Numerical Algorithms **35:** 373–388, 2004.

A genetic algorithm for solving a capacitated p-median problem

Elon Santos Correa[a], Maria Teresinha A. Steiner[b], Alex A. Freitas[c] and Celso Carnieri[b]

[a] *Computer Science Department, The University of Manchester, Oxford Road, Manchester, M13 9PL, UK*
E-mail: correae@cs.man.ac.uk

[b] *Departamento de Matematica, Universidade Federal do Parana, Centro Politecnico, Curitiba-PR, Brazil*
E-mail: {tere;carnieri}@mat.ufpr.br

[c] *Departamento de Informatica, Pontificia Universidade, Catolica do Parana, Imaculada Conceicao, 1155 Curitiba-PR, Brazil*
E-mail: alex@ppgia.pucpr.br

Received 20 December 2001; accepted 22 June 2002

Facility-location problems have several applications, such as telecommunications, industrial transportation and distribution. 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 suggested in this work. The proposed GA is compared with a tabu search algorithm.

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

AMS subject classification: 90B06, 90C27

1. Introduction

Facility-location problems have several applications, such as telecommunications, industrial transportation and distribution. 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 [17]. 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 accommodate. The p-median problem is *NP-hard* [13]. 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 among a set of 43 available facilities. Each facility has a fixed capacity, i.e., there is a maximum number of students who can take their exams at that facility. Each student must be assigned to exactly one facility. The selected facilities must accommodate 19710 candidate students (i.e., all the 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, and section 5 discusses related work.

2. The p-median problem

Throughout this paper, we shall consider n and p as positive integer numbers and $(p < n)$. Informally, the goal of the p-median problem is to select p facilities in a predefined set with n 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 that all vertices of a graph are potential medians, the p-median problem can be defined as follows: let $G = (V, A)$ be an undirected graph where V represents the set of the n vertices belonging to the graph and A represents the edges (each adge represents the distance between a pair of vertices. The goal is to find a subset of vertices $Vp \subseteq V$ (median set) with cardinality p, such that the sum of the distances between each remaining vertex in $\{V - Vp\}$ (demand set) and its nearest vertex (median) in Vp is minimized.

We present below a formulation of the p-median problem in terms of integer programming proposed by Revelle and Swain [16]. This formulation allows that each vertex be considered, at the same time, as demand and facility (potential median). However, in many cases, including our real-world application, demands (students) and facilities (potential medians, i.e., exam locations) are completely different entities.

p-median problem:

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

subject to the restrictions:

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, 2, \ldots, n, \tag{2}$$

$$x_{ij} \leqslant y_j, \qquad i, j = 1, 2, \ldots, n, \tag{3}$$

$$\sum_{j=1}^{n} y_j = p, \tag{4}$$

$$x_{ij}, y_j \in \{0, 1\}, \quad i, j = 1, 2, \ldots, n, \tag{5}$$

where

$n =$ total number of vertices in the graph,
$a_i =$ demand of vertex i,
$d_{ij} =$ distance from vertex i to vertex j,
$p =$ number of facilities used as medians,
a_i, d_{ij} are positive real numbers,

$$x_{ij} = \begin{cases} 1, & \text{if vertex } i \text{ is assigned to facility } j, \\ 0, & \text{otherwise;} \end{cases}$$
$$y_j = \begin{cases} 1, & \text{if vertex } j \text{ is a facility chosen as median,} \\ 0, & \text{otherwise.} \end{cases}$$

The objective function (1) minimizes the sum of the (weighted) distances between the demand vertices and the median set. The constraint set (2) guarantees that all demand vertices are assigned to exactly one median. The constraint set (3) prevents that a demand vertex be assigned to a facility that was not selected as a median. The total number of median vertices is defined by (4) as equal to p. Constraint (5) ensures that the values of the decision variables x and y are binary (0 or 1).

Similarly, assuming all vertices of a graph are potential medians, the *capacitated p-median problem* can be formally defined as follows: let $G = (V, A)$ be an undirected graph where V represents the set of the n vertices belonging to the graph and A represents the edges (each adge represents the distance between a pair of vertices. The goal is to find a subset of vertices $Vp \subseteq V$ (median set) with cardinality p, such that: (a) the sum of the distances between each remaining vertex in $\{V - Vp\}$ (demand set) and its nearest vertex in Vp is 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 application*

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 on the same days and at the same time. For the 2001 admission exam an optimization in the assignment of candidate students to the facilities where they will take the exam has been proposed. The goal was to assign 19710 candidate students to facilities as close as possible to their respective homes. (In order to obtain the distances 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). Besides, 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 available. It is possible to formulate 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 $Vp \subseteq V$ ($|Vp| = 26$) 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 subset $Vp \subseteq V$ that minimizes the total sum of distances between each candidate student's home and its nearest exam location (median).

3. The proposed GA

3.1. *Genetic algorithms*

Genetic algorithms (GAs) are optimization methods that evolve a population of candidate solutions to a given problem by repeatedly applying operators based on natural selection and genetic recombination to the current population. In the standard GA the initial population of solutions is randomly generated with a uniform distribution. The individuals, composing a population of candidate solutions, are called chromosomes. The chromosomes are usually represented by fixed-length strings over a finite alphabet. The measure of quality of candidate solutions is called fitness. This is a measurement of how well the chromosome optimizes the objective function. Chromosome fitness is used to probabilistically select which individuals from the population will recombine and possibly generate new solutions. A genetic operator called crossover is used to create two new chromosomes (offspring) from a pair of selected chromosomes (parents) by swapping random subsets of the genetic material from both parents. Due to the selective pressure applied on the population through a number of generations, the overall trend is towards higher-fitness chromosomes. Mutations are used to help preserve diversity in the population by introducing random changes into the chromosomes. Both crossover and mutation are usually applied with user-defined probabilities, and in general the crossover probability is much larger than the mutation probability.

For a comprehensive, detailed, discussion about genetic algorithms in general the reader is referred to the book by Goldberg [10], or to the book by Back et al. [1].

The next subsections describe our proposed GA for the capacitated p-median problem, Cap-p-Med-GA.

3.2. *Individual representation*

We define a candidate solution for our real-world (capacitated p-median) problem as feasible if (1) it has exactly p medians selected among the 43 facilities available and (2) it can accommodate all the 19710 students. In the real-world problem being solved in this work, any set containing exactly 26 medians selected among the 43 possible facilities can always accommodate all the 19710 students. Therefore it is always a feasible candidate solution. In terms of genetic algorithms we represent a candidate solution (individual) as follows: each individual (chromosome) has exactly p genes, where p is the desired number of medians, and the allele of each gene represents the index (a unique identity number) of a facility selected as median. For instance, consider a problem with 15 facilities (potential medians) represented by the indices $1, 2, \ldots, 15$. Suppose one wants to select 5 medians. In our GA, the individual [2, 7, 5, 15, 10] represents a candidate solution for this problem where the 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 indices, in the mathematical sense of a set – i.e., there are no duplicated indices and there is no ordering among the indices.

3.3. *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 (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 candidate facilities. By contrast, the computation of formula (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 performed by a procedure proposed by Correa [3]. Since this procedure is orthogonal to the use of a GA, we mention here only its basic ideas.

Once the 26 medians are selected, this method operates according to the following instructions: to start with, for each student (i) the students assignment procedure (SAP) finds $L_1(i)$ and $L_2(i)$, the nearest and the second nearest median to the student's home, respectively. At the same time, for every single student (i) it computes the value $d(i) = L_2(i) - L_1(i)$ which represents the addition to the objective function if the student (i) is assigned to the second nearest median to his or her home instead of to the nearest one. As a result, all the students are placed in an "assignment priority list" (APL) in decreasing order by their respective $d(i)$ values. When two or more students have the same $d(i)$ value, they are placed in the APL (in subsequent positions) by the order in which they were considered for the computation of their $d(i)$ values. The assignment process begins by assigning the first students in the list (i.e., those students with higher $d(i)$ values). When a median runs out of vacancies, the list is immediately recomputed. However, the new APL is computed excluding all those students who have already been assigned and those medians which are already fully allocated. After that, the procedure restarts the assignment process for all those students who have not yet been assigned, and for the medians that still have vacancies. The assignment phase finishes when each of the 19710 candidate students has been assigned to a median (exam location).

Immediately after completing the assignments of all students, the SAP performs improvements on the solution. First of all, it places all those students who have not been assigned to the nearest median to their respective homes in an "exchange-list" (EL). Afterwards, the SAP tries to exchange each student present in the EL with each student assigned to a median that does not have vacancies available. The reason why only medians with no vacancies available are considered is that, during the assignment process, if a student was not assigned to a median that did have (considering that it still has) at least one vacancy, this would mean that this median is not a better assignment for this student, and therefore does not need to be considered. The exchange that most reduces the total sum of distances is performed by swapping the assignments between the two students being considered (i.e., the medians with which the students are associated, are inverted). In an attempt to be as fair as possible, the method also inserts into the exchange-list the student that was involuntarily swapped with the student from the EL (provided that he or she was not already present in the list). When none of the experimented swaps improves the solution, the student from the EL being considered remains associated with his or her original median, the present solution remains unaltered and the next student from the list is considered. The improvement process finishes when all students in the exchange-list have been considered.

Once the assignment procedure and its improvements have been completed, the fitness of the individual (representing the set of the 26 chosen medians) is computed by formula (1).

3.4. Selection

We use a ranking-based selection method proposed by Mayerle [14], given by the formula below:

$$Select(R) = \left\{ r_j \in R \mid j = L - \left\lfloor \frac{-1 + \sqrt{1 + 4rnd(L^2 + L)}}{2} \right\rfloor \right\}, \tag{6}$$

where R is a list ($R = (r_1, r_2, \ldots, r_L)$), with L 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 less than or equal to b. Formula (6) 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 in the list – i.e., the best (smallest fitness) individuals.

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

3.5. 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 indices 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 $P_1 = [1, 2, 3, 4, 5]$ and $P_2 = [2, 5, 9, 10, 12]$. Then their respective exchange vectors would be: $E_1 = [1, 3, 4]$ and $E_2 = [9, 10, 12]$. Once the facility indices that can be exchanged have been identified, the crossover operator is applied, as described below.

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 k, varying from 1 to the number of elements in the exchange vectors minus 1, is generated. Then the first k facility indices in E_1 and E_2 are swapped between the two parent individuals, producing two children. The number k determines how many facility indices 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.

3.6. 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.

3.7. 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. It then tries to improve the fitness of each of the selected individuals as follows: for every single 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 indices 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:
        $Y = (X - \{j\}) \cup \{i\}$
        Calculate the fitness of $Y$
        IF fitness($Y$) < fitness($BEST$) THEN
           $BEST = Y$
        END IF
     END FOR
     IF fitness($BEST$) < fitness($X$) THEN
        $X = BEST$
     END IF
 – 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, labelled $\{1, 2, 3, 4, 5\}$, 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 lets $j = 2$, so that the following new individuals are assessed: $\{2, 4, 5\}$, $\{1, 2, 5\}$ 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 lets $BEST = \{1, 2, 5\}$. At this point the algorithm lets $j = 3$, so that the following new individuals are assessed: $\{3, 2, 5\}$, $\{1, 3, 5\}$, $\{1, 2, 3\}$. 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 indices 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 being compared with a tabu search algorithm also specifically developed for the p-median problem. The tabu search algorithm used here is our implementation of the algorithm proposed by Glover (unpublished work). In essence, this tabu search algorithm works as described below.

To simplify the explanations we shall consider that, given a set V with n candidate facilities, any subset $Vt \subseteq V$ containing p or $p \pm 1$ medians can satisfy the set of all demand points (i.e., it can accommodate all the 19710 students). Practically, we simply discard those generated solutions, with $p - 1$ medians, which eventuality cannot accommodate all the 19710 students.

Let V be a set with n candidate facilities. At first, an initial solution $Vt \subseteq V$ containing exactly p medians is randomly selected (theoretically $2 < p < n$). New solutions are then generated according to one of the "moves" as presented below.

In the search heuristic used with tabu search, each iteration of the search focuses on a "neighborhood" of the current solution (set Vt of facilities chosen as medians). The neighborhood is defined as that set of solutions which can be reached from the current solution by a single "move". Each move (operator) of the tabu search implemented in this application is a procedure that consists of adding (ADD), dropping (DROP) or swapping (SWAP) in Vt the median that leads to the best (smallest) value of the objective function (1). At each iteration, one of these three moves is necessarily performed and they are always applied in a cyclic order as shown by figure 1.

Note that by applying the three moves cyclically as shown above, the number of medians in the set Vt will constantly vary in the range: $p - 1 \leqslant |Vt| \leqslant p + 1$. This phenomenon is called "strategic oscillation". The strategic oscillation is designed to exploit the hypothesis that good solutions for $|Vt| = p$ are likely to be found in "regions close to" good solutions for $|Vt| = p \pm 1$. Moreover, allowing the search to intentionally pass through infeasible solutions is also a way to escape local optimum.

$$\begin{tikzcd}
DROP \arrow[r] & ADD \arrow[r] & SWAP \arrow[d] \\
SWAP \arrow[u] & DROP \arrow[l] & ADD \arrow[l]
\end{tikzcd}$$

Figure 1. Tabu search moves sequence.

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

Procedure ADD.

- Select a candidate facility from $\{V - Vt\}$ which when added to Vt results in the best possible value of solution. Then add this candidate facility to Vt. Note that each ADD move considers $|V - Vt|$ 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 DROP.

- Select a median from Vt which when dropped from Vt results in the best possible value of solution. Then drop this median from Vt moving it into $\{V - Vt\}$. Note that each DROP move considers $|Vt|$ medians as candidate to be dropped 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 Vt and one facility from $\{V - Vt\}$, which, when swapped, result in the greatest improvement in the feasible solution value (all possible pairwise swaps are considered). Each SWAP move considers $|Vt| \times |V - Vt|$ pairs of facilities as candidate to be swapped (i.e., $26 \times 17 = 442$ candidate pairs for the real-world problem addressed in this work).

As these moves are performed, tabu restrictions are applied to prevent moving back to previously investigated solutions. On the whole, these restrictions can be associated with any or all the permissible moves. However, in the heuristic implemented here, they are linked only to the ADD move. That is to say, a tabu list memorizes the number of the iteration in which each median is added to a solution. As a consequence, 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. Aspiration criteria in tabu search are used as insurance against restricting moves that could lead to finding high quality solutions. In other words, the aspiration criteria determine when a vertex can be moved even if tabu. In this application, we used a typical aspiration criterion. It 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 [9].

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. The second version of Cap-p-Med-GA, by contrast, 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 550 MHz and 128 Mbytes of RAM. In order to make the comparison among 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 of them assess 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 assessment. More precisely, the algorithms' parameters determining the number of assessed candidate solutions were set as given below:

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 to undergoing hypermutation = 10% of Population size.

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 assessed in that iteration. The small number of iterations of

Table 1
Table of computational results.

	GA with heuristic hypermutation	GA without heuristic hypermutation	Tabu search
Number of assessed solutions	24,200	24,300	24,301
Run time	01:43:34	01:43:21	01:23:37
Total distance (km)	45,999	47,313	46,660
Average distance (km)	2.33	2.40	2.37
% of students assigned to the nearest facility	83%	79%	82%

tabu search also reflects the fact that in a single iteration of the search (consisting of all possible adding, dropping and swapping moves) many different candidate solutions are assessed.

The computational results obtained by each of the three algorithms are reported in table 1.

The first row of table 1 indicates the total number of candidate solutions assessed 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 assesses 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 1 are indicators of quality of the produced solution. The third and fourth rows report respectively the average and total distance traveled by the students, measured in kilometers (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 nearest facility 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 the 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 works

Hosage and Goodchild [12] (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 ($L = 25$), 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 assesses 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) [4] 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 $L = 1000$ and 150 generations. They compared the results of their GA with the results obtained by the heuristic algorithm of Teitz and Bart [17], 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. [15] 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 contrast 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. [5] 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 [2] and Fairley [6]. This operator uses a string of change, which consists of a binary vector generated for each parent of a crossover. An exclusive OR (XOR) operator is applied to both parents. 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 indices that can be swapped between the parents, our exchange vector mechanism considers that each individual contains a (*unordered*) set of facility indices. By contrast, the string-of-change, XOR mechanism considers that each individual contains a (*ordered*) list of facility indices. For instance, in the above example, the facility indices 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 indices 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 ($4{,}21 \times 10^{11}$) c andidate solutions. Our GA uses an individual representation and genetic operators specifically developed 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 assessed 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 problems [7,8].

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 travelled 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 travelled by the students. Going further, a more elaborated algorithm could perhaps directly consider the trade-off between minimizing the distance travelled 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.

Acknowledgements

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

[1] T. Back, D.B. Fogel and T. Michalewicz, *Evolutionary Computation 1: Basic Algorithms and Operators* (Institute of Physics, Bristol, UK, 2000).

[2] L.B. Booker, Improving search in genetic algorithms, in: *Genetic Algorithms and Simulated Annealing*, ed. L. Davis (Morgan Kauffmann, Los Altos, CA, 1987) pp. 61–73.

[3] E.S. Correa, Algoritmos geneticos e busca tabu aplicados ao problema das p-medianas, Master dissertation, Programa de Pós-Graduação em Metodos Numericos em Engenharia, Universidade Federal do Parana, Brazil (2001).

[4] C. Dibble and P.J. Densham, Generating interesting alternatives in GIS and SDSS using genetic algorithms, *GIS/LIS Symposium*, University of Nebraska, Lincoln, 1993.

[5] E. Erkut, B. Bozkaya and J. Zhang, An effective genetic algorithm for the p-median problem, Preprint (2001).

[6] A. Fairley, Comparison of choosing the crossover point in the genetic crossover operation, Preprint, Department of Computer Science, University of Liverpool (1991).

[7] F. Glover, Scatter search and path relinking, Preprint, Graduate School of Business, University of Colorado, Boulder (1999).

[8] F. Glover, Tabu search for the p-median problem, Preprint, University of Colorado, Boulder (1999).

[9] F. Glover and M. Laguna, *Tabu Search* (Kluwer Academic, University of Colorado, 1997).

[10] D.E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning* (Addison-Wesley, Menlo Park, CA, 1989).

[11] M.F. Goodchild and V. Noronha, *Location-Allocation for Small Computers*, Monograph No. 8 (University of Iowa, 1983).

[12] C.M. Hosage and M.F. Goodchild, Discrete space location-allocation solutions from genetic algorithms, Ann. Oper. Res. 6 (1986) 35–46.

[13] O. Kariv and S.L. Hakimi, The p-median problems, in: *An Algorithmic Approach to Network Location Problems*, SIAM J. Appl. Math. 37 (1979) 539–560.

[14] S.F. Mayerle, Um algoritmo genetico para o problema do caixeiro viajante, Preprint, Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina (UFSC), Florianopolis, Santa Catarina, Brazil (1996).

[15] J.A. Moreno-Perez, J.M. Moreno-Vega and N. Mladenovic, Tabu search and simulated annealing in p-median problems, Talk at: *The Canadian Operational Research Society Conf.*, Montreal, 1994.

[16] C. Revelle and R. Swain, Central facilities location, Geographical Analysis 2 (1970) 30–42.

[17] M.B. Teitz and P. Bart, Heuristic concentration: Two-stage solution construction, Oper. Res. Soc. 16 (1968) 955–961.