Vladimir Medvid *

TWO EFFICIENT ALGORITHMS FOR WEIGHTED *p*-MEDIAN PROBLEM

We propose a genetic algorithm for a weighted p-median problem. It is a facility location problem. The algorithm generates a good solution quickly. Computational tests were realized on five different tasks from 21 vertices to 100 vertices and from p-median from p=3 to p=6. The tests were performed 100 times on every task. There were created some modifications of these tasks for a proposed genetic algorithm.

The best solution generated by this algorithm is within 0.6% of the optimum for 80% of the tasks. The other 20% of the tasks is within 1.6% of the optimum. Time of realization is within 5.9 s.

Keywords: *Model, linear programing, p-median problem, optimal solution, arithmetic mean.*

1. Introduction

In this paper, we propose two different algorithms for a facility location problem which is well-known as a p-median problem. This is a combinatorial optimization problem well-known to be NP-hard [1], [2] and [3]

The goal of the problem is to select locations of p facilities to serve n demand points so as to minimize the total travel between the facilities and the demand points. This is a combinatorial optimization problem known as an NP-hard problem.

The first of the proposed algorithms is a classical deterministic exchanged algorithm and the second is a genetic algorithm.

Genetic algorithms are heuristic search methods that are designed to solve some optimization problems by the use of mimicking the evolution process. New solutions are produced from old solutions in ways that are reminiscent of the interaction of genes. Genetic algorithms have been applied successfully to problems with very complex objective functions.

Our genetic algorithm is presented in some modification ways to find the best solution as far as possible.

2. Model of the problem

There are given the following items:
p is a number of centers placed
I is a set of customers
J is a set of candidates for the center placements
c_i is the price of i-th customer
d_{ij} is a distance from the-th customer to the -th center.

Decision variables

x_j=1 if candidate is used, 0 otherwise
y_{ij}=1 if the requirement is fulfilled by-thcenter, 0 otherwise.

Objective function

$$\min \sum_{i \in I} \sum_{j \in J} c_i d_{ij} y_{ij} \tag{1}$$

subject to

$$\sum_{j \in J} y_{ij} = 1 \tag{2}$$

$$\sum_{j \in J} x_j = p \tag{3}$$

$$y_{ij} \leq x_j, i \in I, j \in J \tag{4}$$

The first condition means that each customer will be served by just one center. The second condition implies p built service centers. The third condition ensures that the customer cannot be operated where the center will not be built. The task is therefore to deploy some habitat service centers so that the sum of the distances from each customer to designated centers is minimal.

* **Vladimir Medvid**
Department of Mathematics, Faculty of Humanities, University of Zilina, Slovakia
E-mail: vladimir.medvid@fhv.uniza.sk

3. Classical deterministic replaceable algorithm

The following strategy is used in the proposed algorithm.

At first it selects vertices with the greatest costs and assigns the closest vertices to the chosen ones. Then a vertex is chosen when the product of its cost and the distance is worst. It becomes a new candidate for covering while the original vertex moves temporarily to the back up. Then we state new assignment and compare it to the original. The better of these two is chosen, the worse is neglected. The algorithm then selects a new vertex with the worst assignment and the process repeats until the set of the vertices with the 'worst' assignment is empty.

This algorithm is published in [4] where is better explained together with the demonstration examples.

4. Genetic algorithm

First, we create an initial population of solutions. Consequently, in the executing phase good properties of these solutions are combined to obtain better solutions. If the algorithm is not able to find a better solution after the predetermined time then the cycle stops.

Construction

1. Read n, h, p, $d(v_i, v_j)$, $c = (c_1, c_2, ..., c_n)$
   where is a number of vertices, is a number of edges, is a number of medians, is a length of each edge and is the evaluation of vertices.
2. Construct a matrix of the shortest distances $D_{(n\times n)}$ where an element in i-th row and in j-th column represents the shortest distance between i-th and j-th vertices. Obviously, the matrix is symmetrical and the elements on the leading diagonal are zeroes.
   Construct a matrix of evaluated edges $ID(n\times n)$ where the edges are represented by incident vertices v_i and v_j, i.e. $d(v_i, v_j)\neq 0$ if $[v_i, v_j]$ is the edge of the graph and $d(v_i, v_j) = 0$ otherwise.
3. Enumerate the number of solutions in the initial population by the formula:

$$k = \max\left(2, \left[\frac{n}{100}\cdot\frac{\binom{n}{p}}{\left[\frac{n}{p}\right]}\right]\right)\cdot\left[\frac{n}{p}\right] \text{[5]}. \qquad (5)$$

4. Create an initial population P of feasible solutions. The initial population P represents the matrix of k rows and p columns where every row represents an individual, i.e. candidates (covered vertices) for some feasible solutions of the problem.
5. Define the objective function and assign values of the objective function to each individual in the population as follows: in i-th row of the matrix P allocate for the candidates the other vertices according to the shortest distances and calculate

$$v(u_{ij}) = \sum_{l=1}^{t} c_l \cdot d_{ij,l} \qquad (6)$$

where is the candidate of i-th row and j-th column of the matrix P and $d_{ij,l}$ is the shortest distance from the l-th vertex to the nearest candidate u_{ij} and is the number of the vertices allocated to the u_{ij} candidate and $j = 1, 2, \ldots, p$.
Subsequently, compute

$$f_i = \sum_{j=1}^{p} v(u_{ij}) \qquad (7)$$

where $j = 1, 2, \ldots, p$, $i = 1, 2, \ldots, k$.
A vector $f = (f_1, f_2, \ldots, f_k)$ represents the evaluation of the solutions of the initial population.
Determine the maximum number of iterations to perform a cycle since the last iteration which found a better solution according to the equation $m = [n \cdot \sqrt{p}]$.
Put $r = 0$, r is a variable that represents the number of iterations. (In the following point the executing phase begins where the cycle is initiated.)
6. *while* $r \leq m$
   Here the application of genetic algorithm follows.
   Randomly select two parents $\alpha_r, \alpha_s \in P$. One parent is represented by one individual, i.e. $\alpha_r = (u_{r1}, u_{r2}, ..., u_{rp})$ and similarly $\alpha_s = (u_{s1}, u_{s2}, ..., u_{sp})$. If the parents have some common gene (some common candidate) then this gene has entered inthe chain offspring. Remaining genes consequently create one set.
   $\aleph = \{\alpha, \alpha \subset \alpha_r \cup \alpha_s\}$. Consequently, genes are gradually excluded. The exclusion takes place according to which gene has the smallest value $v(u_{ij})$ as long as the number of genes drops down to p. This way, it creates offspring β that inherits properties of their parents. It may be that the offspring inherits the genes only from one parent as these genes are more dominant than the other parent's genes.
7. Replacement. Calculate $f(\beta)$.
   If $f(\beta) < \max f = f_l$ then replace the worst l-th individual by β offspring.
   $P_l := \beta$, $f_l := f(\beta)$ and put $r = 0$.
   If $f(\beta) \geq \max f = f_l$ then reject the new offspring from the process and put $r := r + 1$.
8. Go to 7 and repeat the process until a condition of termination is satisfied.
9. After the completion of the cycle we often obtain a population in which individuals are very similar and even identical. We declare the best individual as the best solution of the p-median problem and the corresponding value of objective function as the best rate of availability of covered vertices.

4.1 Modifications

First modification

First, we tried to set a criterion which stated which genes will be inherited by the individual and which will not. The best variant was to select the best combination of genes, but this process is possible only with smaller tasks because the evaluation of each combination is very time consuming. In the end, we decided for a variant in which each vertex of the graph is assigned to a gene according to the shortest distance and the individual will inherit those genes which are assigned by most vertices. This method is not time consuming and offers better results than a genetic operator in the initial algorithm.

Second modification

We found out that the process of algorithm also influences the selection of the primary population which is randomly generated at the beginning. For this reason we decided for an alternative where the primary population is selected according to a certain system, not randomly, so that a gene is covered at least twice.

Third modification

While observing the algorithm, we noticed that in the process of substitution there were situations when a "good" gene was contained in an overall weak chromosome and so it was substituted by a better offspring. This caused that genes which could have been parts of the optimal solution were disappearing from the population. To prevent this from happening, we doubled the population where one half of the population, which contained all possible genes in sufficient amount, was not substituted, but kept unchanged until the end of algorithm. The other half is then step by step substituted by better offsprings. In this way, any individual from the population can enter the genetic operator by the process of random selection.

Fourth modification

The last modification can be described as the "arrival of immigrants" into population. Those were developed independently, which increases the biodiversity. In case they are crossed with other individuals from the population, unexpectedly able individuals can be created. In our case it means that the population is broadened by two randomly selected individuals and in each tenth generation, these two are changed for a new pair of randomly selected individuals.

The algorithm of any subsequent modification includes features of previous modifications.

5. Results and evaluations

Algorithms were tested in various graphs with various ranges, and we knew the optimal solutions. The results are presented in separate tables.

5.1 Results of classical deterministic replaceable algorithm

In Table 1, the first column represents the number of vertices of the graph and the number of medians, i.e. the number of vertices which must be covered. The second column represents the optimal and exact solution. The third column represents the solution found by this algorithm. The fourth column represents the time it took the algorithm to find the solution. The last, fifth column, shows the percentage difference between the exact solution and the solution found by this algorithm.

Table 1

Set	Optimum	Found solution	Computing time	Difference
21v, 3p	50	50	0.001s	0.00%
50v, 5p	6624	6656	0.02s	0.48%
50v, 6p	5696	5867	0.02s	3.00%
100v, 5p	32500	32500	0.06s	0.00%
100v, 6p	29577	30274	0.06s	2.36%

5.2 Results of the genetic algorithm

Table 2 shows the results of our initial genetic algorithm. This algorithm was realized 100 times for each set of entry data. Because the algorithm works with random processes, it gives different results for entry data. The column Number of finding the optimum shows how many times the algorithm was able to find the optimum (exact) solution in 100 trials. The column Average solution shows the arithmetic average of all 100 achieved results.

Table 2

Set	Size of population	Opti-mum	The best found solution	Number of finding the optimum	Average time	Average solution	Difference
21v, 3p	30	50	50	100	0.03s	50	0.00%, 0.00%
50v, 5p	44	6624	6656	0	0.03s	7575	0.48%, 14.35%
50v, 6p	36	5696	5807	0	0.03s	6727	1.94%, 18.11%
100v, 5p	84	32500	33276	0	0.13s	36049	2.38%, 10.92%
100v, 6p	68	29577	29658	0	0.12s	33265	0.27%, 12.47%

The last column Difference shows two percentage results. The first value shows the difference between the optimum solution and the best solution found by the algorithm. The other value represents the percentage difference of the optimum and the arithmetic average. Other columns are obvious.

It is obvious from the results in the table that the algorithm works fast but the results are not reliable. It was able to find the exact solution only with the simple practice task.

Table 3 represents the results after *the first modification*.

Table 3

Set	Size of population	Opti-mum	The best found solution	Number of finding the optimum	Average time	Average solution	Difference
21v, 3p	30	50	50	41	0.01s	53	0.00%, 6.80%
50v, 5p	44	6624	6656	12	0.06s	6861	0.00%, 3.57%
50v, 6p	36	5696	5807	0	0.06s	5971	1.01%, 4.82%
100v, 5p	84	32500	33276	0	0.24s	33672	1.00%, 3.61%
100v, 6p	68	29577	29658	0	0.25s	31279	1.01%, 5.75%

We changed the criterion of the selection of inherited genes and the result shows an obvious improvement.

Table 4 represents the results after *the second modification*.

Table 4

Set	Size of population	Optimum	The best found solution	Number of finding the optimum	Average time	Average solution	Difference
21v, 3p	30	50	50	81	0.01s	51	0.00%, 1.22%
50v, 5p	44	6624	6663	0	0.06s	6952	1.00%, 4.95%
50v, 6p	36	5696	5771	0	0.05s	6111	1.01%, 7.30%
100v, 5p	84	32500	32564	0	0.24s	33650	1.00%, 3.54%
100v, 6p	68	29577	29727	0	0.26s	31384	1.00%, 6.11%

This modification of creating the primary population on the bases of certain system brought improvement only in the practice task and one other task.

Table 5 represents the results after *the third modification.*

Table 5

Set	Size of population	Optimum	The best found solution	Number of finding the optimum	Average time	Average solution	Difference
21v, 3p	30	50	50	100	0.03s	50	0.00%, 0.00%
50v, 5p	44	6624	6624	5	0.13s	6685	0.00%, 0.92%
50v, 6p	36	5696	5768	0	0.11s	5821	1.01%, 2.19%
100v, 5p	84	32500	32500	26	0.57s	33024	0.00%, 1.61%
100v, 6p	68	29577	29658	0	0.54s	30103	1.00%, 1.78%

After the third modification where we kept the primary population, we can see a great improvement. The best solutions were achieved with accuracy up to 1% in two cases and in the other three cases we reached the optimum. The average solutions were up to 2.2%.

Table 6 represents results after *the fourth modification* [6].

Table 6

Set	Size of population	Optimum	The best found solution	Number of finding the optimum	Average time	Average solution	Difference
21v, 3p	30	50	50	100	0.38s	50	0.00%, 0.00%
50v, 5p	44	6624	6624	58	1.76s	6642	0.00%, 0.27%
50v, 6p	36	5696	5696	19	1.45s	5764	0.00%, 1.19%
100v, 5p	84	32500	32500	68	5.87s	33683	0.00%, 0.56%
100v, 6p	68	29577	29577	36	5.35s	29719	0.00%, 0.48%

6. Conclusion

This last modification, which includes "the immigrants", brought very nice results. Even though after this modification the algorithm used roughly 10 times more time compared to other methods, it achieved far better results. Out of 100 trials, the algorithm was able to find the optimum many times. The last column shows a perfect stability when only in one "the worst" case it achieved the average less than 1.2 %. In other sets it was less than 0.6%.

This algorithm was able to deal with a large set of 900 vertices and 90 medians in average in 62.9 seconds. Unfortunately, we were not to prove the optimum value, but on the bases of the previous results it is very probable that the results were very close to the optimum.

The results of a deterministic replaceable algorithm are not the worst. It is a deterministic algorithm and it gives only one solution for each set of entry data and depending on the nature of this data we can find a situation where the solution will be far from the optimum and, therefore, it is not reliable.

Acknowledgement

This work has been supported by the research grant VEGA 1/0296/12.

References

[1] BUZNA, L., KOHANI, M., JANACEK, J.: Proportionally Fairer Public Service Systems Design, *Communications - Scientific Letters of the University of Zilina*, vol. 15, No. 1, 2013, pp. 14-18.

[2] JANACEK, J., GABRISOVA, L.: Lagrangean Relaxation Based Approximate Approach to the Capacitated Location Problem, *Communications - Scientific Letters of the University of Zilina*, vol. 8, No. 3, 2006, pp. 19-24.

[3] JANACEK, J., JANOSIKOVA, L.: Computability of the Emergency Service System Design Problem, *Communications - Scientific Letters of the University of Zilina*, vol. 10, No. 2, 2008, 2013, pp. 5-9.

[4] MEDVID, V.: *A Proposal of Algorithm for Solving -median Problem,* Proc. of 21th annual conference Technical computing Prague 2013 [electronic source] Prague: Institute of chemical technology, 2013 - CD-ROM [7].

[5] ALP, O., ERCUT, E.: An Efficient Algorithm for the -Median Problem, *Annals of Operations Research*, 122, 2003, pp. 21-42.

[6] HERDA, M.: *Construction of Some Models of Linear Programming Problems*, Thesis, *Faculty of Humanities: University of Zilina*, 2014, pp. 39-46.