A Genetic Algorithm for Multi-Period Location Problem with Modular Emergency Facilities and Backup Services

Roghayyeh Alizadeh† and Tatsushi Nishi†

In this paper, we develop a dynamic model and a solution procedure for maximal covering location problem with backup services for locating emergency facilities like fire stations. In this study, two main concepts are used to model a real life condition to improve the service level for demand nodes. These concepts are modular arrangement of different fire trucks and ambulances, as an example for modules, and backup services for demand nodes. The mathematical model is a pure 0-1 integer programming problem. To solve the problem, an efficient genetic algorithm is used because the algorithm has appeared as a strong method to solve integer programming problems. Some test problems are generated and solved by CPLEX, GA and a developed heuristic to be able to compare the results with managerial insights.

1. Introduction

Unexpected disasters are always a threat to human lives and assets. These disasters may be nationwide like earthquakes, typhoons, floods, etc. or other kinds of incidents that may happen to individuals like fires, car accidents, electrocutions (injury by electric shock), water rescue and etc., which require to call an emergency to ask for help. According to the importance of these incidents and disasters, managing them in the case of trying to provide the best service for the emergency calls seems so vital in order to reduce the casualties. For this importance, providing backup services for service request calls in addition to the primary service would be an approach while locating the emergency facilities. Emergency facility location problem seeks to find the optimal locations for facilities like fire stations, police stations, emergency departments and roadside emergency services. The set covering location problem (SCLP) and the maximal covering location problem (MCLP) are two categories of covering problems, which are applicable in emergency facility siting problems. However, most studies in the literature have used MCLP as seen in [1–5]. For a review on location problems for emergency facilities, interested readers may refer to [6].

Emergency facilities mostly consist of constructional units, like different kind of firefighting trucks in fire stations, ambulances in emergency stations or police cars in police stations. These separable units that specify the size and the capacity of the facilities are called modules of the facility, which should be considered to be located [7]. Correia and Captivo called the location problems with such capacity constraints as modular capacitated location problems [8]. In their study, they tried to find the location of capacitated facilities and the objective function is to give services to the customers at a minimum cost. In this paper, the main structure of the emergency facility is called “facility” and the equipment like different kind of cars and trucks are called “modules”. Without backup service, when an emergency call for service occurs, the nearest module serves the demand and would be no more available. If backup service had been prepared and other call arises for that module before the previous one returns the base, the backup module can go for the duty. This backup service might have been provided from the other existing modules in the base or from the other facilities that cover the demand node for a backup service. The need for backup service seems to be mandatory in regions of high demand situations [3]. Hogan and ReVelle provided a brief review of backup services as well as a multi-objective model formulation considering both primary and backup services for demand nodes [9]. Their approach, which implicitly attempts to capture finite facility capacities, is unique because before all nodes being assigned to primary coverage, backup coverage could be provided for some nodes. Pirkul and Schilling considered the allocation of primary and backup service to capacitated facilities, but in the context of minimizing average distance [4]. The main components of the model in our study are multiperiod, capacitated facilities and two levels of service.
for demand nodes. The most important one, i.e. modular design of facilities has been studied more or less in the literature, but our work encompasses all the concepts, trying to make a more practical model for real life situation. In our model, the facilities (for example, fire stations’ buildings and locations) and the equipment (modules like fire trucks and ambulances) are supposed to have different decision variables, in this way that the demand nodes would be assigned to the modules, and the modules would be assigned to the facilities. A new formulation as well as a solution procedure for the multi-period modular maximal covering location problem is proposed in this paper. The model development is outlined in 2, followed by the solution method in 3. The computational experiments are organized in 4. Conclusions are stated in 5.

2. Multi-Period Location Problem with Modular Emergency Facilities with Backup Services

In the mathematical model, once the facilities are located, their location cannot be changed, because changing the location would impose the fixed cost. But we can have different arrangements for the module assignments in each time period and in the other word, the modules may transfer between facilities according to the demand changes. The size of modules, in this formulation, refers to the number of modules that would be assigned to each facility. In addition, the capacity of each module is related to the amount of service that it can provide, which is an inherent specification of the modules. The notation for our model is as follows:

Indices and sets:
- \( i \in \{1, 2, ..., I\} \) Index of potential facility sites.
- \( j \in \{1, 2, ..., J\} \) Index of customers.
- \( l \in \{1, 2, ..., L\} \) Index of modules.
- \( k \in \{1, 2, ..., K\} \) Index of size.
- \( t \in \{1, 2, ..., T\} \) Time periods.

Parameters:
- \( c^m_{ij} \) Binary parameter which is 1 if \( d_{ij} \leq S^m \), 0 otherwise.
- \( c^b_{ij} \) Binary parameter which is 1 if \( d_{ij} \leq S^b \), 0 otherwise.
- \( c_l \) Capacity of each module \( l \).
- \( d^m_{jlt} \) Expected demand of node \( j \) for primary service of module \( l \) at time period \( t \).
- \( d^b_{jlt} \) Expected demand of node \( j \) for backup service of module \( l \) at time period \( t \).
- \( d_{ij} \) Distance between potential facility \( i \) and demand node \( j \).
- \( e_{lk} \) Size of size index \( k \) of module \( l \).
- \( p \) Number of facilities to be sited.
- \( q_i \) Total number of available module \( l \).
- \( S^m \) Maximum service distance for primary services.
- \( S^b \) Maximum service distance for backup services.

Decision variables (binary variables):
- \( x^m_{ijlt} \) 1 if demand node \( j \) gets primary service from module \( l \) assigned to facility \( i \) at period \( t \), 0 otherwise.
- \( x^b_{ijlt} \) 1 if demand node \( j \) gets backup service from module \( l \) assigned to facility \( i \) at period \( t \), 0 otherwise.
- \( y_{iklt} \) 1 if module \( l \) with size \( k \) is assigned to facility \( i \) at period \( t \), 0 otherwise.

Thus, the mathematical model can be formally stated as:

\[
\max \sum_{i} \sum_{j} \sum_{l} \sum_{t} (c^m_{ij} d^m_{jlt} x^m_{ijlt} + c^b_{ij} d^b_{jlt} x^b_{ijlt})
\]

\[
y_{iklt} \leq z_i \quad \forall i, l, k, t
\]

\[
x^m_{ijlt} + x^b_{ijlt} \leq \sum_{k} y_{iklt} \quad \forall i, j, l, t
\]

\[
\sum_{j} (d^m_{jlt} x^m_{ijlt} + d^b_{jlt} x^b_{ijlt}) \leq \sum_{k} e_{lk} c_i y_{iklt} \quad \forall i, l, t
\]

\[
\sum_{k} y_{iklt} \leq 1 \quad \forall i, l, t
\]

\[
z_i \leq p
\]

\[
\sum_{k} e_{lk} y_{iklt} \leq q_i \quad \forall l, t
\]

\[
z_i, y_{iklt}, x^m_{ijlt}, x^b_{ijlt} \in \{0,1\} \quad \forall i, j, l, k, t
\]
Constraints (7) also restrict the number of available modules. Suppose there are just $q_l$ numbers of module $l$ available, the left-hand side of the constraints calculate the total number of module $l$ that are sited in facilities and this number should not exceed the value $q_l$. Constraints (8) are imposed for the nature of decision variables, where all variables are binary variables that makes our model to be a 0-1 facility location problem.

3. Solution Approach

3.1 A Solution Method Based on Genetic Algorithm

Metaheuristics like Tabu Search (TS), Simulated Annealing (SA) and Genetic Algorithm (GA) are applicable methods to solve a wide range of optimization problems. The metaheuristics’ general rule to approach the new solutions is to generate a good neighborhood solution as good up to now. In this case, GA has a big difference with other methods in this way that in GA “good” does not come from the whole solution (as in TS and SA) but comes from the parts of the solution. In other words, it pays attention to the parts of the solution that have made it to be a good solution. GA then uses these parts in its recombination mechanism to produce new solutions [10].

To solve the developed multi-period location problem with modular emergency facilities with back-up services, a genetic algorithm (GA) is applied in this research, as the evolutionary algorithms have been proven to be one of the best methods to solve facility location models and it is a better approach to solve MCLP compared to local search procedures such as SA and the TS [11]. GA as an evolutionary algorithm has got many extensions and also has been used to solve various kinds of problems like unconstrained and constrained optimization problems. Solving constrained problems needs more effort to design and code the algorithm. There are many techniques to handle constraints and the most well-known one is the penalty function method, in which the violation amount of constraints are accommodated in the objective function, so that the problem is converted to an unconstrained one [12]. Some disadvantages have been pointed to this technique and the major one is the difficulty to select the penalty factors. The other method to face the constrained optimization problem which is used in the algorithm design of this paper is to repair the infeasible solutions by modifying them according to the constraints to generate feasible solutions. In this way that in the fitness function calculation procedure, we start from constraint (6) to assure the number of facilities. The checking order for constraints would be (6), (2), (5), (7), (3) and (4). For each constraint the right-hand side and left-hand side values are obtained and if the constraint is found feasible, the next constraint will be checked, otherwise the bits of the solution are modified to satisfy the constraint. After obtaining feasible solutions the fitness function would be calculated as (1). It is also noteworthy that the GA used is in the category of continues genetic algorithm, and the reason to use continues space to generate initial solutions and operate uniform crossover and mutation is that this kind of GA makes the search space larger and can have more exploration options. Also uniform crossover and mutation seem to have better functionality in the continuous GA.

In a typical genetic algorithm, the main mechanisms are: $N$ population size, $pc$ crossover ratio and $pm$ mutation ratio. The main structure of genetic algorithm used in this paper is:

0 Encoding scheme: representation is one of the important steps in developing a GA, as it has direct influence on runtime and also crossover and mutation. To have a larger search space, we use a continuous interval between 0 and 1 for each bit and then round it to a binary one whenever it is needed. Each bit represents the status of the related variable according to the definition of variables in the previous section.

1 Creating initial populations $N$.

2 Repair the solutions to create feasible solutions from possibly infeasible solutions and calculate fitness function for each chromosome in the population.

3 Producing new population. Repeat the following steps for $N$ times to produce new population.

3.1 Selecting two parents from current population using selection approach. In this study, a roulette wheel selection approach is used.

3.2 Applying crossover operator on the selected parents with ratio of $pc$ and create offspring.

3.3 Applying mutation operator on offspring with ratio of $pm$.

3.4 Adding new offspring to the population.

4 Selecting new population based on the fitness function value.

5 Check the termination criteria, if it is not satisfied go to Step 2.

3.2 Solution Representation

One of the important parts of metaheuristic algorithms is defining a solution representation. This solution representation in GA is called Chromosome. As we have four decision variables, the chromosome contains the information of all these variables as shown in Fig. 1. The bits 1 to $i$ in the chromosome illustrates the values related to variable $z_{i}$, having the exact meaning that is defined before in the decision variables introduction. The bits from $i+1$ to $i+l.l.k.t$ contain the solution for variable $y_{ilkt}$. For instance, if $i=5$, $l=2$, $t=2$, $k=3$, there would be 5 first bits containing the solutions for variable $z_{i}$, and the following 60 bits starting from the bit number 6 to 66 contains the solutions of variable $y_{ilkt}$. The remained
bits illustrate the variables $x_{ijlt}^m$ and $x_{ijlt}^b$ in the same way. The equation (1) is used to calculate the fitness function. Having specified bits for each variable and the ability of the algorithm to generate feasible solutions, we can obtain meaningful solutions that can be utilized in the decision making evaluations and comparisons.

![Chromosome representation](image)

3.3 GA Operators

3.3.1 Crossover

In genetic algorithm and evolutionary algorithms, crossover operator is used for combining the genetic information of two parents to generate new offspring from the population. Running many test examples showed $pc = 0.9$ fits better for our problem. First two new chromosomes (same as the initial population was produced) should be produced. We call them $R_1$ and $R_2$. We also call the two chosen parents $x_1$ and $x_2$. The new offspring are called $y_1$ and $y_2$ which are produced in this way: $y_1 = (R_1 * x_1) + (R_2 * x_2)$ and $y_2 = (R_2 * x_1) + (R_1 * x_2)$, respectively. Fig. 2 illustrates a small size example of the uniform crossover operator. By doing this procedure, we have obtained two offspring that contain the continuous values between 0 and 1, and they will be converted to binary values when they will be used in the fitness function to calculate the objective function.

![A simple illustration of the uniform crossover operator](image)

3.3.2 Mutation

While crossover tries to converge to a specific point in the landscape, the mutation does its best to avoid convergence and explore more areas. One of the main benefits of this operator is to avoid trapping into a local optimum. For mutation ratio $pm$ the ratio 0.1 is used in this research. To establish the uniform mutation mechanism, we take the following steps:

1. Calculate the number of elements in the chromosome ($n$)
2. Choose $I = Rn$ ($R$ is equal to 0.01 in this paper) random sample genes of $n$ genes in the chromosome. ($R$ is the percent of genes that would be selected to perform mutation operator out of $n$ genes, so $I$ defines the number of genes that we select for mutation. For example, if the number of bits or genes ($n$) in the chromosome is 300 and 0.01 of them has to be selected, it gives that $I = 3$ genes have to be selected to conduct the mutation).
3. Produce $I$ random values between (-0.1, 0.1).
4. Add or subtract the random values obtained in Step 3 to/from the samples chosen in Step 2.
5. Stop.

The parameters for $pc$ and $pm$ have been chosen to the above values after running the algorithm for different values and these values were selected because they were producing better results for test problems used in this paper. The termination criteria for this problem sets to 70 iterations.

3.4 A heuristic to evaluate the results

In this section, we introduce a heuristic algorithm as an auxiliary tool to enable us to evaluate the quality of produced results for computational experiments, especially for the cases that CPLEX is unable to reach the optimal solution. The heuristic starts with using the possibly covered demands to locate the $p$ facilities that can cover the maximum number of demands for primary and backup service. In the next step, the demand nodes for both kinds of services and each module in each period that can be covered by most facilities are allocated to the $p$ located facilities to obtain the number of the necessary modules. The total number of allocated demands for each module and in each period are compared with the existing resources to refrain infeasibility in Step 4. Step 1 of the method satisfies constraint (6), constraints (2) and (3) are satisfied in Step 2 and constraints (4), (5) and (7) are satisfied in Steps 3 and 4. At the final step 5, the objective function for the heuristic method ($OF_{VIH}$) is obtained by summing the total number of modules. The heuristic algorithm is as follows.

Step 1 $\mu_i = \sum_{jlt} (c_{ijlt}^m d_{ijlt}^m + c_{ijlt}^b d_{ijlt}^b)$. Sort $\mu_i$ and select $p$ maximum values. $i_{pmax}$ is the set of $i$ that provides $p$ maximum values of $\mu_i$ and put the related $z_{i \in i_{pmax}} = 1$, otherwise $z_i = 0$.

Step 2 Calculate $\phi_{jlt}^m = \sum_{i \in i_{pmax}} c_{ijlt}^m d_{ijlt}^m$ for each $j, l, t$. Sort $\phi_{jlt}^m$ with respect to $j$ for each $l$ and $t$, and allocate the $p$ first largest values into $i = 1, ..., p$ opened facilities and set $x_{ijlt}^m = 1$ such that $\sum_{ij} x_{ijlt}^m = p$. The selected $j$ is added to the set of $i_{pmax}$. Calculate $\phi_{jlt}^b = \sum_{i \in i_{pmax}} c_{ijlt}^b d_{ijlt}^b$ without the allocated $j$ where $x_{ijlt}^b = 1$ for each $i, l, t$. Sort $\phi_{jlt}^b$ with respect to $j$ for each $l$ and $t$ and allocate the $p$ first largest values into $i = 1, ..., p$ opened facilities and set $x_{ijlt}^b = 1$ such that $\sum_{ij} x_{ijlt}^b = p$. The selected $j$ is added to the set of $i_{pmax}$.

Step 3 Set $\sigma_{lt} = 0$, $\theta_{lkt} = 0$, $y_{ijkt} = 0$. $\delta_{ijlt} = c_{ijlt}^m d_{ijlt}^m + c_{ijlt}^b d_{ijlt}^b$.
Step 4 For each $l,t$, iterate $\theta_{ilt} \leftarrow \theta_{ilt} + \delta_{ilt}$, until $\theta_{ilt} \leq (\max_k \epsilon_{ikt})c_l$ for $i \in i_{p_{\text{max}}}^l$ and $j \in j_{p_{\text{max}}}^l$.

For each $l,t$, iterate $e = \left\lfloor \frac{\theta_{ilt}}{c_l} \right\rfloor$, if $k = e$, $y_{ikt} = 1$ for $i \in i_{p_{\text{max}}}$, and for $k = 1, \ldots, K$. For each $l,t$, iterate $\sigma_{ilt} \leftarrow \sigma_{ilt} + \epsilon_{ikt}y_{ikt}$ until $\sigma_{ilt} \leq q_l$ for $i \in i_{p_{\text{max}}}$.

Step 5 Obtain $OFV_{ilt} = \sum_{l,t} \sigma_{ilt}$.

4. Computational Experiments

To have a better insight of the model and its application in real life, there is a simple illustration of a problem studied in two time periods in Fig. 3. In this figure, black filled nodes are the demand nodes for primary service and white filled nodes have been used to refer to the backup service request, also black hexagons are the fire stations located according to variable $z_l$. The circles around the facilities show the covered area (black is for covered area for primary service and gray depicts the covered area with back-up service). The numbers around the hexagons are showing the number of each module assigned to each facility. In this example, we suppose that there are three kinds of modules in which the total number of modules should be the same in each time period. There may be some nodes colored in black, which are in the primary coverage area of one facility, but in the secondary coverage of another facility. It shows that the primary service has been provided from the main facility, but the backup service has been provided from other one due to insufficient resources. To generate test problems, the parameters of the problem are generated randomly. The distance between nodes are generated using the Euclidean distance of two-dimensional coordinates having uniform distribution between 1 and 100. The expected demand for primary service $d_j^{pm}$ and backup service $d_j^{bmt}$ for each demand node $j$ and module $m$ in each time period $t$ are generated using uniform integer distribution between 0 and 2. The covering radius for primary service $S^m$ and backup service $S^b$ are set to be 30 and 35, respectively. The number of available modules for each kind of module is generated using a uniform integer distribution for each interval indicated in Tables 1 and 2. It is worth mentioning that all instances are solved using GAMS (CPLEX solver)/MATLAB software on a PC with a 3.4-GHz Core i7-6700 CPU and 8 GB of RAM running Windows 10 (64 bit). Computational results of the problem are summarized in Tables 1 and 2.

In Tables 1 and 2, the first column contains the information for number of demand nodes and potential locations for facilities. Column $p$ is for the number of facilities to be located, $T$ contains the number of time periods and $l$ refers to the type of available modules. Furthermore, the largest possible size of each module was set to be at most 3 ($\epsilon_{ikt} = K = 3$). Two columns under “CPLEX”, “GA” and “Heuristic” contain the objective function value as “OFV” and the computational time as “time” in seconds.

As the GA algorithm is a stochastic method that may produce different solution in each time the problem is solved, we have solved each test problem 10 times and the results in the Tables 1 and 2 contain the averaged values of these 10 times running for both “OFV” and “time”. The third column under GA calculates the gap between the results of CPLEX and GA in percent. Comparing the computational time for the CPLEX and GA, GA does not show any specific superiority. The heuristic also has an acceptable performance as its main application is to investigate the results trend. Fig. 4 illustrates the OFV values of three solving procedures. The illustrated diagrams give a better insight than the results in Table 1. These diagrams show that the CPLEX has the maximum values among the three methods and the heuristic has the lower values. It also should be added that although the heuristic has the lower values, but its trend is the same as CPLEX. The GA also is placed between two other diagrams, but much closer to the CPLEX that approves the ability of this proposed method to obtain satisfactory values. The test problems generated in Table 1 may just have some limited practical application and are mostly used to be able to evaluate the solving procedures, which necessitates to develop larger test problems as in Table 2 that can have real life applications like a town or city.

These large scale problem instances are solved by all methods and the results are summarized in Table 2. We should note that because of the inherent feature of variables which are all binary variables, the complexity of problems increases exponentially as the problem’s size increase. For which the CPLEX was able to solve the problems up to 600 nodes (only for some of the test problems of this size), but the proposed GA is still able to solve larger problem instances. The results in Table 2 show that for problems up to 600 nodes, GA has satisfying values, where the maximum gap is 0.035 also GA is able to solve the problems in a shorter time than CPLEX.

The most important observation is related to problems with 800 and 1000 nodes. For these problems, CPLEX is unable to solve them and runs out of memory, but GA has still satisfactory performance at a
Table 1  Computational results for CPLEX, GA and heuristic method for small size problems

| i, j | p   | T   | l   | q1   | OFV time | OFV time | %gap | OFV time | time |
|------|-----|-----|-----|------|----------|----------|------|----------|------|
| 200  | 10  | 3   | 3   | 15,25| 179      | 8        | 175  | 8        | 28   | 0.02 | 162  | 0.21 |
|      |     |     |     | 20,30| 226      | 8        | 211  | 29       | 207  | 0.06 | 207  | 0.21 |
| 4    | 15,25| 234  | 13  | 227  | 28       | 0.02     | 207  | 0.24     |
|      |     |     |     | 20,30| 293      | 33       | 274  | 29       | 270  | 0.28 |
| 5    | 15,25| 302  | 21  | 292  | 33       | 0.03     | 270  | 0.39     |
|      |     |     |     | 20,30| 379      | 24       | 357  | 36       | 345  | 0.32 |
| 4    | 15,25| 394  | 18  | 383  | 45       | 0.02     | 345  | 0.39     |
|      |     |     |     | 20,30| 479      | 36       | 469  | 47       | 450  | 0.37 |
| 300  | 20  | 3   | 3   | 30,50| 363      | 25       | 349  | 35       | 342  | 0.03 |
|      |     |     |     | 40,60| 450      | 26       | 426  | 36       | 412  | 0.05 |
| 4    | 30,50| 463  | 32  | 448  | 43       | 0.03     | 441  | 0.81     |
|      |     |     |     | 40,60| 591      | 43       | 567  | 47       | 567  | 0.88 |
| 5    | 30,50| 604  | 45  | 589  | 56       | 0.02     | 570  | 1.06     |
|      |     |     |     | 40,60| 751      | 70       | 735  | 57       | 720  | 0.98 |

Table 2  Computational results for CPLEX, GA and heuristic method for large size problems

| i, j | p   | T   | l   | q1   | OFV time | OFV time | %gap | OFV time | time |
|------|-----|-----|-----|------|----------|----------|------|----------|------|
| 500  | 30  | 5   | 3   | 50,70| 905      | 536      | 882  | 164      | 0.025 |
|      |     |     |     | 65,85| 1112     | 407      | 1103 | 178      | 1100 | 3.8  |
| 4    | 50,70| 1185 | 514 | 1172 | 179     | 0.010   | 1155 | 4.2      |
|      |     |     |     | 65,85| 1485     | 399      | 1480 | 188      | 1455 | 4.6  |
| 7    | 50,70| 1267 | 400 | 1233 | 184     | 0.026   | 1229 | 4.6      |
|      |     |     |     | 65,85| 1285     | 422      | 1259 | 202      | 1260 | 4.8  |
| 4    | 50,70| 1659 | 1130| 1634 | 199     | 0.015   | 1617 | 6.3      |
|      |     |     |     | 65,85| 2050     | 889      | 2039 | 210      | 2037 | 6.2  |
| 600  | 40  | 5   | 3   | 70,90| 1205     | 310      | 1190 | 182      | 0.012 |
|      |     |     |     | 80,100| 1355     | 331      | 1330 | 180      | 1323 | 5.8  |
| 4    | 70,90| 1585 | 562 | 1566 | 203     | 0.011   | 1545 | 7.5      |
|      |     |     |     | 80,100| 1875     | 708      | 1722 | 233      | 1695 | 7.9  |
| 7    | 70,90| 1687 | 947 | 1680 | 218     | 0.004   | 1638 | 7.8      |
|      |     |     |     | 80,100| 1897     | 1091     | 1850 | 216      | 1840 | 8.1  |
| 4    | 70,90| -    | -   | -    | 2477     | 286      | 2163 | 10.8     |
|      |     |     |     | 80,100| -        | 2622     | 292   | 2457     | 10.8  |
| 800  | 50  | 5   | 3   | 95,110| -        | -        | -    | -        | -    |
|      |     |     |     | 100,120| -        | -        | -    | -        | -    |
| 4    | 95,110| -    | -   | -    | 2123     | 410      | 1995 | 17.9     |
|      |     |     |     | 100,120| -        | 2223     | 421   | 2145     | 18.6  |
| 7    | 95,110| -    | -   | -    | 2210     | 380      | 2121 | 18.6     |
|      |     |     |     | 100,120| -        | 2266     | 398   | 2260     | 18.9  |
| 4    | 95,110| -    | -   | -    | 2800     | 466      | 2793 | 25.6     |
|      |     |     |     | 100,120| -        | 2930     | 485   | 2903     | 25.7  |
| 1000 | 60  | 5   | 3   | 110,130| -        | -        | -    | -        | -    |
|      |     |     |     | 115,140| -        | -        | -    | -        | -    |
| 4    | 110,130| -    | -   | -    | 2280     | 701      | 2255 | 35.3     |
|      |     |     |     | 115,140| -        | 2400     | 712   | 2390     | 34.6  |
| 7    | 110,130| -    | -   | -    | 2411     | 845      | 2399 | 37.5     |
|      |     |     |     | 115,140| -        | 2633     | 845   | 2620     | 36.7  |
| 4    | 110,130| -    | -   | -    | 3170     | 968      | 3097 | 47.6     |
|      |     |     |     | 115,140| -        | 3393     | 980   | 3386     | 57.1  |
reasonable computational time. The main purpose for proposing the heuristic method was to provide a comparing tool to evaluate the performance of the GA for the cases that CPLEX is unable to solve. According to the values in Table 2 and the Fig. 5, that depicts these results in schematic format, it is apparent that the heuristic has kept its correct trend and is producing well enough values as a base to evaluate the GA. This fact approves that we can rely on the proposed GA to solve larger problems and obtain satisfactory results.

5. Conclusion

In this paper, an extension of maximal covering location problem (MCLP) has been developed for locating emergency facilities, composed of discrete structural components. These components are called modules of facilities. In the developed model, demand nodes are assigned to modules first and then modules are allocated to facilities. The problem is formulated as a 0-1 integer programming model. We utilize a genetic algorithm to solve the problem because of this metaheuristic’s strength to solve binary optimization problems and other extension of MCLP. In addition, a heuristic method has been proposed to enable us to assess the results obtained from the GA. The computational results illustrate a very small gap in the objective function values for the test problems that could be solved with the exact method. Our future research is to try to solve the problems of real cases to observe the model and develop a new mathematical model for emergency facility location, which could be a combination of SCPLP and MCLP to exploit all the advantages of these two problems.

Acknowledgements

This research was supported in part by the funding provided by JSPS KAKEN HI KIBAN (A) 18H03826.

References

[1] N. R. Paul, B. J. Lunday and S. G. Nurre: A multiobjective, maximal conditional covering location problem applied to the relocation of hierarchical emergency response facilities; Omega, Vol. 66, Part A, pp. 147–158 (2017)
[2] P. Yin and L. Mu: Modular capacitated maximal covering location problem for the optimal siting of emergency vehicles; Applied Geography, Vol. 31, pp. 247–254 (2012)
[3] H. Pirkul and D. A. Schilling: The capacitated maximal covering location problem with backup service; Annals of Operations Research, Vol. 18, pp. 141–154 (1989)
[4] H. Pirkul and D. A. Schilling: The siting of emergency service facilities with workload capacities and backup service; Management Science, Vol. 34, pp. 896–908 (1988)
[5] R. Alizadeh and T. Nishi: Mixed-integer bilevel dynamic hub location problem for freight transportation; Proceedings of 2018 International Symposium on Flexible Automation, Paper 37 (2018)
[6] X. Li, Z. Zhao, X. Zhu and T. Wyatt: Covering models and optimization techniques for emergency response facility location and planning: a review; Mathematical Methods of Operations Research, Vol. 74, No. 3, pp. 281–310 (2011)
[7] I. Correia and T. Melo: A multi-period facility location problem with modular capacity adjustments and flexible demand fulfillment; Computers & Industrial Engineering, Vol. 110, pp. 307–321 (2017)
[8] I. Correia and M. E. Captivo: Bounds for the single source modular capacitated plant location problem; Computers & Operations Research, Vol. 33, No. 10, pp. 2991–3003 (2006)
[9] K. Hogan and C. ReVelle: Concepts and applications of backup coverage; Management Science, Vol. 32, pp. 1434–1444 (1986)
[10] E. Falkenauer: Applying genetic algorithms to real-world problems; L. D. Davis, K. De Jong, M. D. Vose & L. D. Whitley (Eds.), Evolutionary Algorithms, pp. 65–88, Springer (1999)
[11] M. H. F. Zarandi, S. Davari and S. A. Haddad Sisakht: The large-scale dynamic maximal covering location problem; Mathematical and Computer Modelling, Vol. 57, Nos. 3-4, pp. 710–719 (2013)
[12] A. Haissat, A. Diabat and I. Rahwan: A genetic algorithm approach for location-inventory-routing problem with perishable products; Journal of Manufacturing Systems, Vol. 42, pp. 93–103 (2017)
Authors

Roghayyeh Alizadeh
Alizadeh Roghayyeh received M.S. degree of industrial engineering in 2015 from Alzahra University, Tehran, Iran. From 2018, She has been a Ph.D. student with Graduate School of Engineering Science, Osaka University. Her research interests include location problems in supply chain management.

Tatsushi Nishi (Member)
Tatsushi Nishi received Ph.D. degree from Kyoto University in 2001. From 2001 to 2006, he was an Assistant Professor with the Department of Electrical and Electronic Engineering, Okayama University. He is currently an Associate Professor with Mathematical Science for Social Systems, Graduate School of Engineering Science, Osaka University. His research interests include discrete optimization, scheduling, robotics and supply chain optimization.