A Genetic Algorithm For P-Median Location Problem

M. Mahmoudi and K. Shahanaghi
Department of Industrial Engineering, Iran University of Science and Technology, Tehran 1684613114, Iran.

Abstract

In this paper, we have proposed a new genetic algorithm for p-median location problem. In this regard, prior genetic algorithms were designed for p-median location problem by proposing several methods that are used in generation of initial population, crossover and mutation operators, and new operator so- called re-allocation has been incorporated into the algorithm that causes to find the optimal solution faster. Finally, efficiency of the algorithm has been shown by a numerical example.

Keywords: Genetic Algorithm, p-Median Location Problem, Multi Parent Crossover, Re-allocation.

1. Introduction

Genetic Algorithm (GA) is a meta-heuristic method which is used to obtain a near optimal solution in cases that numerical and mathematical methods are not able to solve the problem in reasonable time. GA generates a new population from the current population by using several operators, repeating these methods until a proper solution is obtained, and termination criteria are met. Each individual of the population, which is called a chromosome, is a solution for the investigated problem. Chromosomes are composed of genes.

GA has five steps: initial population generation, parent selection, crossover operation, mutation operation and new population selection. Initial population is generated randomly. Parents are selected randomly or the population is ordered decreasingly by fitness function value and then, parents are selected from top of the list. The crossover operator is performed by separating chromosome and replacing the separated parts. In mutation operation, a gene is selected randomly and changed [1]. To perform selection operator, generated offspring go to the next stage immediately or offspring along with their parents are ordered decreasingly according to fitness function value. Then the new population is selected from above the list to fill the population.

In this paper, we apply GA for solving p-median location problem and we improve it by applying some methods in each stage.

2. P-median location problem

In p-median location problem, we have several demand nodes. The aim of the problem is to choose p nodes to locate facilities in those nodes and allocate facilities to demand nodes. The objective is to minimize total transportation (distance is weighted by demand) between demand nodes and facilities.

The following assumptions are considered in the problem:

The capacity of each facility is unbounded. Each node can be supplied by just one facility, but each facility can service to several demand nodes. To formulate this problem, the following notations are used:

i and j : index of demand nodes;

h_i : demand in node i ;

d_{ij} : distance between demand nodes i and j ;

n : number of demand nodes;

p : number of facilities that be located;

Decision variables:

x_j : is equal to 1 if facility is located in node j , else is equal to 0;

y_{ij} : is equal to 1 if facility j is allocated to demand node i , else is equal to 0.

Therefore, the p-median location problem is formulated as integer linear programming (1): [2]

$$Min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} h_i y_{ij} d_{ij}$$

$$s.t:$$

$$\sum_{j=1}^{n} y_{ij} = 1 \quad \forall i \qquad (1)$$

$$y_{ij} \le x_j \quad \forall i, j$$

$$\sum_{j=1}^{n} x_j = p$$

$$x_i, y_{ij} \in \{0,1\} \quad \forall i, j$$

The first constraint represents that each demand node can be supplied by one facility. The second constraint ensures that a node is allocated to a demand node when a facility is located in that node. Third constraint indicates the number of facilities that must be located. The last constraint shows the type of each decision variable.

3. GA for solving p-median problem

In this section, we introduce the proposed genetic algorithm that is used to solve the p-median location problem.

3.1. Introducing chromosomes

Number of genes that form chromosomes of the proposed algorithm is equal to the number of demand nodes. Each gene is filled by the index of the node which is selected to locate a facility. These genes show the facility node to which the demand node is allocated. For example, consider 7 demand nodes. Figure 1 shows a chromosome.

Gene No.	1	2	3	4	5	6	7	8
	4	2	2	4	2	7	7	4

Figure 1. A sample chromosome of proposed algorithm

Genes of this chromosome indicate a facility location and the demand allocation. This chromosome shows that facilities are located at nodes 2, 4 and 7. Facility node 2 is allocated to nodes 2, 3 and 5, facility node 4 is allocated to nodes 1,4 and 8, and facility node 7 is allocated to nodes 6 and 7.

3.2. Initial population

To generate initial population, genes are usually generated randomly. In the proposed algorithm, to generate initial population, we randomly choose p nodes from demand nodes and locate facilities at those nodes. Then, each demand node is allocated to the nearest facility node. The generation code of initial population is as follows:

```
function ip = Initial Population (nop, nof)
% nop : number of nodes
% nof : number of facilities
% sof : Set of facilities
sof = generate nof random site nodes for facilities through demand nodes;
for i=1:nop
    if i is in sof
        ip(i) = i;
    else
        ip(i) = nearest facility to node i;
    end
end
```

Using this method, generated chromosomes certainly are feasible and so feasible test is required.

3.3. Fitness function

The fitness function is calculated using the objective function of the mathematical model.

3.4. Parent selection process

In parent selection process, parents are selected randomly from population.

3.5. Crossover operation

Crossover operator is used to generate children from parents. Based on parent selection procedure, we choose two chromosomes (parents). We can perform crossover operator in two ways: one point or two points (one of these ways is selected randomly). In one point crossover, one gene is selected randomly. To perform crossover operation, we use this gene and the second part that is created by it. In two-point crossover, two random genes are selected in the chromosome. To perform crossover operation, we use these genes and the middle part that is created by them. Figure 2 shows one point crossover and figure 3 shows two points crossover.

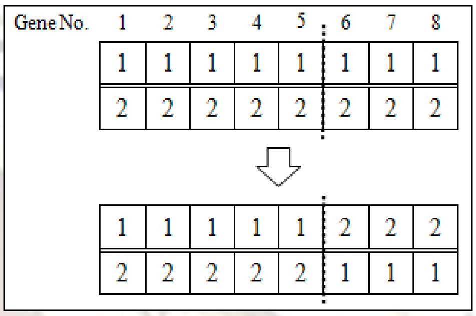

Figure 2. One point crossover

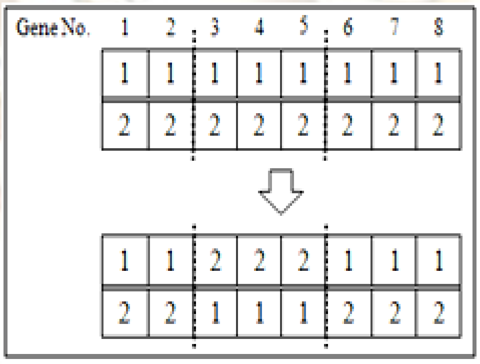

Figure 3. Two points crossover

In the proposed algorithm, to perform crossover operator, we use one of the above methods randomly for each pair of parents [3]. In order to increase the efficiency of the algorithm, after crossover operator, offspring are renewed. After crossover operator, the number of located facilities may not be equal to p. Thus, the crossover must be repeated. By renewing offspring, the number of located facilities will be equal to p. Therefore, generated chromosome is feasible. Children renewing code is following:

```
function nc = new child (c, sof, nof, nofs)
% c : child
% sof : set of facilities
% nofs : number of facilities in set
% nof : number of facilities
if nofs > nof
    for k=1:(nofs-nof)
        select randomly two sites and replace one
        of them with another;
    end
elseif nofs < nof
    for k=1:(nof-nofs)
```

```
                select randomly a facility site (fs) and a
                non-facility site (nfs). Then, set a facility in
                nfs and change allocations from fs to nfs
                alternately;
        end
else
        child c is ok;
end
```

In addition, in order to increase effectiveness of the crossover operator, in each performance of crossover, we run it several times (number of dates) and the best crossover is introduced as the main crossover operator.

In large scale problems, crossover operator might not show satisfactory efficiency. Therefore, we can perform it on several parents instead of two parents [4, 5]. In this paper, results of the crossover operator using three parents are six children. Figure 4 shows a two points crossover with three parents.

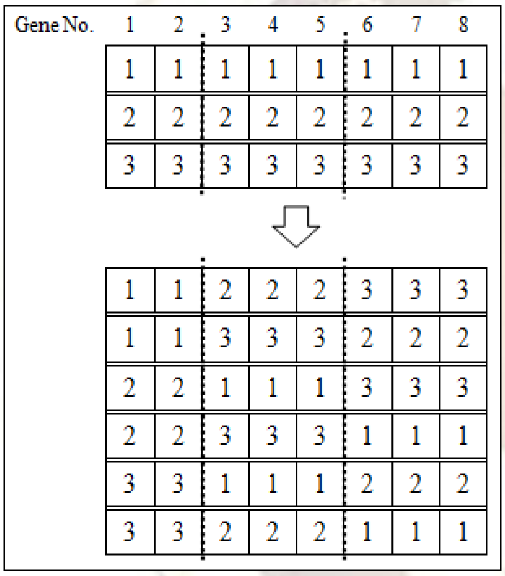

Figure 4. Two points crossover with three parents

3.6. Mutation operation

In mutation operator, one or more genes of a chromosome are selected and changed regarding the mutation rate. In the proposed algorithm, we use one of these three types of mutation operator, randomly: one point, two points and three points mutation which are performed on one, two and three gene respectively. These genes are replaced with one, two and three numbers that are points to one demand node. As an example, consider that 6th gene must be replaced with a facility which is located at node number 4. Figure 3 illustrates one point mutation operator.

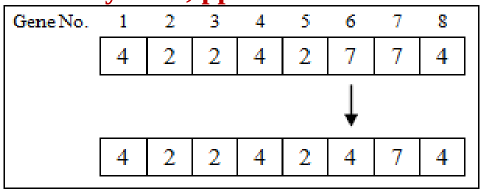

Figure 5. One point mutation operation

After performing mutation operator, generated children are renewed similar to the crossover operator. Therefore, the generated children are feasible.

3.7. Re-allocation

re-allocation operator which is performed on each individual, without changing facility's sites, allocations of demand nodes to facilities are changed. Each demand node is allocated to the nearest facility. Population resulting from this re-allocation is added to the current population. This operator causes a faster moving to the optimal solution. The last population (population after crossover and mutation operations and population after re-allocation operation) enters the next stage which is the selection of the new population.

3.8. Selection new population process

Old and new population (population after crossover and mutation operations and population after re-allocation operation) create a list of individuals. This list is sorted by fitness function decreasingly. Then, we choose certain number of chromosomes from the top of the list. These selected chromosomes are selected as the new population.

3.9. stopping criteria

To terminate the algorithm, two criteria must be satisfied:

a) Algorithm reaches the maximum number of iterations;

b) The best solution is not changed after a pre-specified number of generations.

Finally, chromosomes with the best fitness value are selected as optimal solutions. A numerical example is solved in the next section to show the efficiency of the algorithm.

4. Numerical example

To show the efficiency of the proposed algorithm, we use Galvao's standard data[1]. In Galvao's problem, there are 100 demand nodes. We want to locate p facilities to minimize total weighted distance (p-median location problem). We solved this problem using our proposed algorithm and a genetic algorithm proposed by Daskin's Sitation software[2] with p=5, 10, 15, 20, 25, 30, 35 and 40. Other parameters of this software are set as table 1.

Table 1. Parameters of GA in Daskin's software

Parameter	Value
Population Size	50
Number to Improve	0
Initial Population Size	150
Max # of Generation	300
# of Generation w/o Improvement	300
Min Compatibility	0.1
Number of Dates	8
Mutation Probability	0.4

Solutions of both algorithms are shown in table 2.

Table 2. Solution of both of proposed algorithm and Daskin's GA on Galvao's problem

N	P	Galvao Optimal	Proposed Algorithm	Daskin's Sitation Software
100	5	5703	5703	5703
100	10	4426	4440	4491
100	15	3893	3894	3974
100	20	3565	3565	3608
100	25	3291	3296	3312
100	30	3032	3039	3056
100	35	2784	2790	2815
100	40	2542	2545	2557

As table 1 shows, answers obtained by the proposed algorithm are closer to the optimal solution. In addition, proposed algorithm performs better than Daskin's Sitation software.

5. Conclusion

In this paper, after reviewing p-median location problem, we proposed new and efficient methods to improve GA for p-median location problem. These methods contain initial population; crossover operator containing one and two points crossover randomly, renewing children, and crossover with three parents. Mutation operator containing one, two and three points mutation randomly and renewing children and re-allocation operator. To show the efficiency of the algorithm, we solved a numerical example using the proposed algorithm. The results were compared to those obtained by genetic algorithm of Daskin's Sitation software that shows the superior performance of our proposed algorithm.

References

[1] D. Beasley, D. R. Bull, R. R. Martin, An Overview of Genetic Algorithms: Part 1, Fundamentals, *University Computing*, Vol. 15, No. 2, 1993, pp. 58-69.

[2] S. H. Owen, M. S. Daskin, Strategic facility location: A review, *European Journal of Operational Research*, Vol. 111, No. 3, 1998, pp. 423-447.

[3] A. Chipperfield, P. Fleming, H. Pohlheim, C. Fonseca, *Genetic Algorithm Toolbox User's Guide*, University of Sheffield, .

[4] A. E. Eiben, P. E. Raué, Zs. Ruttkay, Genetic algorithms with multi-parent recombination, *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, 1994, pp. 78-87.

[5] S. Tsutsui, M. Yamamura, T. Higuchi, Multi-parent Recombination in Genetic Algorithms with Search Space Boundary Extension by Mirroring, *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature*, 1998, pp. 428-437.