Parameter Adjustment for Genetic Algorithm for Two-Level Hierarchical Covering Location Problem

MIROSLAV MARIC, MILAN TUBA, JOZEF KRATICA
Faculty of Mathematics
University of Belgrade
Studentski trg 16, 11000 Belgrade
SERBIA
maricm@matf.bg.ac.yu tubamilan@ptt.rs http://www.matf.bg.ac.yu/~maricm

Abstract: - In this paper the two-level Hierarchical Covering Location Problem - HCLP is considered. A new genetic algorithm for that problem is developed, including specific binary encoding with the new crossover and mutation operators that keep the feasibility of individuals. Modification that resolves the problem of frozen bits in genetic code is proposed and tested. Version of fine-grained tournament [5] was used as well as the caching GA technique [12] in order to improve computational performance. Genetic algorithm was tested and its parameters were adjusted on number of test examples and it performed well and proved robust in all cases. Results were verified by CPLEX.

Key-Words: Genetic Algorithms, Evolutionary computing, Location problem, Hierarchical location, Covering models

1 Introduction

The multitude of practical applications is a major reason for the great interest in network-based facility location modeling. Location models are application specific, and their structural form (the objectives, constraints, variables) is determined by a particular problem under study. There does not exist a general location model that is appropriate for all existing and potential discrete location problems. Much of the literature on the facility location modeling has been directed to formulating new and flexible location models that will be adequate for different applications, and to developing efficient solution techniques for solving more general models.

Covering location models are often used in applications related to the location of emergency facilities. A demand area is covered if it is within a predefined service distance from at least one of the existing facilities. A primary objective in siting service facilities is to "cover" as much of the potential customer demand as possible. Many types of location models have been developed using covering objectives. One of the most important models is the maximal covering location problem (MCLP). A node is "covered" if there exists a facility within a pre-specified coverage radius. The objective of the MCLP is to locate a fixed number of facilities so as to maximize the total coverage. We refer the readers to [7] for a discussion on the problem and its application.

1.1 Covering Models

Maximum covering models are suitable for siting desirable facilities. However, some facilities such as garbage dumps, nuclear reactors and prisons are "undesirable" or "obnoxious". Although these "undesirable" facilities provide service to the society, they may have an adverse effect on the people living nearby. In such instances, maximin or maxisum objectives may be appropriate. For the single facility case, the maximin objective is to find the location of the undesirable facility such that the least (weighted) distance to all nodes is maximized, while the maxisum objective is to site a facility so as to maximize the (weighted) sum of the distances from the facility to all customers located at the nodes of the network. For the multiple facilities case, there are many well-motivated problems, depending on how one defines the objective function. For example, one version of maximin problems is the p-dispersion problem, in which there are *p* facilities to be located on the network so that the minimum distance between any two facilities is as large as possible. In the maxisum dispersion problem, the objective is to maximize the summation of all distances between the *p* facilities. The p-dispersion problem and the maxisum dispersion problem on general networks are both NP-hard. We refer the readers to [7] for a comprehensive literature review of undesirable facility location problems.

The problem of locating undesirable facilities on a networks employing a coverage type objective function is often considered in practice. In the literature that problem is called as the minimum covering location problem with distance constraints (MCLPDC). Through locating a fixed number of facilities, intention is to minimize the number of covered customers (where a customer is considered covered if her distance to the closest facility is less than a pre-determined radius).

When the number of facilities is greater than one, minimal distance constraints are imposed to prevent all facilities to be located at the same point. To motivate the MCLPDC consider the problem of locating nuclear reactors. Nuclear reactors may pose a serious danger to the individuals living nearby. The fewer people "covered", the better. Sometimes, for sensitivity and safety reasons, they should also be separated (e.g., if several reactors are clustered in the same region, they may all be attacked by an aggressor). Separation between nuclear reactors can be constrained by a pre-specified minimum distance.

A related problem is the expropriation location problem (ELP), which was introduced by Berman et al. [1]. Each demand point is associated with a given expropriation price. Demand points within a pre-specified distance from a facility will be expropriated. The ELP seeks the location set of a fixed number of obnoxious facilities such that the total cost of expropriation is minimized. Berman et al. studied the Minimum Covering Location problem on the plane on a network and presented an algorithm to solve the problem. The sensitivity of the coverage radius was also analyzed. They also investigated the Minimum Covering Location problem which they call Problem 2 of ELP on a network and generalized the search for the optimal solution to a dominant set of points. They defined all demand nodes whose weighted distance from the facility is less than a pre-specified radius as covered. All previous approaches only considered the problem of locating a single facility. Therefore, there was no need to incorporate distance constraints between facilities.

Of particular relevance in the practice is the location set covering problem (LSCP), where the objective of the LSCP is to find the minimum number of facilities that cover all customer demand. If there is a cost associated with each demand node, the objective of the LSCP with variable weights minimizes the total cost of siting facilities so as to cover the whole network. As mentioned before, facilities are not always completely desirable. In some cases this fact is recognized and basic LSCP is extended to include distance constraints, which restrict facilities to be no closer than some specified distances from demand nodes. It is finally shown that this model has the same structure as the LSCP.

1.2 Hierarchical Models

The hierarchical models may be important in some situations, for example in the location of schools and health care facilities. For example, the lower level facilities - clinics provide only a level one service, whereas the higher level facilities - hospitals provide both (level one and level two) services. This hierarchy is successively inclusive in the sense that a facility provides its own level of service and all lower levels of service. Other example of the HCLP is higher education systems which consist of technical schools and universities (universities can cover both academic and applied studies). Third example, the production - distribution systems may consist of factories and warehouses, where a given product can be shipped to a client directly from the factory or through one of the warehouses. As can be seen from previous examples, two levels of hierarchy are usually enough for practical applications.

In the present paper the 2-level hierarchical extension of the maximal covering problem is considered. This problem is NP-hard as a generalization of the well-known p-median problem.

2 Problem Formulation

We use the integer linear programming formulation of the HCLP from [6]. Let R_1 be the service distance for level 1 service provided by the lower level facility. This service distance could be considered to be equal for the same type of service provided by the higher level facility, but in real life people may be prepared to travel an extra distance to obtain the same service from a facility with more resources. So T_1, the service distance for level 1 service provided at the higher level facility, is supposed to satisfy $T_1 > R_1$. On the other hand, let R_2 be the distance for level 2 service. This type of service is offered only by the higher level facilities and in practice people will be prepared to travel longer distances to obtain the more sophisticated level 2 services. Therefore, in the HCLP model we will consider that $R_2 > T_1 > R_1$. The mathematical programming formulation for the hierarchical covering location problem is given below:

$$\max\left\{\sum_{j\in J} f_j x_j\right\}, \tag{1}$$

s.t.

$$\sum_{i\in I} a_{ij} y_i + \sum_{i\in I} b_{ij} z_i - x_j \geq 0,\ j \in J, \tag{2}$$

$$\sum_{i\in I} c_{ij} z_i - x_j \geq 0,\ j \in J, \tag{3}$$

$$\sum_{i\in I} y_i = p, \tag{4}$$

$$\sum_{i\in I} z_i = q, \tag{5}$$

$$x_j \in \{0,1\}, j \in J, \qquad (6)$$

$$y_i, z_i \in \{0,1\}, i \in I . \qquad (7)$$

Where:

$J = \{1, 2, \ldots, m\}$ is the set of demand areas,

$I = \{1, 2, \ldots, n\}$ is the set of potential facility sites,

f_j is the population of demand area j,

$a_{ij} = 1$ if demand area j can be covered by level 1 service (within distance R_1) offered at a lower level facility located at $i \in I$ ($a_{ij} = 0$ otherwise),

$b_{ij} = 1$ if demand area j can be covered by level 1 service (within distance T_1) offered at a higher level facility located at $i \in I$ ($b_{ij} = 0$ otherwise),

$c_{ij} = 1$ if demand area j can be covered by level 2 service (within distance R_2) offered at a higher level facility located at $i \in I$ ($c_{ij} = 0$ otherwise),

p is the number of lower level facilities to be located,

q is the number of higher level facilities to be located

x_j , y_i and z_i are the decision variables:

$x_j = 1$ if demand area j is covered ($x_j = 0$ otherwise);

$y_i = 1$ means that a lower level facility is located at site $i \in I$ ($y_i = 0$ otherwise),

$z_i = 1$ means that a higher level facility is located at site $i \in I$ ($z_i = 0$ otherwise).

In the formulation above the objective function (1), to be maximized, represents the total population covered by both level 1 and level 2 services. Constraints (2) state that a demand area $j \in J$ is covered by level 1 service if there is at least either one lower level facility within distance R_1 or one higher level facility within distance T_1. Constraints (3) state that a demand area $j \in J$ is covered by level 2 service if there is at least one higher level facility within distance R_2. Constraint (4) limits the number of the lower level facilities in the solution to p; whereas constraint (5) limits the number of the higher level facilities in the solution to q. Finally, constraints (6)–(7) define the 0–1 nature of the decision variables.

This general problem formulation was used for adjustment of the problem for the CPLEX program when comparing results obtained by our genetic algorithm with the exact solutions obtained by CPLEX.

In this paper we consider a more specialized case where potential facility sites are the same locations where demand areas are. This approach is more realistic in many cases, because hospitals or fire stations, for example, are normally located inside populated areas. So we used $J = I = \{1,2, \ldots, n\}$.

3 Problem Solution

Genetic Algorithms (GAs) are robust and adaptive methods that can be used to solve search and optimization problems. They represent problem-solving metaheuristic method rooted in the mechanisms of evolution and natural genetics. The main idea was introduced by Holland, and in the last three decades GAs have emerged as effective, robust optimization and search methods.

GAs solve problems by creating a population of individuals (usually 10 - 200), represented by chromosomes which are encoded solutions of the problem. The representation is the genetic code of an individual and it is often a binary string, although other alphabets of higher cardinality can be used. A chromosome is composed of basic units named genes, which control the features of an individual. To each chromosome a fitness value measuring its success is assigned. The initial population (the first generation of individuals) is usually randomly initialized, although in some situation, the population may be fully or partially generated by some initial heuristic. The second approach usually has problems with reduced diversibility of genetic material. It can produce better solutions in several starting generations, but later it gives worse results. The individuals in the population then pass through a procedure of simulated "evolution" by means of randomized processes of selection, crossover, and mutation.

The selection operator favors better individuals to survive through the generations. The probability that a chromosome will be chosen depends on its fitness. The higher fitness value of a chromosome provides higher chances for its survival and reproduction. There are different ways of selecting the best-fitted individuals. One of the most often used is tournament selection. Crossover and mutation operators are also used during reproduction. The crossover operator provides a recombination of genetic material by exchanging portions between the parents with the chance that good solutions can generate even better ones.

Mutation causes sporadic and random changes by modifying individual's genetic material with some small probability. Its role is to regenerate the lost or unexplored genetic material into the population. Mutation has a direct analogy with nature and it should prevent premature convergence of the GA to suboptimal solutions. Multiple usage of selection and crossover (without mutation) results in loosing of genes variety and some regions of search space are not reachable. This usually causes the premature convergence in local optimum far from global optimal value. The mutation is basic mechanism for restoring lost genes into the population. This increases the diversibility of genetic material and previously not reachable regions of search space may be reachable again.

There are different policies for generation replacement. In every generation of SGA entire population is replaced with new individuals through selection, crossover and mutation. This variant is named generational GA and ensures maximal gradient in genetic search. Unfortunately, it does not have an absolute mechanism for preserving the excellent individuals from unlucky applying of some genetic operator. If the good solution is destroyed, it has to be reexplored again by genetic search, but the running time is wasted. Therefore some number of individuals may skip selection or even all genetic operators going directly into the next generation. In the case of steady-state replacement with elitist strategy a part of population skips all genetic operators and their objective values are evaluated only in the first generation. In all subsequent generations they are directly proceeded and reevaluation is not necessary. Since the objective value function is usually most computationally expensive part of GA, the elite individuals are obtained very cheaply. On the other side, important individuals or genes can be preserved by this policy. This approach is named the steady-state generation replacement policy with elitist strategy. It provides a smaller gradient in the genetic search, but preserves good individuals from the past generations.

There can be many modifications of the GA, but implementing the GA usually involves the following steps:

- Evaluating the fitness of all individuals in a population.
- Selecting the best-fitted individuals.
- Creating a new population by performing crossover and mutation operators.

The process of reproduction and population replacement is repeated until a stopping criterion (fixed number of generations or satisfied quality of solutions obtained) is met.

The genetic algorithm approach is widely used for solving various combinatorial optimization problems, which include location problems such as: Simple Plant Location Problem, Index Selection Problem, Dynamic Facility Layout Problem, etc.

GAs are also used for solving some other hub location problems: Uncapacitated Multiple Allocation p-Hub Median Problem-UMApHMP, Uncapacitated Single Allocation Hub Location Problem-USAHLP, Uncapacitated Multiple Allocation p-Hub Center Problem-UMApHCP. These problems have similar names as our problem, but up to now known solution approaches for solving these problems are substantially different. For example, different allocation schemes in UMApHMP and USApHMP have great impact on the problem complexity. For the fixed set of hubs, the multiple allocation sub-problem is solved in polynomial $O(n^{2p})$ time, while the single allocation sub-problem remains NP-hard. Therefore, proposed genetic algorithms for solving these problems have quite different natures.

Extensive computational experience on various optimization problems shows that GA often produces high quality solutions in a reasonable time. Some of recent applications are:

- hub location [6, 19 – 22],
- biconnectivity augmentation [14, 15],
- multidimensional knapsack [17,18],
- graph coloring [10],
- low-autocorrelation binary sequence[11],
- optimal power flow [2],
- cooperative control [3],
- bluetooth positioning [8],
- multimodal function optimization [9],
- decision support [13].

Moreover, GA has shown to be robust with respect to parameter choice on quite different problems [6,10,11,14,15,17-22]. In most cases GA has shown to be robust with respect to parameter choice in reasonable bounds.

3.1 GA for HCLP

The basic scheme of this GA implementation can be represented as a standard GA outline:

```
Input_Data();
Population_Init();

while not Finish() do
        for i:=1 to $N_{pop}$ do
            $p_i$ := Objective_Function($ind_i$);
        endfor

    Fitness_Function();
    Selection();
    Crossover();
    Mutation();
endwhile

Output_Data();
```

N_{pop} denotes the number of individuals in a population, p_i is the objective value of the i-th individual (ind_i).

3.2 Representation

The binary encoding is used for solving the HCLP, so the feasible solution of GA is represented by a 2n-dimensional binary vector. On odd positions are encoded level 1 facilities and on even positions level 2 facilities are encoded. Each bit with value 1 in the genetic code denotes that a certain facility is established, while 0 denotes that it is not. By such encoding y_i and z_i are defined so x_j can be computed by using constraints (2) and (3). Therefore, all constraints except (4) and (5) are satisfied by default.

In order to obtain more correct individuals in the initial population, the probability of generating ones in the genetic code is set to p/n on odd positions and q/n on even positions. The individuals which have a number of ones in their genetic code that is different from p on odd positions (denoted as k_p, $k_p \neq p$) are corrected by adding/erasing $| p - k_p |$ ones at/from the end of the genetic code on odd positions. Similarly, the same procedure is performed on even positions.

After the correction procedure, constraints (4) and (5) are satisfied, because the number of established facilities on level 1 is fixed to p and the number of established facilities on level 2 is fixed to q (all individuals become feasible). The described correction is performed only in the first generation, since the applied genetic operators are designed to preserve the feasibility of individuals.

The appearance of infeasible individuals was a limiting factor for using binary encoding. This difficulty was overcome in this GA implementation successfully (as we described above), and it can also be seen from the computational results.

3.3 Genetic Operators

GA implementation experimented with tournament and fine-grained tournament selection – FGTS (described in [5,6]). The FGTS depends on a real parameter F_{tour} – the desired average tournament size that takes real values. Actually, the average tournament size should be as close as possible to F_{tour}. It is realized by using two types of tournaments. During one generation, tournaments are held with different number of competitors. The first tournament type is held k_1 times and its size is $\lceil F_{tour} \rceil$. The second type is performed k_2 times with $\lfloor F_{tour} \rfloor$ individuals participating ($\lfloor x \rfloor = r$ and $\lceil x \rceil = s \Leftrightarrow r \leq x \leq s$ and $r,s \in Z$, $x \in R$) that implies $F_{tour} \approx k_1 * \lfloor F_{tour} \rfloor + k_2 * \lceil F_{tour} \rceil$. Running time for the FGTS operator is $O(F_{tour})$. In practice F_{tour} is considered to be constant (not depending on the problem size), that gives a constant time complexity.

The crossover operator is applied on a selected pair of parents producing two offspring. The standard crossover usually randomly chooses crossover points and simply exchanges the segments of the parents' genetic codes. The previous approach cannot be applied in our GA implementation, since it may produce incorrect offspring for the HCLP. The number of ones in an offspring may become different from *p* on odd positions, although its parents had exactly *p* ones on odd positions in their genetic codes. To overcome this problem, the basic crossover is modified in GA. The operator is simultaneously tracing the genetic codes of the parents from right to left searching the odd position *i* on which the first parent has 1 and second 0 (Figure 1.).

```
parent1:001100101011⟶  001100101011  ⟶
parent2:011110100001    011110100001
                        →j        i←


⟶ 011100110001  ⟶ 011100110001    ⟶
   001110100011     001110100011
                     →j i←


⟶   011110100001 offspring1
     001100110011 offspring2
```

Fig. 1 Modified crossover operator

The individuals exchange bits on the found position (identified as crossover point), and a similar process is performed starting from the left side of the genetic codes. The operator is searching the odd position *j* where the first parent has 0 and the other 1. Bits are exchanged on the *j*-th position and the number of located facilities on level 1 in each individual remains unchanged. The described process (step) is repeated until $j \geq i$. Same procedure is applied for bits on even positions, i.e. for facilities on level 2.

During the GA execution it may happen that (almost) all individuals in the population have the same bit at a certain position. These bits are called frozen. If the number of frozen bits is *l*, the search space becomes 2^l times smaller and the possibility of premature convergence rapidly increases. Selection and crossover operator cannot change the bit value of any frozen bit and the basic mutation rate is often insufficiently small to restore the lost sub-regions of the search space. If the basic mutation rate is increased significantly, the genetic algorithm becomes a random search. For this reason, the mutation rate is increased only on frozen bits not more than a few times.

In GA implementation the modified simple mutation operator with frozen bits is applied to offspring generated by the crossover operator. The mutation operator is performed by changing a randomly selected bit in the genetic code (0 to 1, 1 to 0). Applied mutation rates are constant through generations of the GA. Note that, in order to preserve the feasibility of individuals, it is necessary to count and compare the numbers of mutated ones and zeros for each individual. In case that these numbers are not equal, we have to mutate additional bits in order to equalize them. In this way, the mutation operator is preserving *p*,*q* ones in the genetic code, and keeps the mutated individual feasible.

3.4 Caching GA

The running time of the GA is improved by caching. Evaluated objective functions are stored in a cache memory, together with the corresponding genetic codes. When the same genetic code is obtained again during the GA, the objective value is taken from the cache memory, instead of computing the objective function.

For caching GA we use free system memory and allocate it dynamically. Cache memory is divided into the blocks of equal size. Every cache memory block saves information about one individual from population and contains following data: genetic code, his objective value and indicator of individual’s validity.

In this implementation we use hash-queue structure of pointers to all cache memory blocks for providing following operations:

- Searching cache memory by hash table for a particular block which contains given individual, if such block exists,
- Removing the oldest block from cache memory (by queue) if it is necessary,
- Putting current individual into the cache memory, instead of the removed block, if cache memory previously does not contain that individual.

Queue of pointers to cache memory blocks is ordered by information about the last access to a particular block. On the top of the queue is placed the newest accessed block, following by blocks accessed before it, and finishing by the oldest cache memory block.

We perform one level caching hierarchy, which is simple to implement and avoids problems about cache consistency. The Least Recently Used (LRU) strategy is used for caching GA. The number of cached function values is limited to 5000 in this implementation.

3.5 Caching algorithm

We perform caching to the Objective_Function() on the schematic form of GA given in previous section. Instead of that function, we have program segment:

```
if ($ind_i$ in cache_memory)
then
        $p_i$ := Get_From_Cache($ind_i$);
else
        $p_i$ :=Objective_Function($ind_i$);
        if (Cache_is_Full())
        then
                Remove_From_Cache(Oldest_Block());
                Put_In_Cache($ind_i$);
        endif
endif
Newest($ind_i$);
```

First if-statement searches the cache memory for a given individual. It returns pointer to the cache memory block that contains a given individual, or information that given string does not exist in the cache memory (in that case it returns NULL pointer).

If particular individual is found in the cache memory, we set current individual objective value from founded cache memory block, instead of computing an objective value. After that given cache memory block is marked as the newest used. In the other case, if cache block that contains a current individual is not found, the Objective_Function(ind_i) is called to compute its objective value. If cache memory is full, the oldest block is discarded from cache memory. Afterward, we put data of the current individual (genetic code, objective value and validity) to free cache memory block, instead of previously discarded block. Finally, given cache memory block is marked as a newest. Detailed description of caching GA technique can be found in [12].

3.6 Other GA aspects

The population consists of 150 individuals and in the first generation the initial population is randomly generated. This approach provides maximal diversity of genetic material and better gradient of objective function. A steady-state generation replacement with elitist strategy is used. In this replacement scheme different number (1 to one half) of best individuals are directly passing in the next generation preserving highly fitted bits. The elite individuals do not need recalculation of objective value since each of them is evaluated in one of the previous generations.

Duplicated individuals are removed from each generation. Their fitness values are set to zero, so that selection operator prevents them from entering the next generation. This is a very effective method for saving the diversity of genetic material and keeping the algorithm away from premature convergence. Individuals with the same objective function but different genetic codes in some cases may dominate in the population. If their codes are similar, the GA can lead to a local optimum. For that reason, it is useful to limit their appearance to some constant. In this GA application this constant is set to 40.

4 Experimental results

In order to validate our GA solutions we used CPLEX on integer linear programming model from [4] described in section 2. The optimality of GA results on instances given below is verified by CPLEX. Since, both methods (GA and CPLEX) quickly converges on mentioned HCLP instances (less than 0.1 second), their running times are not reported.

For the initial testing of our GA implementation we generated two HCLP instances, both of the size 20. To get the initial idea of the algorithm behavior we used Euclidean 2-D distance. The first example was generated to include 20 random locations on the square of the size 70. The second example was used to test more irregular situation where 20 locations on the same square are randomly located in three loosely defined neighborhoods. Distances were then calculated for both examples as Euclidian 2-D distances.

4.1 Example 1

For n = 20, $R_1 = 10$, $T_1 = 12$, $R_2 = 22$;

Population f = [32 36 17 11 15 38 35 36 17 21 13 39 23 33 34 36 16 20 21 15]

and the distance matrix (random locations) that is given in the Table 1, we get the following results:

Case 1

For parameters p = 3, q = 3:

Optimal solution value is 355.

At lower level potential facilities 8, 13 and 15 are established.

At higher level potential facilities 2, 7 and 14 are established.

Demand areas 1, 2, 5, 6, 7, 8, 10, 13, 14, 15, 16 and 17 are covered.

The solution was found in the 8th generation.

Case 2

For parameters p = 6, q = 5:

Optimal solution value is 497.

At lower level potential facilities 2, 3, 8, 13, 18 and 19 are established.

At higher level potential facilities 5, 7, 11, 16 and 17 are established.

All demand areas except location 4 are covered.

The solution was found in the 54th generation.

$$d = \begin{bmatrix}
0 & 6 & 51 & 66 & 14 & 30 & 41 & 58 & 23 & 42 & 71 & 65 & 43 & 46 & 62 & 51 & 67 & 84 & 58 & 71 \\
6 & 0 & 45 & 60 & 10 & 24 & 35 & 52 & 18 & 36 & 64 & 59 & 40 & 41 & 56 & 47 & 62 & 78 & 56 & 66 \\
51 & 45 & 0 & 15 & 45 & 24 & 16 & 14 & 41 & 27 & 27 & 30 & 62 & 47 & 42 & 58 & 52 & 55 & 78 & 62 \\
66 & 60 & 15 & 0 & 59 & 38 & 28 & 13 & 54 & 36 & 18 & 27 & 72 & 54 & 43 & 65 & 53 & 50 & 86 & 64 \\
14 & 10 & 45 & 59 & 0 & 21 & 32 & 49 & 9 & 29 & 60 & 53 & 30 & 32 & 48 & 37 & 53 & 71 & 47 & 56 \\
30 & 24 & 24 & 38 & 21 & 0 & 11 & 28 & 17 & 15 & 41 & 36 & 40 & 30 & 37 & 40 & 45 & 57 & 57 & 52 \\
41 & 35 & 16 & 28 & 32 & 11 & 0 & 17 & 26 & 12 & 29 & 25 & 46 & 31 & 31 & 42 & 40 & 49 & 62 & 49 \\
58 & 52 & 14 & 13 & 49 & 28 & 17 & 0 & 43 & 23 & 14 & 16 & 59 & 41 & 31 & 52 & 41 & 41 & 73 & 51 \\
23 & 18 & 41 & 54 & 9 & 17 & 26 & 43 & 0 & 21 & 53 & 45 & 25 & 23 & 39 & 29 & 44 & 62 & 42 & 48 \\
42 & 36 & 27 & 36 & 29 & 15 & 12 & 23 & 21 & 0 & 32 & 24 & 36 & 20 & 22 & 31 & 30 & 43 & 51 & 38 \\
71 & 64 & 27 & 18 & 60 & 41 & 29 & 14 & 53 & 32 & 0 & 12 & 65 & 46 & 30 & 56 & 39 & 32 & 78 & 49 \\
65 & 59 & 30 & 27 & 53 & 36 & 25 & 16 & 45 & 24 & 12 & 0 & 54 & 35 & 18 & 44 & 27 & 25 & 66 & 38 \\
43 & 40 & 62 & 72 & 30 & 40 & 46 & 59 & 25 & 36 & 65 & 54 & 0 & 19 & 40 & 14 & 39 & 61 & 17 & 36 \\
46 & 41 & 47 & 54 & 32 & 30 & 31 & 41 & 23 & 20 & 46 & 35 & 19 & 0 & 21 & 11 & 22 & 43 & 32 & 25 \\
62 & 56 & 42 & 43 & 48 & 37 & 31 & 31 & 39 & 22 & 30 & 18 & 40 & 21 & 0 & 28 & 10 & 23 & 49 & 21 \\
51 & 47 & 58 & 65 & 37 & 40 & 42 & 52 & 29 & 31 & 56 & 44 & 14 & 11 & 28 & 0 & 25 & 47 & 22 & 22 \\
67 & 62 & 52 & 53 & 53 & 45 & 40 & 41 & 44 & 30 & 39 & 27 & 39 & 22 & 10 & 25 & 0 & 22 & 45 & 11 \\
84 & 78 & 55 & 50 & 71 & 57 & 49 & 41 & 62 & 43 & 32 & 25 & 61 & 43 & 23 & 47 & 22 & 0 & 66 & 29 \\
58 & 56 & 78 & 86 & 47 & 57 & 62 & 73 & 42 & 51 & 78 & 66 & 17 & 32 & 49 & 22 & 45 & 66 & 0 & 38 \\
71 & 66 & 62 & 64 & 56 & 52 & 49 & 51 & 48 & 38 & 49 & 38 & 36 & 25 & 21 & 22 & 11 & 29 & 38 & 0
\end{bmatrix}$$

Table 1: Distance Matrix for Example 1

Case 3

For parameters p = 7, q = 5:

Optimal solution value is 508.

At lower level potential facilities 1, 3, 4, 8, 13, 18 and 19 are established.

At higher level potential facilities 5, 7, 11, 16 and 17 are established.

All demand areas are covered.

The solution was found in the 840th generation.

From these cases we can see that the algorithm behaves well in the average as well as in the border cases. It always finds the optimal solution and the number of generation necessary to find the optimal solution increases with the complexity of the problem, but is always very reasonable.

4.2 Example 2

For n = 20, $R_1 = 10$, $T_1 = 12$, $R_2 = 22$;

Population f = [32 36 17 11 15 38 35 36 17 21 13 39 23 33 34 36 16 20 21 15]

and the distance matrix (three neighborhoods) given the in Table 2:

$$d = \begin{bmatrix}
0 & 12 & 13 & 19 & 23 & 26 & 63 & 56 & 62 & 55 & 60 & 47 & 52 & 51 & 62 & 69 & 60 & 63 & 65 & 75 \\
12 & 0 & 10 & 15 & 12 & 19 & 60 & 50 & 56 & 45 & 52 & 35 & 40 & 39 & 52 & 60 & 48 & 52 & 54 & 64 \\
13 & 10 & 0 & 7 & 14 & 13 & 51 & 44 & 49 & 42 & 47 & 41 & 44 & 40 & 50 & 56 & 52 & 53 & 54 & 64 \\
19 & 15 & 7 & 0 & 14 & 8 & 45 & 37 & 43 & 36 & 40 & 40 & 42 & 37 & 44 & 50 & 50 & 50 & 49 & 59 \\
23 & 12 & 14 & 14 & 0 & 12 & 52 & 40 & 47 & 34 & 41 & 27 & 30 & 27 & 40 & 49 & 38 & 40 & 42 & 51 \\
26 & 19 & 13 & 8 & 12 & 0 & 41 & 31 & 37 & 29 & 33 & 35 & 36 & 30 & 37 & 42 & 44 & 43 & 41 & 52 \\
63 & 60 & 51 & 45 & 52 & 41 & 0 & 17 & 9 & 32 & 24 & 67 & 62 & 52 & 38 & 29 & 68 & 59 & 48 & 58 \\
56 & 50 & 44 & 37 & 40 & 31 & 17 & 0 & 9 & 15 & 8 & 51 & 45 & 35 & 21 & 16 & 51 & 42 & 32 & 42 \\
62 & 56 & 49 & 43 & 47 & 37 & 9 & 9 & 0 & 23 & 15 & 59 & 54 & 44 & 28 & 20 & 59 & 50 & 39 & 49 \\
55 & 45 & 42 & 36 & 34 & 29 & 32 & 15 & 23 & 0 & 9 & 37 & 31 & 21 & 9 & 15 & 36 & 28 & 18 & 29 \\
60 & 52 & 47 & 40 & 41 & 33 & 24 & 8 & 15 & 9 & 0 & 47 & 40 & 30 & 14 & 9 & 45 & 36 & 24 & 34 \\
47 & 35 & 41 & 40 & 27 & 35 & 67 & 51 & 59 & 37 & 47 & 0 & 9 & 17 & 38 & 51 & 14 & 23 & 33 & 37 \\
52 & 40 & 44 & 42 & 30 & 36 & 62 & 45 & 54 & 31 & 40 & 9 & 0 & 10 & 30 & 43 & 8 & 13 & 24 & 28 \\
51 & 39 & 40 & 37 & 27 & 30 & 52 & 35 & 44 & 21 & 30 & 17 & 10 & 0 & 21 & 34 & 16 & 13 & 17 & 25 \\
62 & 52 & 50 & 44 & 40 & 37 & 38 & 21 & 28 & 9 & 14 & 38 & 30 & 21 & 0 & 13 & 34 & 23 & 11 & 21 \\
69 & 60 & 56 & 50 & 49 & 42 & 29 & 16 & 20 & 15 & 9 & 51 & 43 & 34 & 13 & 0 & 47 & 36 & 23 & 30 \\
60 & 48 & 52 & 50 & 38 & 44 & 68 & 51 & 59 & 36 & 45 & 14 & 8 & 16 & 34 & 47 & 0 & 12 & 25 & 26 \\
63 & 52 & 53 & 50 & 40 & 43 & 59 & 42 & 50 & 28 & 36 & 23 & 13 & 13 & 23 & 36 & 12 & 0 & 13 & 14 \\
65 & 54 & 54 & 49 & 42 & 41 & 48 & 32 & 39 & 18 & 24 & 33 & 24 & 17 & 11 & 23 & 25 & 13 & 0 & 11 \\
75 & 64 & 64 & 59 & 51 & 52 & 58 & 42 & 49 & 29 & 34 & 37 & 28 & 25 & 21 & 30 & 26 & 14 & 11 & 0
\end{bmatrix}$$

Table 2: Distance Matrix for Example 2

we get the following results:

Case 1

For parameters p = 3, q = 3:
Optimal solution value is 441.
At lower level potential facilities 4, 11 and 13 are established.
At higher level potential facilities 2, 15 and 17 are established.
All demand areas except 7, 9 and 20 are covered.
The solution was found in the 9th generation.

Case 2

For parameters p = 6, q = 3:
Optimal solution value is 493.
At lower level potential facilities 6, 13, 15, 16, 18 and 19 are established.
At higher level potential facilities 2, 9 and 14 are established.
All demand areas except location 20 are covered.
The solution was found in the 4th generation.

Case 3

Even though locations are grouped in the three neighborhoods, location 20 is a bit too far to be covered in the third neighborhood with the radius R_2 = 22. Introduction of the 4th hospital (level 2 facility) reduces the need for clinics (level 1 facility) to 3.

For parameters p = 3, q = 4:
Optimal solution value is 508.
At lower level potential facilities 4, 11 and 13 are established.
At higher level potential facilities 2, 9, 17 and 19 are established.
All demand areas are covered.
The solution was found in the 10th generation.

4.3 Parameter Sensitivity

Different parameters were adjusted during testing and robustness of the algorithm was examined.

In the previous examples initial population was set to 150. However, the algorithm performed well with smaller initial population, increasing the number of generations to reach the optimal solution:

Initial population:	20	50	150	400	
Generations:	414	58	8	6	Ex 1, Case1
	259	91	54	45	Ex 1, Case2
	1558	745	840	363	Ex 1, Case3

Elitist strategy was used and again the algorithm proved to be robust to selection of this parameter. Any number between 1 individual and the two thirds of the population was acceptable, but better results were obtained with number of elitist units fixed at 10% of the whole population.

Number of elitist:	1	10%	30%	70%	
Generations:	8	7	4	12	Ex 1, Case1
	54	48	55	37	Ex 1, Case2
	840	23	65	60	Ex 1, Case3

4.4 Example 3

This example represents a case with larger number of locations:

For n = 50, R_1 = 10, T_1 = 12, R_2 = 22;

and the distance matrix that can be downloaded from http://www.matf.bg.ac.yu/~maricm/instances/hclp/hclp50.txt good behavior of the algorithm is again established:

Case 1

For parameters p = 4, q = 4:

Optimal solution value is 737.

At lower level potential facilities 15, 16, 42 and 45 are established.

At higher level potential facilities 13, 26, 33 and 40 are established.

Demand areas 2, 3, 4, 7, 12, 13, 15, 16, 17, 18, 19, 20, 25, 26, 29, 30, 31, 32, 33, 36, 38, 40, 41, 42, 45, 46, 48 and 50 are covered.

The solution was found in the 35th generation.

Case 2

For parameters p = 9, q = 14:

Optimal solution value is 1257.

At lower level potential facilities 6, 11, 12, 24, 26, 35, 36, 39 and 43 are established.

At higher level potential facilities 1, 2, 3, 8, 9, 13, 15, 27, 28, 34, 37, 38, 41 and 47 are established.

All demand areas are covered.

The solution was found in the 157th generation.

4.5 Example 4

The algorithm was also tested on a non-Euclidean distance matrix which represents the most general form of this problem.

For n = 100, R_1 = 10, T_1 = 12, R_2 = 22;

and the distance matrix that can be downloaded from http://www.matf.bg.ac.yu/~maricm/instances/hclp/hclp100.txt performance faster then the CPLEX was noted:

Case 1

For parameters p = 5, q = 6:

Optimal solution value is 1892.

At lower level potential facilities are 3, 21, 46, 81 and 86 are established.

At higher level potential facilities 4, 10, 48, 10, 93 and 100 are established.

Demand areas 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 16, 17, 18, 20, 22, 24, 25, 26, 27, 28, 29, 31, 32, 33, 34, 35, 36, 37, 39, 40, 42, 43, 44, 45, 46, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 68, 69, 70, 71, 72, 73, 74, 75, 76, 78, 79, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 93, 94, 95, 97, 98, 99 and100 are covered.

The solution was found in the 228th generation.

Case 2

For parameters p = 9, q = 12:

Optimal solution value is 2195.

At lower level potential facilities 7, 23, 25, 26, 27, 33, 45, 87 and 95 are established.

At higher level potential facilities 2, 13, 14, 22, 28, 41, 48, 52, 54, 55, 86 and 90 are established.

All demand areas are covered.

The solution was found in the 59th generation.

5 Conclusion

In this paper, we present one new and robust heuristic, based on a genetic search framework for solving the Hierarchical Covering Location Problem (HCLP). We use binary representation, so new crossover and mutation operators are constructed to keep the individuals feasible, i.e. preserve exactly *p*,*q* ones in their genetic codes. In order to increase the divisibility of genetic material we use mutation with frozen bits. Performance of GA implementation is improved by using caching GA technique. On numerous examples algorithm and the software implementation proved to be robust and behaved well on different types and sizes of the problem. GA parameters were adjusted and favorable results compared to CPLEX were obtained.

Our future research will be directed to parallelization of the presented GA and/or testing on more powerful computer on larger instances like B300, B500 and B700 problems from Beasley's Library for the p-median problem. Also, other directions can be incorporation in exact methods and application for solving similar location problems.

References:

[1] Berman, O., Drezner, Z., Wesolowsky. G.O., The expropriation location problem, *Journal of Operational Research Society*, Vol. 54, 2003, pp. 769–776.

[2] Bouktir, T., Slimani, L., Optimal power flow of the Algerian Electrical Network using genetic algorithms, *WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS*, Vol. 3, Issue 6, 2004, pp. 1478-1482.

[3] Cruz, J.B., Genshe C., Dongxu, L., Garagic, D., Target Selection in UAV Cooperative Control under Uncertain Environment: Genetic Algorithm Approach, *WSEAS TRANSACTIONS on CIRCUITS AND SYSTEMS*, Vol. 3, Issue 3, May 2004.

[4] Espejo, L.G.A., Galvao, R.D., Boffey, B., Dual-based heuristics for a hierarchical covering location problem, *Computers & Operations Research*, Vol. 30, 2003, pp. 165-180.

[5] Filipović, V., Fine-grained Tournament Selection Operator in Genetic Algorithms, *Computing and Informatics*, Vol .22, 2003, pp. 143-161.

[6] Filipović, V., *Selection and Migration operators and Web Services in Parallel Evolutionary Algorithms* (in Serbian), PhD thesis, University of Belgrade, Faculty of Mathematics, 2006.

[7] Galvao, R.D., ReVelle, C.S., A Lagrangean heuristic for the maximal covering location problem, *European Journal of Operational Research,* Vol. 88, 1996, pp. 114–123.

[8] Genco, A., Bluetooth Positioning Optimization by Genetic Algorithm, *WSEAS TRANSACTIONS ON INFORMATION SCIENCE AND APPLICATIONS*, Vol. 1, Issue 6, Dec. 2004, pp. 1584-1590.

[9] Hua, Q., Wu, B., Tian, H., A Detecting Peak's Number Technique for Multimodal Function Optimization, *WSEAS TRANSACTIONS on INFORMATION SCIENCE & APPLICATIONS*, Vol. 5, Issue 2, February 2008.

[10] Juhos, I., Van Hemert, J.I., Improving graph colouring algorithms and heuristics using a novel representation, *Lecture Notes in Computer Science*, Vol. 3906, 2006, pp. 123-134.

[11] Kovacevic, J., Hybrid Genetic Algorithm For Solving The Low-Autocorrelation Binary Sequence Problem, submitted to *Yugoslav Journal of Operations Research*.

[12] Kratica, J., Improving Performances of the Genetic Algorithm by Caching, *Computers and Artificial Intelligence*, Vol. 18**,** 1999, pp. 271-283.

[13] Li, S.S., Chen, R.C., Lin, C.C., A Genetic Algorithm-based Decision Support System for Allocating International Apparel Demand, *WSEAS Transactions on Information Science and Applications*, Vol. 3, No. 7, pp. 1294-1299, 2006

[14] Ljubić, I., *Exact and Memetic Algorithms for Two Network Design Problems*, PhD thesis, Institute of Computer Graphics, Vienna University of Technology, 2004.

[15] Ljubić, I., Raidl, G.R., A memetic algorithm for minimum-cost vertex-biconnectivity augmentation of graphs, *Journal of Heuristics*, Vol. 9, 2003, pp. 401-427.

[16] Mitchell, M., *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, Massachusetts, 1999.

[17] Puchinger, J., Raidl, G.R., Pferschy, U., The core concept for the multidimensional knapsack problem, *Lecture Notes in Computer Science*, Vol. 3906, 2006, pp. 195-208.

[18] Raidl, G.R., Gottlieb, J., Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem, *Evolutionary Computation*, Vol. 13, No. 4, 2005, pp. 441-475.

[19] Stanimirović, Z., *Solving Some Discrete Location Problems by Using Genetic Algorithms*, (in Serbian), Master's thesis, Faculty of Mathematics, University of Belgrade, 2004.

[20] Stanimirović, Z., *Genetic Algorithms for Solving Some NP-hard Hub Location Problems*, (in Serbian), Ph.D. thesis, University of Belgrade, Faculty of Mathematics, 2007.

[21] Stanimirović, Z., A genetic algorithm approach for the capacitated single allocation p-hub median problem, *Computing and Informatics*, Vol. 27, 2008, in press.

[22] Stanimirović, Z., Solving the Capacitated Single Allocation Hub Location Problem Using Genetic Algorithm, *Computing and Informatics*, Vol. 27, 2008, in press.

This research is part of the Project 144007, Ministry of Science, Serbia