطوير نموذج رياضي لتقليل المجموع الكلي لأوقات السفر المتوقعة للعملاء والمجموع الكلي المتوقع لأوقات الانتظاروذلك باستخدام تحليل نظام قوائم الانتظار. ونظراً لأن مسائل تحديد مواقع شبكة توزيع الخدمات تُعد NP-hard، لذلك تم بحث وتطوير ثلاثة من خوارزميات التجريبيات (Metaheuristic) لحل المسألة المقترحة، وهم كالاتي: الخوارزمية الجينية (Genetic Algorithm)، خوارزمية ميميتيك (Memetic Algorithm) ومحاكاة التلدين (Simulated Annealing). وأظهرت النتائج أفضلية استخدام خوارزمية ميميتيك المُقترحةعلى الخوارزميتين الأخريتين من حيث القيم الموضوعية.