Çukurova Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 32(4), ss. 1-10, Aralık 2017
Çukurova University Journal of the Faculty of Engineering and Architecture, 32(4), pp. 1-10, December 2017

An Evolutionary Algorithm for p-Median Problem with Attribute Equity Constraint

Ertan YAKICI[*1], Aybeniz YİĞİT[2]

[1]*Deniz Harp Okulu, Milli Savunma Üniversitesi, İstanbul*
[2]*ASELSAN Araştırma Merkezi, Ankara*

Geliş tarihi: 25.01.2017 *Kabul tarihi: 19.12.2017*

Abstract

In this study, an evolutionary algorithm is proposed for solving the p-median problem with attribute equity constraint. The basic p-median problem aims to choose p facility locations out of n nodes and allocate the remaining demand nodes to the selected facility locations in order to minimize the total distance between demand nodes and assigned facilities. The problem studied in this paper has an extra constraint which keeps the maximum difference between total attributes of any pair of p clusters within a specified threshold. Attributes of nodes may represent problem dependent properties, like sales volume or population of districts. An evolutionary algorithm is developed to solve the problem. The algorithm is experimented with test problems found in the relevant literature and good results are obtained.

Keywords: Facility location, p-Median problem, Evolutionary algorithms

Değer Eşitliği Kısıtlı p-Medyan Problemi için Evrimsel Bir Algoritma

Öz

Bu çalışmada, talep noktalarının arz noktalarına adil biçimde atanmasını sağlayan ilave bir kısıtı ihtiva eden p-medyan probleminin çözümü için evrimsel bir algoritma önerilmiştir. Temel haliyle bir p-medyan problemi toplam n adet nokta içerisinden p adedini tesis yeri olarak seçerek geriye kalan talep noktalarından her birini tesislerden birine atarken, talep noktaları ile atandıkları tesis arasındaki toplam mesafeyi enazlamayı amaçlar. Bu makalede incelenen problem, aynı tesise atanan noktaların oluşturduğu p adet grup için hesaplanan grup değerleri arasındaki azami farkı belirlenmiş bir sınır içerisinde tutan ilave bir kısıta sahiptir. Bir grubun değeri, o grup içerisindeki tüm noktalar için belirlenmiş değerlerin toplamına eşittir ve bahsedilen değer satış hacmi, nüfus gibi özellikler olup problemden probleme farklılık gösterebilir. Söz konusu problemin çözümü için evrimsel bir algoritma geliştirilmiş, ilgili literatürden alınan test problemleri ile yapılan testlerde iyi çözümler alındığı tespit edilmiştir.

Anahtar Kelimeler: Tesis yerleşimi, p-Medyan problemi, Evrimsel algoritmalar

[*]Corresponding author (Sorumlu yazar): Ertan YAKICI, *eyakici@dho.edu.tr*

1. INTRODUCTION

The p-median problem (PMP) is the problem of choosing *p* sites out of *n* sites for establishing facilities to serve all of the sites while keeping the total distance between facilities and their customer sites minimum. Weighted distances can also be considered instead of only distances.

Location-allocation problems, like p-median problem, have been studied extensively with various constraints and considerations reflecting practical requirements. One of the mostly addressed constraints in the literature is the capacity limitation of suppliers, which converts the basic p-median problem to the capacitated p-median problem (CPMP). The problem studied in this paper is similar to CPMP, however there is a major difference between two problems. Attribute equity constraint forces assignment of nodes to facilities such that obtained cluster scheme has equitable attribute sharing. In order to reach an equitable attribute sharing, proposed model utilizes a pre-specified threshold that serves as a bound on the maximum difference found in total attributes of clusters.

Satisfying equitable attribute sharing among facilities can also be introduced to the basic p-median problem as a second objective, which turns the problem into a multi-objective one. However we have chosen introducing it as a constraint in our approach. As it is clear, with this approach, Pareto-optimal frontier can be obtained by solving our problem many times with different values for threshold.

Our motivation in this study is an operational problem which aims to position naval platforms to search certain potential points, using their helicopters, where enemy submarines can locate. The objective of the problem is to minimize the risk raised for anti-submarine warfare helicopters while keeping the total search time distributed among the platforms evenly allowing acceptable differences which is determined by decision maker. Beyond this specific case, there may be several application areas of this practical problem, both for commercial and governmental purposes.

Since the problem is NP-Hard, we propose a heuristic method, which is based on evolutionary techniques. Although researchers have used many different methods, evolutionary algorithms have not been applied frequently in solving the CPMP or Capacitated Clustering Problem (CCP), which allows different capacity limits from cluster to cluster. Evolutionary algorithms are proved to be efficient in solving difficult combinatorial problems as we analyze in this study.

Since our problem is not studied before, we review the CPMP and CCP literature, which is the most relevant one, with a focus on evolutionary metaheuristic techniques.

Maniezzo et al. [1] defined a Bionomic Algorithm for the CPMP. Bionomic Algorithms are evolutionary metaheuristic algorithms that update a whole set of solutions (a population of solutions) at each iteration. They differ from Genetic Algorithms and Evolution Strategies, because they explicitly direct the choice of the solutions to combine in order to define an offspring.

Shieh and May [2] applies a genetic algorithm to solve the capacitated clustering problem. For the 0-1 nature of CCP, this problem is coded as binary strings for genetic operating. Binary coding facilitates the evolutionary search with the standard steps of a genetic algorithm, i.e., modifying the genetic operators is unnecessary and infeasible solutions do not exist except for violation of the capacity constraint. An adaptive penalty function to handle the capacity constraint can be effectively applied to guide the search direction.

Lorena and Furtado [3] have introduced a new approach called the Constructive Genetic Algorithm (CGA), which allows schemata evaluation and other new features. Problems are modeled as bi-objective optimization problems that consider the evaluation of two fitness functions. This double fitness process evaluates schemata and structures in a common basis. Evaluation is executed considering an adaptive rejection threshold that takes both objectives into account and assigns a rank to each individual solution. The CGA is applied to both of the PMP and CPMP.

Correa et al. [4] have proposed a GA for the CPMP, and have applied it to a real-world problem with a quite large search space, with 421 billion feasible solutions. The GA uses an individual representation and genetic operators specifically developed for the PMP. The experiments show that the GA outperforms the tabu search algorithm.

Ghoseiri and Ghannadpour [5] proposed a genetic algorithm with a new consideration in assignment of the demand nodes. The demand points are assigned to the facilities considering urgencies to prioritize the demand point with higher urgency.

Resurreccion [6] extends the p-median problem to incorporate existing facilities. A GA based heuristic, along with a generation procedure based on opportunity cost, is proposed. The lost opportunity of not choosing the facility which is closest to the considered node is called opportunity cost. Although the proposed algorithm is a constructive method, the performance tests with the assumption of no existing facility show that good results that are close to best known solutions can be obtained.

In Section 2, we define the proposed problem with integer programming formulation. The heuristic method and the experiments are discussed in Sections 3 and 4, respectively. We give the conclusion of our study in Section 5.

2. PROBLEM DEFINITION

Required sets, indices, parameters and variables to formulate the p-median problem with attribute equity constraint and the formulation are given in this section. Since we assume all of the demand nodes are also median candidates, we have slightly modified the usual formulation of ReVelle and Swain [7].

$N = \{1,...,n\}$ is defined as the index set of sites to be allocated and also candidates of medians, where p facilities will be located. $[d_{ij}]_{nxn}$ and $[x_{ij}]_{nxn}$ are the distance and allocation matrices, where $x_{ij}=1$ if site i is allocated to median j, and $x_{ij}=0$, otherwise. The variable x_{jj} is set to 1, if median j is chosen, otherwise it is equal to 0. Parameter a_i is the attribute of each site.

The model given below defines the modified p-median problem discussed above.

$$Min\ \sum_{i\in N}\sum_{j\in N} d_{ij}x_{ij} \qquad (1)$$

subject to

$$\sum_{j\in N} x_{ij} = 1 \quad i\in N \qquad (2)$$

$$\sum_{j\in N} x_{jj} = p \qquad (3)$$

$$x_{ij} \leq x_{jj} \quad i\in N,\ j\in N \qquad (4)$$

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

$$\left|\sum_{i\in N} a_i x_{ik} - \sum_{i\in N} a_i x_{il}\right| \leq$$

$$max_{i\in N}(a_i) + \sum_{i\in N} a_i\,(2 - x_{kk} - x_{ll})$$

$$k,l : k\in N,\ l\in N, k<l \qquad (6)$$

Objective function (1) aims to minimize the total distance between medians and allocated demand sites. Constraint (2) impose that each entity is allocated to only one median. Constraint (3) ensures to select exactly p medians from n candidate sites. Constraint (4) imposes that an entity can be assigned to only a selected median and (5) provides the binary conditions. The constraint (6) aims to ensure that the maximum absolute difference in total attributes of any possible pair of clusters, is smaller than or equal to the maximum attribute value of the entities. Of course, any threshold value (right-hand-side of constraint (6)) can be used instead of maximum attribute value of the entities, as long as it does not violate the feasibility.

3. PROPOSED EVOLUTIONARY ALGORITHM

Initial medians are selected randomly in some studies like Mulvey and Beck [8] and Maniezzo et al. [1], whereas Osman and Christofides [9] generates initial medians by considering the distances between nodes, in order to have a good spread of medians. Their heuristic for choosing the initial set of medians includes finding the pair of nodes having the largest distance and assigning a median on each of these nodes. Then, their algorithm continues choosing the remaining (p-2) medians in such a way that the product of the distances to the chosen medians are tried to be maximized in each selection. We adopt a similar initial median selection approach like Osman and Christofides [9] with some differences. We choose the medians randomly from the nodes which are at least at a certain distance far from the previously chosen medians.

Mulvey and Beck [8] assigns the nodes to the medians in order to minimize the total node assignment regret. They give higher priority to allocation to the nodes having higher regrets. The assignment regret for a node is defined as the difference in contribution to objective function between the cases where the node is assigned first and second nearest medians. We adopt a similar regret approach. We give priority to the nodes having higher regrets, however, we modify the regret definition such that all the regrets for not assigning the node to its closest median is calculated for all medians and the allocation is performed according to the weighted regret of nodes.

We applied two improvement procedures: *intra-cluster improvement* and *swap improvement*. The first one evaluates the cases where each one of the nodes of a cluster is assigned as median, and changes the original median in order to have the maximum gain that is feasible. The second one, swap improvement procedure, involves swapping of nodes between clusters. These widely known local search methods are applied in several previous studies [1, 2, 8-11].

3.1. Parameters

The parameters used in the proposed evolutionary algorithm are as follows:

s: population size for evolutionary algorithm (EA),

p_s, p_c, p_a, p_w: parameters which control the forms of probability distribution functions used in selection procedure, construction of next generation and selection of the allocation method (either probabilistic or deterministic), and the form of the weight function, respectively,

n_s: the number of the same indexed solutions that is allowed to appear in the mating pool,

n_i: the number of the identical solutions that is allowed to appear in the mating pool,

n_a: the number of iterations executed between applications of swap improvement procedure.

3.2. Components of the Algorithm

The evolutionary algorithm first chooses p facilities, considering the distances between facilities. Then the other nodes are assigned to these facilities according to their weighted regret values. The regrets for each node, taking the base level as not being assigned to the nearest facility, are calculated for all facilities. While allocating the nodes to facilities, a probabilistic or a deterministic allocation method is chosen by the algorithm according to a selection parameter, p_a. Infeasible solutions are repaired in order to satisfy our attribute equity constraint. Two improvement procedures are applied; intra-cluster improvement and swap improvement procedures. The first one is applied to all generated solutions and gives chance to each node for becoming a median as long as it provides a gain and does not violate feasibility. The second improvement procedure is applied once in every n_a iterations. In order to prevent dominance of population by some solutions and maintain diversity, controls are applied before and after forming the mating pool. In the procedures for selection of parents and the construction of next generation, the fitness ranking method is employed, that is, we sort the individuals according to their raw fitness, then, assign reproductive trials according to their rank.

The flow diagram of the algorithm is given in Figure 1. The details of each component of the algorithm are explained in the following subsections. The termination of the algorithm is controlled by total number of iterations specified by the decision supporter.

Note that, throughout the algorithm run, the feasibility of solutions are maintained by calling a repair procedure which is used in two stages where infeasible solutions may be produced: generation of initial population and reproduction.

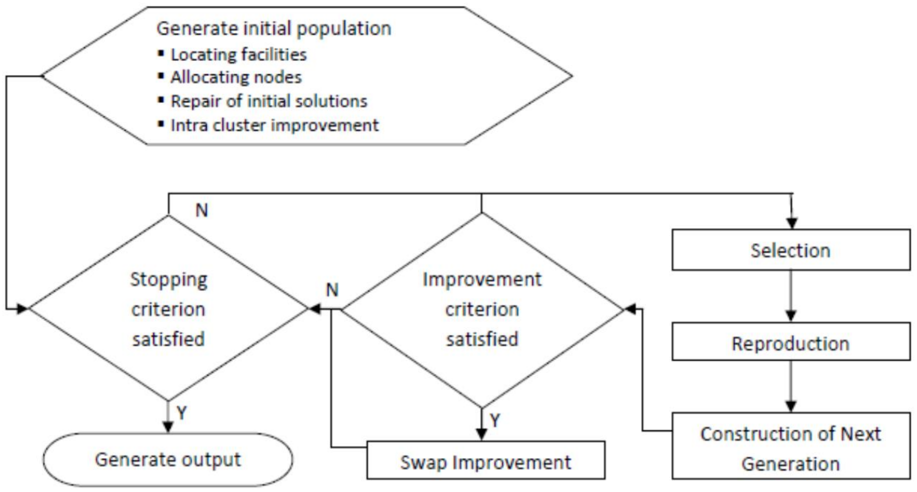

Figure 1. Flow diagram of the proposed algorithm

3.2.1. Location of Facilities for the Initial Population

Initially one node is chosen randomly out of n nodes for locating the first facility. After selecting the first facility randomly, in order to maintain a good dispersion of facilities, the next facility is chosen randomly from the remaining nodes such that it will be at least ($distance_{max} / p$) far from the first node, where $distance_{max}$ is the maximum distance between any pair of nodes in the considered problem instance. If there is not such a node, then ($distance_{max}$ / 2p) is tried. We continue until a proper node is chosen for locating the current facility. This procedure is applied until all of the p facilities are located.

Figure 2 illustrates an example of determining candidate locations for second facility in a 3-median problem with 10 nodes, where the first facility is located in the center of the circle. The candidate locations for the second facility are the nodes out of the circle.

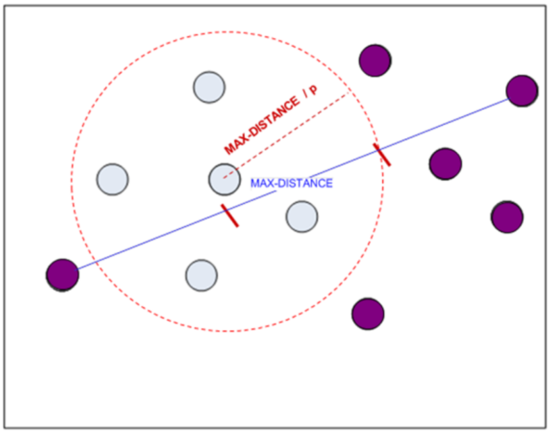

Figure 2. Determining candidate locations for the second facility

3.2.2. Allocation of Nodes to Facilities

We have two methods for allocation of nodes to facilities: probabilistic and deterministic. Both of the allocation methods use weighted regrets for determining allocation order of the nodes. Deterministic allocation method allocates the nodes to their closest facilities, whereas probabilistic allocation method performs allocation probabilistically by giving higher allocation probability for closer medians. Let node *i* be the considered demand node to be assigned to a facility in the set *J*, and an order index $P_j \in I^+$, where I^+ is the set of positive integer numbers, be given to each facility $j \in J$ with respect to distance between node *i* and facility *j*, starting with farthest. Then, weights of facility *j'* and regret for node *i* is calculated as follows:

$$w_{j'} = e^{\frac{P_{j'}}{p_w}} \Big/ \sum_{all\, j} e^{\frac{P_j}{p_w}} \qquad (7)$$

$$r_i = \sum_{j \in J} w_j \left(d_{ij} - d_{ij':P_{j'}=1} \right) \qquad (8)$$

where w_j indicates the weight of facility *j*.

The nodes, except the nodes where facilities are located, are ordered according to their regret values in a decreasing order, and they are allocated to the facilities starting with the first node. The parameter p_a determines the probability of allocation to be made whether probabilistically with probability p_a, or deterministically with remaining probability $1\text{-}p_a$. In performing allocations, if the total attribute of a cluster increased up to a level that is equal to or greater than $(\sum i\ a_i \ / \ p \ + \ max_i(a_i))$, then that facility is removed from consideration. The allocation weights are recalculated disregarding that facility and the allocation continues with the new weights, the remaining facilities and the remaining nodes that are not allocated yet. By this way, the repair of a solution is expected to be less costly.

3.2.3. Repair of Initial Solutions

It is checked if there is any pair of nodes which violate the attribute equity constraints. If there is any violation, the pair of clusters having greatest violation (i.e. the cluster with maximum total attribute and the cluster with minimum total attribute) is selected. In order to decrease the total attribute difference between these pair of clusters, the node last added to the cluster with maximum total attribute is reallocated to the other cluster having minimum total attribute. This procedure is applied until all violations are cleared.

3.2.4. Intra-Cluster Improvement Procedure

This procedure, which is a steepest descent algorithm, improves each cluster of a solution independently. In a cluster, each of the nodes is assigned as median, in order to check whether there is improvement or not. The node which contributes the most improvement is assigned as the new median. This procedure is applied to all of the members of population at each iteration. By this improvement procedure, every node is given chance to become a median as long as it brings improvement to the objective function. Figure 3 illustrates an application example for intra-cluster improvement procedure.

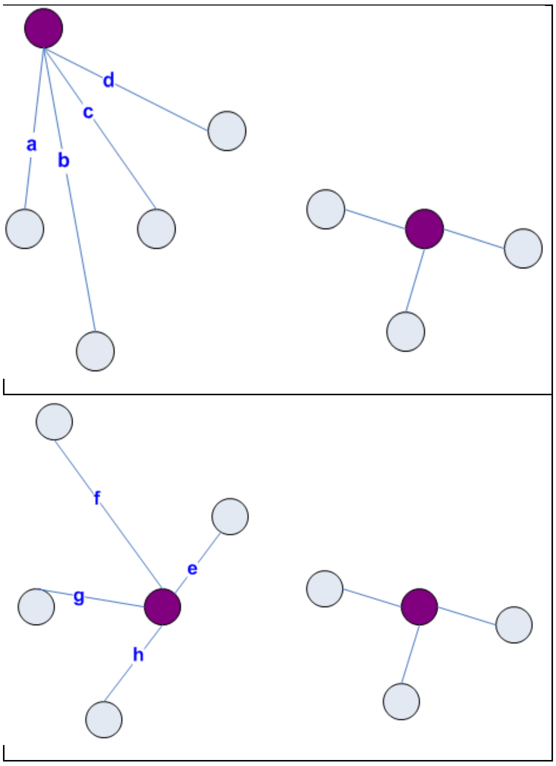

Figure 3. Effect of intra-cluster ımprovement procedure

3.2.5. Selection Procedure

Each solution l is given an order index $O_l \in I^+$ with respect to its objective function value starting with greatest, and is assigned a selection probability with respect to its position in the order. The individual having the minimum objective function value has the greatest chance of being selected.

Firstly, the solution with best objective function, is put into the mating pool. The other individuals are chosen with respect to a probability distribution. Let the order index number of solution l' be $O_{l'}$. Then, the probability of being chosen for a solution l', $Pr_{l'}$, is given as follows:

$$\mathrm{Pr}_{l'} = e^{\frac{O_{l'}}{p_s}} \Big/ \sum_{\mathrm{all}\ l} e^{\frac{O_l}{p_s}} \tag{9}$$

Note that a solution may exist more than once in the mating pool. However, we limit multiple existence by parameter n_s. Although we limit the multiple existence of a solution in the mating pool, after a number of generations, identical solutions may accumulate in the population. Therefore, in order to avoid a possible premature convergence, we put a limit on the number of solutions that are identical in the mating pool by a parameter, n_i. We change the solutions that are above allowed limit with new solutions produced by initial solution generation procedure. Note that, in checking identical solutions, only the objective function value is compared, assuming the probability of two different solutions' having the same objective function value in a real life problem is disregardable.

3.2.6. Reproduction Procedure

Two solutions are chosen randomly from the mating pool for reproduction, and the mating pool is updated by removing them. First, the best parent of two is selected and one of its facilities is chosen randomly to assign it as a facility in the child solution. The second facility is chosen from the other parent by using the probability distribution used in weighing facilities in allocation of nodes to facilities. Distant facilities from the first selected facility are favored in probabilistic selection (the highest weight is assigned to the most distant facility, and so on). In order to select the third facility we will consider again the best parent and the total distance of each of the remaining facilities from the selected facilities. Note that we switch the parents at each facility selection starting with the best parent. This procedure continues until selection of required number of facilities is completed. The remaining nodes which are not assigned as facilities are allocated according to the allocation subroutine as applied in the initial solution generation. Two offspring are generated from each pair of parent solutions in the mating pool. The repair procedure and the intra-cluster improvement procedure are called for the offspring.

3.2.7. Construction of Next Generation

Let a temporary set defined to be the union of the current generation and their children. The best solution from temporary set is selected and assigned as first solution of the next generation. The remaining selections are performed according to the same probabilistic selection method used in selection procedure utilizing the function form parameter p_c instead of p_s. However, unlike the selection procedure, we do not select the same individual more than once in order to keep the diversity of the population in the next generation. Whenever a solution is selected, it is removed from the temporary set and the probability distribution for the remaining individuals is recalculated.

3.2.8. Swap Improvement Procedure

With this improvement procedure, we check the solutions once in every certain number of iterations, n_a, in order to exploit any possible improvement. For a certain solution, all possible swaps of nodes (other than facilities) between clusters, without violating the attribute constraint, define the solution's neighborhood. This procedure works as first improvement algorithm: which swaps the nodes between clusters at the first time it encounters an improvement and continues with the next node. After scanning all of the nodes, the intra-cluster improvement procedure is called.

3. EXPERIMENTS

Due to the problem's computational complexity, for only small-size problems a good solution can be obtained by exact solution algorithms in a reasonable period of time. Two small-size instances, 3-median with 15 nodes and 2-median with 50 nodes, are randomly generated and solved by CPLEX solver (12.6.2.0) using given IP formulation in a PC with 1.9 GHz processor and 4 GB RAM. We have observed that our algorithm finds the optimum value for both of the instances in less than 4 seconds, while CPLEX finds optimal solutions in 3.5 and 40 minutes, respectively.

The parameters p_s, p_w and p_c, which are used during selection of parents for the mating pool, reproduction and construction of next generations, respectively, should be fine-tuned in order to obtain a good balance between diversification and intensification. On the other hand, the p_a parameter which affects the allocation method, should also be assigned carefully. Therefore, we have tried to tune these parameters for the problems having size of 50 and 100 nodes with 5 and 10 facilities respectively. 10 random instances are generated and evaluated for each problem size. We have performed a full factorial analysis for the mentioned parameters. For each setting, each problem instance is solved for 10 times. The average of the 10 objective function value for each problem is taken as the response variable. The levels of the parameters for full factorial experiments, which are determined experimentally, and suggested parameter settings are presented in Table 1.

Table 1. Parameter levels and the ANOVA results

Parameter	Low Level of the Parameter	High Level of the Parameter	Suggested Levels by ANOVA For n=50/p=5	Suggested Levels by ANOVA For n=100/p=10
p_s (p_c=2p_s)	p_s=10 p_c=20	p_s=25 p_c=50	Low Level	Low Level
p_w	1	2	High Level	High Level
p_a	0.1	0.3	High Level	Low Level

After tuning the problem parameters, the performance of the algorithm is compared using the OR-Library CPMP problems [12]. Note that, optimal solution for the CPMP is not necessarily the optimal solution for our problem; however, it serves as a lower bound for our problem unless the maximum total attribute of our solution exceeds the capacity of the CPMP problem. In order to make our problem comparable to CPMP instances, we adjusted the threshold value, which is RHS of the constraint (6). That is, we increased this value in order to release RHS and allow, when needed, the maximum total attribute value reach the capacity of the facilities in CPMP problem. Therefore, any solution with maximum total attribute exceeding the capacity of the CPMPs, which is equal to 120, is disregarded.

In preliminary trial experiments, we have observed that we obtain better results when their maximum total attribute values are between 110 and 120. Therefore, we adjusted the threshold value and obtained 5 solutions with maximum total attribute value between 110 and 120 for each test problem. These obtained solutions are considered for evaluating the performance of our algorithm. The results are presented in Table 2. First 10 instances have 50 nodes and 5 medians, while the remaining have 100 nodes and 10 medians.

Iterations are terminated at 1000th generation and n_a is set to 10, which determines the frequency of swap improvement application. Population size is set to 20, while the parameters n_s and n_i are set to 2 and 4 respectively. It is observed that the EA rarely injected new solutions to the mating pool with this setting.

Table 2. The results of performance tests

Problem No	Best solution for CPMP (a practical benchmark)	Best EA solution	Average elapsed time in seconds	Percentage deviation of best result from the lower bound	Percentage deviation of average result from the lower bound	Percentage deviation of worst result from the lower bound
1	713	739	34.6	3.65	4.01	5.47
2	740	756	32.9	2.16	2.16	2.16
3	751	773	32.2	2.93	3.28	4.66
4	651	665	34.6	2.15	2.24	2.46
5	664	681	34.5	2.56	2.56	2.56
6	778	797	34.3	2.44	2.44	2.44
7	787	808	34.8	2.67	2.90	3.05
8	820	838	37.2	2.20	2.68	3.66
9	715	731	36.4	2.24	2.41	2.66
10	829	847	38.6	0.97	1.30	2.65
Average Percentage Deviation				**2.40**	**2.60**	**3.18**
11	1006	1055	80.3	4.87	7.26	10.83
12	966	998	77.9	3.31	7.54	10.35
13	1026	1067	68	4.00	6.96	11.99
14	982	1032	86.4	5.09	11.12	15.99
15	1091	1138	83.4	4.31	7.15	12.65
16	954	985	70.8	3.25	5.39	8.60
17	1034	1059	79	2.42	3.95	10.06
18	1043	1079	81.4	3.45	5.66	9.49
19	1031	1076	75.7	4.36	9.49	14.06
20	1005	1069	91.23	6.37	9.53	11.64
Average Percentage Deviation				**4.14**	**7.40**	**11.57**

Although the benchmarks obtained by CPMP do not serve as optimal values for our problem, it is seen that the results of our algorithm are very close to these practical lower bounds. The average of the best solutions deviates from these benchmarks by 2,40% and by 4,14% for 50-node problems and for 100-node problems, respectively.

4. CONCLUSION

This study applies an evolutionary algorithm to the p-median problem having attribute equity constraint. Since we do not have the optimal solutions for large instances, our algorithm is tested with CPMP test instances. The optimum values of these problems serve as practical lower bounds for our problem. In order to tune the parameters of the evolutionary algorithm, a factorial analysis is performed. The result of the experiments with OR-Library suggests that the proposed technique can produce good solutions.

Finally, we must note that, the proposed algorithm can be adapted to a multi-dimensional attribute case with small modifications.

5. REFERENCES

1. Maniezzo, V., Mingozzi, A., Baldacci, R., 1998. A Bionomic Approach to the Capacitated p-Median Problem, Journal of Heuristics 4(3), 263-280.
2. Shieh, H.M., May, M.D., 2001. Solving the Capacitated Clustering Problem with Genetic Algorithms, Journal of the Chinese Institute of Industrial Engineers 18(3), 1-12.

3. Lorena, L.A.N., Furtado, J.C., 2001. Constructive Genetic Algorithm for Clustering Problems, Evolutionary Computation 9(3), 309-327.

4. Correa, E.S., Steiner, M.T.A., Freitas, A.A., Carnieri, C., 2004. A Genetic Algorithm for Solving a Capacitated p-Median Problem, Numerical Algorithms 35, 373-388.

5. Ghoseiri K., Ghannadpour S.F., 2007. Solving Capacitated p-Median Problem using Genetic Algorithm, Proceeding of IEEE International Conference on Industrial Engineering and Engineering Management, 885-889.

6. Resurreccion J., 2006. An Opportunity Cost-based Genetic Algorithm for a Modified Capacitated p-Median Problem, Philippine Engineering Journal 27(2), 1-26.

7. ReVelle, C.S., Swain R., 1970. Central Facilities Location, Geographical Analysis 2(1), 30-42.

8. Mulvey, J.M., Beck, M.P., 1984. Solving Capacitated Clustering Problems, European Journal of Operational Research 18(3), 339-348.

9. Osman, I.H., Christofides, N., 1994. Capacitated Clustering Problems by Hybrid Simulated Annealing and Tabu Search, International Transactions in Operational Research 1(3), 317-336.

10. Lorena, L.A.N., Senne, E.L.F., 2003. Local Search Heuristics for Capacitated p-Median Problems, Networks and Spatial Economics 3(4), 407-419.

11. Franca, P.M., Sosa, N.M., Pureza, V., 1999. An Adaptive Tabu Search Algorithm for the Capacitated Clustering Problem, International Transactions in Operational Research 6(6), 665-678.

12. Beasley, J.E., 1990. OR-Library: Distributing Test Problems by Electronic Mail, Journal of the Operational Research Society 41(11), 1990, 1069-1072.