A Genetic Algorithm for the P-Median Problem

Elon Santos Correa
Departamento de Matematica
Colegio Militar de Curitiba
Timoteo Jose Ferreira, 72
Curitiba-PR, Brazil
ZIP Code: 82600-590
Tel. (55) (41) 256-5917
elonsc@yahoo.com
www.geocities.com/elonsc

Maria Teresinha A. Steiner
Departamento de Matematica
Universidade Federal do Parana
Centro Politecnico
Curitiba-PR, Brazil
ZIP Code: 81531-990
Tel. (55) (41) 361-3403
tere@mat.ufpr.br

Alex A. Freitas
Departamento de Informatica
Pontificia Universidade Catolica do Parana
Imaculada Conceicao, 1155
Curitiba-PR, Brazil
ZIP Code: 80215-901
Tel. (55) (41) 330-1669
alex@ppgia.pucpr.br
www.ppgia.pucpr.br/~alex

Celso Carnieri
Departamento de Matematica
Universidade Federal do Parana
Centro Politecnico
Curitiba-PR, Brazil
ZIP Code: 81531-990
Tel. (55) (41) 361-3403
carnieri@mat.ufpr.br

Abstract

Facility-location problems have several applications in telecommunications, industrial transportation and distribution, etc. One of the most well-known facility-location problems is the p-median problem. This work addresses an application of the capacitated p-median problem to a real-world problem. We propose a genetic algorithm (GA) to solve the capacitated p-median problem. The proposed GA uses not only conventional genetic operators but also a new heuristic "hypermutation" operator proposed in this work. The proposed GA is compared with a tabu search algorithm.

Keywords: facility location, p-median problem, genetic algorithms, tabu search.

1 INTRODUCTION

Facility-location problems have several applications in telecommunications, industrial transportation and distribution, etc. One of the most well-known facility-location problems is the p-median problem. This problem consists of locating p facilities in a given space (e.g. Euclidean space) which satisfy n demand points in such a way that the total sum of distances between each demand point and its nearest facility is minimized. In the non-capacitated p-median problem, one considers that each facility candidate to median can satisfy an unlimited number of demand points. By contrast, in the capacitated p-median problem each candidate facility has a fixed capacity, i.e. a maximum number of demand points that it can satisfy. The p-median problem is *NP-hard* [Kariv and Hakimi, 1979]. Therefore, even heuristic methods specialized in solving this problem require a considerable computational effort.

In this work we apply the capacitated p-median problem to a real-world problem, namely the selection of facilities for a university's admission examination. The goal is to select 26 facilities among 43 available facilities. Each facility has a fixed capacity, i.e. a maximum number of students who can take an exam at that facility. Each student must be assigned to exactly one facility. The selected facilities must satisfy 19710 candidate students (i.e. students who have applied to the university's admission exam). In addition, the 26 facilities must be selected in such a way that the total sum of the distances between each student's home and the facility to which the student is assigned is minimized.

In order to solve this problem we propose a genetic algorithm (GA) specific for the capacitated p-median problem. The proposed GA is compared with a tabu search algorithm proposed by Glover (unpublished work).

This paper is organized as follows. Section 2 formally defines the p-median problem and the real-world application addressed in this work. Section 3 introduces the proposed GA. Section 4 reports computational results. Section 5 discusses related work. Finally, section 6 concludes the paper.

2 THE P-MEDIAN PROBLEM

Informally, the goal of the p-median problem is to determine p facilities in a predefined set with n ($n > p$) candidate facilities in order to satisfy a set of demands, so that the total sum of distances between each demand point and its nearest facility is minimized. The p facilities composing a solution for the problem are called medians.

Formally, assuming all vertexes of a graph are potential medians, the p-median problem can be defined as follows. Let $G = (V, A)$ an undirected graph where V are the vertexes and A are the edges. The goal is to find a set of vertexes $V_p \subset V$ (median set) with cardinality p, such that the sum of the distance between each remaining vertex in $\{V - V_p\}$ (demand set) and its nearest vertex in V_p be minimized.

We present below a formulation of the p-median problem in terms of Integer Programming proposed by Revelle and Swain (1970). This formulation allows that each vertex be considered, at the same time, as demand and facility (potential median), but in many cases (including our real-world application) demand and facilities belong to disjoint sets.

$$\text{Min} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i \, d_{ij} \, x_{ij} \tag{2.1}$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = 1 \, , \; i = 1, 2, ..., n \tag{2.2}$$

$$x_{ij} \leq y_j \, , \; i, j = 1, 2, ..., n \tag{2.3}$$

$$\sum_{j=1}^{n} y_j = p \tag{2.4}$$

$$x_{ij}, y_j \in \{0, 1\}, \; i, j = 1, 2, ..., n \tag{2.5}$$

where,

n = total number of vertexes in the graph

a_i = demand of vertex j.

d_{ij} = distance from vertex i to vertex j.

p = number of facilities used as medians.

$$x_{ij} = \begin{cases} 1, \text{if the vertex i is assigned to facility j} \\ 0, \text{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1, \text{if the vertex j is a facility used as a median} \\ 0, \text{otherwise} \end{cases}$$

The objective function (2.1) minimizes the sum of the (weighted) distances between the demand vertexes and the median set. The constraint set (2.2) guarantees that all demand vertexes are assigned to exactly one median. The constraint set (2.3) forbids that a demand vertex be assigned to a facility that was not selected as a median. The total number of median vertexes is defined by (2.4) as equal to p. Constraint (2.5) guarantees that the values of the decision variables x and y are binary (0 or 1).

Assuming all vertexes of a graph are potential medians, the p-median problem can be formally defined as follows. Let $G = (V, A)$ an undirected graph where V are the vertexes and A are the edges. The goal is to find a set of vertexes $V_p \subset V$ (median set) with cardinality p, such that: (a) the sum of the distance between each remaining vertex in $\{V - V_p\}$ (demand set) and its nearest vertex in V_p be minimized; and (b) all demand points are satisfied without violating the capacity restrictions of the median facilities. By comparison with the p-median problem, the capacitated p-median problem has the following additional constraints: (1) Each facility can satisfy only a limited number of demands (capacity restrictions); and (2) All demand points must be satisfied by respecting the capacities of the facilities selected as medians.

2.1 A REAL-WORLD APLICATION

The Federal University of Parana (UFPR), located in Curitiba, Brazil, was founded in 1912 as the first federal Brazilian university. It currently offers 61 undergraduate courses, 84 specialization courses (at the graduate level), 37 M.Sc. or M.A. courses and 21 Ph.D. courses. Undergraduate students are selected via a written admission exam applied to all candidate students. For the 2001 admission exam it has been proposed an optimization in the assignment of candidate students to the facilities where they will take the exam. The goal was to assign 19710 candidate students to facilities as close as possible to their corresponding homes. (In order to obtain the distance between each candidate student's home and each facility, all the addresses in question have been precisely located in a digitized map of the city of Curitiba). It was previously determined, for operational and economic reasons, that an algorithm should select 26 facilities to satisfy all 19710 candidate students, among a set of 43 candidate facilities. We cast this problem as a capacitated p-median problem, as follows:

1. The set of 43 facilities (potential exam locations) is the set V ($|V|$ = 43) of all facilities candidate to median (actual exam locations).
2. Let $Vp \subset V$ ($|Vp|$ = 26) be the set of the 26 selected exam locations.
3. Each of the 43 potential exam locations can satisfy only a limited number of candidate students.
4. The goal is to select a set $Vp \subset V$ that minimizes the total sum of distances between each candidate student's home and its nearest exam location (median).

3 THE PROPOSED GA

This section describes our proposed GA for the capacitated p-median problem, Cap-p-Med-GA.

3.1 INDIVIDUAL REPRESENTATION

Each individual (chromosome) has exactly p genes, where p is the number of medians, and the allele of each gene represents the index (a unique id number) of a facility selected as median. For instance, consider a problem with 15 facilities (potential medians) represented by the indexes 1,2,...,15. Suppose one wants to select 5 medians.

In our GA, the individual [2, 7, 5, 15, 10] represents a candidate solution for the problem where facilities 2, 5, 7, 10 and 15 have been selected as medians. In Cap-p-Med-GA the genome is interpreted as a set of facility indexes, in the mathematical sense of set - i.e. there are no duplicated indexes and there is no ordering among the indexes.

3.2 FITNESS EVALUATION

In essence, the fitness of an individual is given by the value of the objective function for the solution represented by the individual - as measured by formula (2.1). However, there is a caveat in the computation of the fitness of an individual. Note that Cap-p-Med-GA is used only to optimize the choice of the 26 medians, out of the 43 facilities. However, the computation of formula (2.1) requires that each of the 19710 candidate students be assigned to exactly one of the selected medians (i.e. the facility where the student will take the admission exam).

This assignment is done by a procedure that is used by Cap-p-Med-GA as a black box. Since this procedure is orthogonal to the use of a GA, it will not be described in detail here. For details the reader is referred to Correa (2001). Here we just mention the basic idea of this procedure. Once the 26 medians are selected, this procedure tries to assign each candidate student to the median (exam location) that is the nearest one to its home. The problem is that, since each median has a fixed capacity, some candidate students will have to be assigned to the second (or third, fourth, ...) nearest median to their homes. Suppose there is an assignment conflict - e.g. there is just one vacancy in one median, and that median is the nearest one for two candidate students. In this case the student-assignment procedure prefers to assign to that median the student that would be most prejudiced if she was assigned to its second nearest median. A student is "prejudiced" to the extent of the difference between two distances, namely the distance between her home and her nearest median and the distance between her home and her second nearest median. Once the student-assignment procedure is complete, the fitness of an individual can be computed by formula (2.1).

3.3 SELECTION

We use a ranking-based selection method proposed by Mayerle (1996), given by the formula below.

$$\text{Select(R)}=\left\{ r_j \in R \;/\; j = P - \left\lfloor \frac{-1+\sqrt{1+4.\,rnd(P^2+P)}}{2} \right\rfloor \right\}, \tag{3.1}$$

where R is a list R = (r_1, r_2, ..., r_p), with P individuals sorted in increasing order by fitness value, rnd $\in$ [0, 1) is a uniformly-distributed random number and the symbol $\lfloor b \rfloor$ denotes the greatest integer smaller than or equal to b. Formula (3.1) returns the position in the list R of the individual to be selected. The formula is biased to favor the selection of individuals in early positions of the list - i.e. the best (smallest fitness) individuals.

The population evolves according to the steady-state method. The offspring produced by crossover (and possibly mutation) is inserted into the population only if they have a better (smaller) fitness than the worst individual of the current population.

3.4 CROSSOVER

As a preprocessing step for the possible application of crossover, Cap-p-Med-GA computes two exchange vectors, one for each parent, as follows. For each gene of parent 1, Cap-p-Med-GA checks whether the allele (facility index) of that gene is also present (in any position) at the genome of parent 2. If not, that facility index is copied to the exchange vector of parent 1. This means that facility index may be transferred to parent 2 as a result of crossover, since this transfer would not create any duplicate facility indexes in parent 2's genotype. The same procedure is performed for each facility index in the genotype of parent 2. For instance, let the two parents be the facility-index vectors [1, 2, 3, 4, 5] and [2, 5, 9, 10, 12]. Their respective exchange vectors are: vp_1 = [1, 3, 4] and vp_2 = [9, 10, 12]. Once the facility indexes that can be exchanged have been identified, the crossover operator can be applied, as follows.

No fixed crossover probability is used in Cap-p-Med-GA. Crossover is performed whenever the two parents are not equal to each other, i.e. whenever there is at least one facility index in the exchange vectors of parent 1 and parent 2. If the two parents are equal to each other, i.e. their exchange vectors are empty, one of the parents is reproduced unaltered for the next generation and the other parent is deleted, to avoid that duplicate individuals be inserted into the population.

Crossover is performed as follows. A random natural number c, varying from 1 to the number of elements in the exchange vectors minus 1, is generated. This number c determines how many facility indexes of each exchange vector will be actually swapped between the two parents. We emphasize that this procedure guarantees that there will be no duplicate facility index in any of the two children produced by crossover.

3.5 MUTATION

Mutation is performed as follows. The gene being mutated has its current allele replaced by another randomly-generated allele (a facility index), subject to the restriction that the new facility index is not present in the current genotype of the individual.

3.6 HEURISTIC HYPERMUTATION

This is a new heuristic operator proposed in this work. It is based on knowledge about the problem being solved.

This operator is applied right after the random generation of the initial population, and after that it is applied with a fixed probability (e.g. 0.5%) to each iteration of the steady-state method (i.e. each selection of two parents, possibly followed by crossover and conventional mutation). This operator starts by randomly selecting a percentage (e.g. 10%) of the individuals of the population. Then it tries to improve the fitness of each of the selected individuals as follows. For each gene of the individual, it tries to replace its facility index by each facility index that is not currently present in the genotype of the individual. For each gene, the replacement that most improves the individual's fitness is performed. Note that this is a very computationally expensive operator, since each time it is applied a large number of fitness functions needs to be performed. The cost-effectiveness of this application-specific, computationally-expensive operator will be evaluated in section 4.

More precisely, the heuristic hypermutation operator proposed in this work is implemented as follows:

Procedure HYPERMUTATION:

Step 1.

Randomly select a subset of 10% of the individuals from the entire population.

Step 2.

FOR EACH individual *X* selected in Step 1 DO

Let *H* be the set of facility indexes that are not currently present in the genotype of individual *X*

FOR EACH facility index "*i*" included in set *H* DO

BEST = *X*

FOR EACH facility index "*j*" that is currently present in the genotype of the individual *X* DO

Let *Y* be a new individual with the set of facilities given by: $(X - \{j\}) \cup \{i\}$

Calculate the fitness of *Y*

If fitness(*Y*) < fitness(*BEST*) then

BEST = *Y*

END FOR

If fitness(*BEST*) < fitness(*X*) then

X = *BEST*

END FOR

Insert the new *X* into the population, replacing the old *X*

END FOR

To illustrate the use of the hypermutation operator, consider a very simple example with only 5 facilities, labeled {1, 2, 3, 4, 5}, out of which we want to select 3 medians. Consider an individual *X*, selected to undergo hypermutation, containing the facilities {1, 4, 5}. Hence, the set *H* is the set {2, 3}, and *BEST* = *X* = {1, 4, 5}. The algorithm first let *j* = 2, so that the following new individuals are evaluated: {2, 4, 5}, {1, 2, 5} and {1, 4, 2}. Suppose the best of these 3 individuals is {1, 2, 5}, which is also better than the original {1, 4, 5}. Then the algorithm let *BEST* = {1, 2, 5}. At this point the algorithm let *j* = 3, so that the following new individuals are evaluated: {3,2, 5}, {1, 3, 5}, {1, 2, 3}. Suppose the best of these 3 individuals is {3,2,5}, but this individual is not better than the previously best individual {1,2,5}. Then *BEST* remains associated with the individual {1,2,5}. At this point all indexes in *H* have been tried, so the current value of *BEST*, {1,2,5}, replaces the original individual *X* in the population. This process is performed for each individual undergoing hypermutation.

4 COMPUTATIONAL RESULTS

As mentioned earlier, the problem being solved consists of selecting 26 medians out of 43 facilities. Therefore, there are C_{43}^{26} = 421,171,648,758 (roughly 421 billion) candidate solutions.

The proposed GA was evaluated by comparing it with another heuristic algorithm developed for the problem, namely a tabu search algorithm. The tabu search algorithm used here is our implementation of the algorithm proposed by (Glover, personal communication). In essence, this tabu search algorithm works as follows.

Consider the set *V* of all candidate facilities and $V_p \subset V$, $|V_p| = p$, the initial set of randomly-selected medians. Each "move" (operator) of the tabu search is a procedure that consists of adding (ADD), removing (REMOVE) or swapping (SWAP) in V_p the median that leads to the best (smallest) value of the objective function (2.1). The moves of adding, removing and swapping are sequentially performed, so that the number of medians in the set V_p, will vary in the range: $p - 1 \leq |V_p| \leq p + 1$.

This phenomenon is called "strategic oscillation". It helps to avoid a convergence to a local optimum.

The ADD, REMOVE and SWAP moves are implemented as follows:

Procedure ADD:

Select a candidate facility from $\{V - Vp\}$ which when added to *Vp* results in the best possible value of solution. Then add this candidate facility to *Vp*. Note that each ADD move considers $|V - Vp|$ facilities as candidate to be added to the current solution (i.e. 17 or 18 facilities for the real-world problem addressed in this work).

Procedure REMOVE:

Select a median from *Vp* which when removed from *Vp* results in the best possible value of solution. Then move this median into $\{V - Vp\}$ (removing it from *Vp*). Note that each REMOVE move considers $|Vp|$ medians as candidate to be removed from the current solution (i.e. 26 or 27 medians for the real-world problem addressed in this work).

Procedure SWAP:

Select two facilities, one median from *Vp* and one facility from $\{V - Vp\}$, which, when swapped, result in the greatest improvement in the feasible solution value (all possible pair-wise exchanges are considered). Each SWAP move considers $|Vp|$ x $|V - Vp|$ pairs of facilities as candidate to be swapped (i.e. 26 x 17 = 442 candidate pairs for the real-world problem addressed in this work).

A tabu list memorizes the number of the iteration in which each median was added to a solution. During a certain number of iterations (called tabu tenure), it is forbidden to re-insert that median to the current solution, i.e. the corresponding move is a tabu (forbidden) move. The aspiration criterion used consists of allowing the tabu restriction to be ignored if the quality of the new solution produced by a tabu move is better than the quality of the best solution generated up to now by the search.

For a comprehensive, detailed discussion about tabu search in general the reader is referred to the book by Glover and Laguna (1997).

The experiments involved a comparison between two versions of Cap-p-Med-GA and the above-described tabu search algorithm. The first version of Cap-p-Med-GA is a full version of the algorithm, using all the genetic operators described in section 3. This version can be considered a hybrid GA/local search algorithm, since the heuristic hypermutation operator effectively incorporates problem-dependent knowledge into the GA. By contrast, the second version of Cap-p-Med-GA is a pure GA, which was obtained by simply switching off the heuristic hypermutation operator - i.e. this operator is never applied. In other words, it uses all the genetic operators described in section 3 except the heuristic hypermutation operator. This second version of Cap-p-Med-GA was included in our experiments to evaluate the cost-effectiveness of our proposed heuristic hypermutation operator in a controlled manner.

All results reported in this section were obtained on a Pentium III PC with 550MHz and 128 Mbytes of RAM. In order to make the comparison between the three algorithms (the two versions of Cap-p-Med-GA and the tabu search) as fair as possible, we have carefully determined the number of iterations performed by each algorithm in such a way that all the three algorithms evaluate roughly the same number of candidate solutions. This is fair because in the three algorithms the majority of processing time is by far taken by candidate-solution evaluation. More precisely, the algorithms' parameters determining the number of evaluated candidate solutions were set as follows:

Cap-p-Med-GA with heuristic hypermutation:

Population Size = 100
Number of iterations = 1000
Probability of conventional mutation = 1%
Probability of heuristic hypermutation = 0.5%
Number of individuals that are selected for undergoing hypermutation = 10% of Population Size = 10

Cap-p-Med-GA without heuristic hypermutation:

Population Size = 100
Number of iterations = 12100
Probability of conventional mutation = 1%

Tabu Search

Number of iterations = 150
Tabu tenure = 10

Note that Cap-p-Med-GA without heuristic hypermutation performs many more iterations than Cap-p-Med-GA with heuristic hypermutation, to compensate for the fact that, when heuristic hypermutation is applied at a given iteration, a very large number of candidate solutions are evaluated in that iteration. The small number of iterations of tabu search also reflects that fact that in a single iteration of the search (consisting of all possible adding, removing and swapping moves) many different candidate solutions are evaluated.

The computational results obtained by the three algorithms are reported in Table 4.1.

Table 4.1: Computational Results

	GA with heuristic hypermutat.	GA without heuristic hypermutat.	Tabu search
No. of eval. solutions	24,200	24,300	24,301
run time	01:43:34	01:43:21	01:23:37
average distance	2.33 Km	2.40 Km	2.37 Km
total distance	45,999 Km	47,313 Km	46,660 Km
% nearest facility	83%	79%	82%

The first row of Table 4.1 indicates the number of candidate solutions evaluated by each algorithm. The second row indicates the run time taken by each algorithm, in the format hours:minutes:seconds. Note that the three algorithms had about the same run time. This is a result of our having carefully determined the number of iterations of each algorithm so that each one evaluates roughly the same number of candidate solutions, as mentioned above. Therefore, a comparison among the three algorithms with respect to the quality of their produced solution is fair. The other rows of Table 4.1 are indicators of quality of the produced solution, as follows.

The third and fourth rows report respectively the average and total distance traveled by the students, measured in Km. The distance traveled by each student is the distance between the student's home and the facility (median) to which the student was assigned. The average distance is simply the total distance traveled by all 19710 students divided by 19710. The fifth row reports the percentage of students that were assigned to the facility that is indeed the facility nearest to the student's home, which is the ideal assignment for a student. Overall the three algorithms did a good job, managing to assign about 80% of the students to their ideal (nearest) facility.

With respect to both the minimization of average (or total) distance traveled by students and maximization of the percentage of students assigned to their nearest facility, the best algorithm was Cap-p-Med-GA with the heuristic hypermutation operator. The second best algorithm was tabu search. The worst algorithm was Cap-p-Med-GA without the heuristic hypermutation operator. Therefore, these results are evidence (in this application) for the cost-effectiveness of extending a conventional GA with a problem-dependent, heuristic operator.

5 RELATED WORK

Hosage and Goodchild (1986) (H&G) seem to have been the first researchers to develop a GA for the p-median problem. They used a simple GA, with conventional genetic operators. Each candidate solution was represented by a binary string, where each bit corresponds to a facility index. Each allele (1 or 0) indicates whether or not the corresponding facility is selected as a median. If the number of bits set to "1" is different from p the solution is deemed invalid and a penalty (proportional to the extent of restriction violation) is applied to the fitness of the individual. H&G tested their GA in a problem where the goal was to select 3 medians out of 20 facilities (i.e. $n = 20$, $p = 3$). They used a population of 25 individuals ($P = 25$), and did experiments with different numbers of generations, varying from 120 to 210. In experiments with randomly-generated problem instances, the GA obtained the optimal solution in about 70% and 90% of the problem instances, when running the GA for 120 and 210 generations, respectively. At first glance these are good results. However, the GA uses a classic binary individual representation, which is not very suitable for this problem. It wastes memory and processing time. The problem instances used to evaluate the algorithm had only 1140 candidate solutions (C_{20}^{3}). However, the GA generates and evaluates 2905 and 5065 solutions, when it is run for 120 and 210 generations, respectively. Although there are only 1140 candidate solutions, the search space for the GA is 2^{20} (all possible binary strings of length 20). Roughly 99.9% of the possible individuals are invalid solutions, and the GA wastes time analyzing them. Our work clearly avoids this problem, since the individual representation used in our work considers only candidate solutions with exactly the desired number of medians.

Dibble and Densham (D&D) (1993) proposed a GA with an individual representation more suitable for the p-median problem. Each individual has exactly p genes, and each gene represents a facility index. This is the same representation as the one used in our work. They used only conventional genetic operators. By contrast, we have developed a problem-dependent operator for the p-median problem, as discussed earlier. D&D applied their GA to a problem where the goal was to select 9 medians among 150. They used population size $P = 1000$ and 150 generations. They compared the results of their GA with the results obtained by the heuristic algorithm of Teitz and Bart (1968), which is a heuristic algorithm specialized for the p-median problem. Although the GA took a considerably longer processing time, both algorithms produced similar solutions.

Moreno-Perez et al. (1994) also developed a GA for the p-median problem. The individual representation is the same as the one used by D&D. They used only conventional genetic operators. Once again, this is in contrasts with our work, which proposed a problem-dependent operator for the p-median problem, as discussed earlier. One distinguishing feature of the GA proposed by Moreno-Perez et al. is that they used multiple population groups (colonies), which exchange candidate solutions with each other (via migration). The authors claim that this method helps to avoid premature convergence to a local optima. In the above reference the authors did not compare their proposed GA with any other algorithm, so it is difficult to say how cost-effective the algorithm is.

Erkut et al. (2001) also developed a GA for the p-median problem. Each individual also has exactly p genes representing a set of p selected medians. In addition to conventional genetic operators, they use the "*String-of-Change Operator*" independently suggested by Booker (1987) and Fairley (1991). This operator uses a string of change, which consists of a binary vector generated for each parent of a crossover. The parents are passed to an exclusive OR (XOR) operator. The expression a XOR b is defined as 1 if $a \neq b$ and 0 otherwise. For instance, applying XOR to the parents [10, 9, 12, 24, 7, 3] and [10, 9, 7, 8, 12, 3] one would obtain the binary vector [0, 0, 1, 1, 1, 0]. In order to avoid that crossover produces offspring identical to the parents, only the genes between the first "1" and the last "1" in the parents can be selected as crossover points.

The basic idea of this string-of-change operator is conceptually similar to the exchange vector used in our work. However, we believe our exchange vector is more suitable for the p-median problem, based on the following rationale. In order to identify the facility indexes that can be swapped between the parents, our exchange vector mechanism considers that each individual contains a (*unordered*) *set* of facility indexes. By contrast, the string-of-change, XOR mechanism considers that each individual contains a (*ordered*) *list* of facility indexes. For instance, in the above example, the facility indexes 12 and 7 were identified as possible crossover points by the string-of-change operator, despite the fact that they are present in both parents, since the position of their occurrence in the genotype is different in the two parents. By contrast, those two facility indexes would not be included in our exchange vector, since they occur in both parents. After all, the position of a facility index in the genotype is arbitrary, from the viewpoint of specifying a candidate solution. E.g., the *set* of medians {7, 12} represents the same solution as the *set* of medians {12, 7}, which is not recognized by the string-of-change operator.

6 CONCLUSIONS AND FUTURE WORK

We have proposed a GA for the capacitated p-median problem, and have applied it to a real-world problem with a quite large search space, containing roughly 421 billion ($4{,}21 \times 10^{11}$) candidate solutions. Our GA uses an individual representation and genetic operators developed specifically for the p-median problem.

In particular, we have proposed a heuristic hypermutation operator, to be used in addition to crossover and conventional mutation operators. We did experiments comparing two versions of our GA, one with this new operator and the other one without it, with a tabu search algorithm. The results show that: (a) the tabu search algorithm outperforms the GA without the heuristic hypermutation operator; but (b) the GA with the proposed heuristic hypermutation operator outperforms the tabu search algorithm. These results are evidence for the cost-effectiveness of the proposed heuristic operator, since all three algorithms evaluated roughly the same number of candidate solutions during their search. The user considered the solution produced by the GA (with the heuristic operator) very satisfactory.

Some directions for future research are as follows. Concerning the p-median problem, it seems worthwhile to develop new algorithms for this problem based on relatively new heuristic algorithms, such as *Scatter Search* and *Path Relinking*. These new heuristic algorithms, also related to evolutionary algorithms, have produced better results than GAs and tabu search in some combinatorial optimization algorithms (Glover, 1999).

Concerning the real-world application problem addressed in this paper, it would be interesting to extend the problem definition to find high-quality solutions (i.e. keeping the distance traveled by the students as small as possible) with a smaller number of selected medians. This would lead to a reduction in the costs of application of the university's admission exam, without increasing too much the distance traveled by the students. Going further, a more elaborated algorithm could perhaps directly consider the trade-off between minimizing the distance traveled by the students (which suggests selecting a larger number of medians) and minimizing the costs of the admission exam (which suggests selecting a smaller number of medians).

Finally, from a GA viewpoint, an interesting research direction is to investigate whether the heuristic hypermutation operator proposed in this work can be adapted to work, in a cost-effective manner, with other combinatorial optimization problems.

ACKNOWLEDGMENTS

We are very grateful to Fred Glover and Erhan Erkut, for having sent us papers on tabu search and GA algorithms applied to the p-median problem. In particular, as mentioned in the text, Fred Glover suggested us the tabu search algorithm implemented in this work.

REFERENCES

BOOKER, L. B. Improving Search in Genetic Algorithms. In: Genetic Algorithms and Simulated Annealing (Edited by L. Davis), 61-73, Morgan Kauffmann, Los Altos, CA, 1987.

CORREA, Elon S. Algoritmos Geneticos e Busca Tabu Aplicados ao Problema das P-Medianas. (In Portuguese) M.Sc. Mestrado em Metodos Numericos em Engenharia. Universidade Federal do Parana (UFPR). Curitiba, Brazil. 2000.

DIBBLE, C.; DENSHAM, P. J. Generating Interesting Alternatives in GIS and SDSS Using Genetic Algorithms. *GIS/LIS* symposium, University of Nebraska, Lincoln, 1993.

ERKUT, Erhan; BOZKAYA, Burçin; ZHANG, Jianjun. An Effective Genetic Algorithm for the p-median Problem. (In press.) 2001.

FARLEY, A. Comparison of Choosing the Crossover Point in the Genetic Crossover Operation. Technical Report. Dept. of Computer Science, University of Liverpool, 1991.

GLOVER, Fred; LAGUNA, Manuel. Tabu Search. *Kluwer Academic Publishers*, University of Colorado, 1997.

GLOVER, Fred. Scatter Search and Path Relinking. Graduate School of Business, University of Colorado, Boulder, *Technical Report*, 1999.

GLOVER, Fred. Tabu Search for the p-median Problem. (Unpublished paper).

GOLDBERG, David E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Menlo Park, CA: Addison-Wesley, 1989.

GOODCHILD, M. F.; NORONHA, V. Location-Allocation for Small Computers. University of Iowa, *Monograph* number 8, 1983.

HOSAGE, C. M.; GOODCHILD, M. F. Discrete Space Location-Allocation Solutions from Genetic Algorithms. *Annals of Operational Research*, 6, 35-46, 1986.

KARIV, O.; HAKIMI, S. L. The p-median problems. In: An Algorithmic Approach to Network Location Problems. SIAM *Journal on Applied Mathematics*, Philadelphia, 37, 539-560, 1979.

MAYERLE, S. F. Um algoritmo genetico para o problema do caixeiro viajante. (In Portuguese) Technical Report. Florianopolis-SC, Brazil: UFSC, 1996.

MORENO-PEREZ, J. A.; MORENO-VEGA, J. M.; MLADENOVIC, N. Tabu Search and Simulated Annealing in p-median Problems. *Talk at the Canadian Operational Research Society Conference*, Montreal, 1994.

REVELLE, C.; SWAIN, R. Central Facilities Location. *Geographical Analysis*, 2, 30-42, 1970.

TAITZ, M. B.; BART, P. Heuristic Concentration: Two-Stage Solution Construction. *Operational Research Society*, London, 16, 955-961, 1968.