J Glob Optim (2007) 38:249–264
DOI 10.1007/s10898-006-9076-2

GASUB: finding global optima to discrete location problems by a genetic-like algorithm

Blas Pelegrín · Juani L. Redondo · Pascual Fernández · Inmaculada García · Pilar M. Ortigosa

Received: 30 December 2005 / Accepted: 29 July 2006 / Published online: 22 November 2006

Abstract In many discrete location problems, a given number s of facility locations must be selected from a set of m potential locations, so as to optimize a predetermined fitness function. Most of such problems can be formulated as integer linear optimization problems, but the standard optimizers only are able to find one global optimum. We propose a new genetic-like algorithm, GASUB, which is able to find a predetermined number of global optima, if they exist, for a variety of discrete location problems. In this paper, a performance evaluation of GASUB in terms of its effectiveness (for finding optimal solutions) and efficiency (computational cost) is carried out. GASUB is also compared to MSH, a multi-start substitution method widely used for location problems. Computational experiments with three types of discrete location problems show that GASUB obtains better solutions than MSH. Furthermore, the proposed algorithm finds global optima in all tested problems, which is shown by solving those problems by Xpress-MP, an integer linear programing optimizer (21). Results from testing GASUB with a set of known test problems are also provided.

Keywords Combinatorial optimization · Discrete location problems · Stochastic algorithms · Multimodal genetic algorithms

B. Pelegrín (✉)· P. Fernández
Department of Statistics and Operational Research,
University of Murcia, Murcia, Spain,
e-mail: pelegrin@um.es

J. L. Redondo · I. García · P. M. Ortigosa
Department of Computer Architecture and Electronics,
University of Almería, Almeria, Spain,
e-mail: juani@ace.ual.es

P. Fernández
e-mail: pfdez@um.es

I. García
e-mail: inma@ace.ual.es

P. M. Ortigosa
e-mail: pilar@ace.ual.es

1 Introduction

In many combinatorial optimization problems one has to select s objects in a set of m of such objects, $s < m$, so that a given function, whose value depends on the selected objects, will be optimized. This is the case of a wide class of discrete location problems where decision makers have to choose s locations for new facilities in a set of m potential facility sites in order to serve a set of customers, which indeed are aggregated in a finite number of demand points (see (1)). Several types of location models have been developed to help decision makers in a variety of situations. These models can be classified in two big groups: non-competitive models and competitive models, depending on whether a single or multiples players in the marked are considered. A detailed taxonomy can be found in the survey papers (2–4).

In a non-competitive situation, usually the objective is the *minimization of transportation cost*, which is due to the interaction between customers and facilities. The most common method for the allocation of demand in this setting is that each customer will be served by its closest facility. This way, the transportation cost is minimized for each fixed set of new facilities. An outstanding problem in this group is known as the s-MEDIAN problem, which frequently appears in many distribution systems (see (5,6)). In a competitive situation, firms compete for the customers, and the usual objectives are the *maximization of market share* and the *maximization of profit*. The objective function is determined by using some facility choice rule, which depends on customer behavior. If customers are supposed to buy at the cheapest facility, strategic decisions on location and price have to be made. Firms normally use either a mill price policy (the seller sets a factory price, equal for all the customers in the market, and the buyer takes care of carriage) or a delivered price policy (the seller charges a specific price in each market area, which includes the freight cost, and takes care of transport). A popular problem under mill pricing is MAXCAP (see (7)). Under delivered pricing, there exist equilibrium prices that are determined by the facilities location (see (8–10)), thus the location-price problem becomes a location problem when firms charge equilibrium prices. Some vertex-optimality and location equilibrium results on a network when each firm opens only one facility are shown in (10,11). However, the resultant location problem for the entering firm when it opens more than one facility, to our knowledge, has only been studied in (12), and it will be referred here as the MAXPROFIT problem. Other patronizing behavior of customer takes into account facility characteristics as it happens with Huff-like based models (see (13,14)).

Finding more than one global optimum is of great interest in real applications of the above mentioned models. Thus, decision makers can take into account other location criteria when they are offered several alternatives which optimize the corresponding objective function. Many of these problems can be formulated as Integer Linear Programs (ILP), but the standard ILP software (e.g., Xpress-MP, CPLEX, ...) generates only one global optimum. The aim of this paper is to propose a multimodal genetic algorithm, GASUB, which is able to find a predetermined number of global optima for such problems, if they exist. The new algorithm is related to a previous algorithm given in (15) which was used to solve continuous problems. We have selected three discrete location problems, which are formulated as ILP in the same framework, in order to show that GASUB is able to find more than one global optimum. This algorithm is also compared with the multi-start substitution heuristic, MSH, a procedure widely used for many combinatorial location problems (see (5)). This paper is organized as follows. In Sect. 2, the location problems together with its

ILP formulations are described. In Sect. 3, a detailed description of the new algorithm is provided. Computational results are shown and analysed in Sect. 4, where solutions for a set of test problems are found by an standard optimizer (Xpress-MP), GASUB and MSH. The proposed algorithm has also been evaluated using two additional sets of known test problems. Finally, some conclusions and future lines of research are mentioned in Sect. 5.

2 The discrete location problems

We will consider the s-MEDIAN, MAXCAP, and MAXPROFIT problems, where locations for a set of s new facilities owned by the same company have to be determined in order to optimize the corresponding objective function. The common ingredients to these problems are: a set of m potential sites for the new facilities, a set of n demand points, a matrix of distances between demand points and potential sites, and a fixed and known demand at each demand point. The differences between them are related to the existence of pre-existing facilities or the price policy when competition is concerned. A characteristic of these problems is that customers are optimally served by its closest facility for any selection of the new facilities. In the following we present the standard procedures to deal with these problems.

2.1 Exact algorithms

The only algorithms developed to find optimal solutions are based in their formulations as ILP. In order to formulate the three problems, we will use the following notation when necessary :

$i, I = 1, 2, \ldots, n$	Index and collection of demand points
$j, J = 1, 2, \ldots, m$	Index and potential sites for facility location
$k, K = 1, 2, \ldots, q$	Index and pre-existing facility locations
w_i	Demand (or buying power) at point i
d_{ij}	Distance between demand point i and point j
$D_i = \min\{d_{ik} : k \in K\}$	Distance from demand point i to the closest pre-existing facility
$N_i^< = \{j \in J : d_{ij} < D_i\}$	Collection of potential locations for servers that are closer to point i than the closest pre-existing facility
$N_i^= = \{j \in J : d_{ij} = D_i\}$	Collection of potential locations for servers that are at the same distance to point i as the closest pre-existing facility
$I^* = \{i \in I : N_i^< \cup N_i^= \neq \emptyset\}$	Collection of demand points that have at least one potential location for server closer than or at the same distance than the pre-existing facilities
t	Unit transportation cost
p_{prod}	Marginal production cost
$p_{\min}$	Minimum selling price at the facility door

In the s-MEDIAN problem, there is no pre-existing facility in the market ($K = \emptyset$) and the objective is to minimize total transportation cost between demand points and their closest facilities. The following decision variables are defined:

$$y_j = \begin{cases} 1, & \text{if a new facility is opened in } j, \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{ij} = \text{ proportion of demand at } i \text{ served from site } j.$$

Then the problem is formulated as follows:

$$(P_1) \begin{cases} \min \sum_{i \in I} \sum_{j \in J} t w_i d_{ij} x_{ij} \\ s.t. \ \sum_{j \in J} x_{ij} = 1, \quad i \in I \\ \quad x_{ij} \le y_j, \quad i \in I, j \in J \\ \quad \sum_{j \in J} y_j = s \\ \quad x_{ij} \ge 0 \\ \quad y_j \in \{0,1\}. \end{cases}$$

In the MAXCAP problem, there already exist some competing pre-existing facilities. Customer pays for transportation and buys at its closest facility. If a new facility and a pre-existing facility are the closest to a demand point i, then demand at i is divided so that a fixed proportion θ_i of customers buy at the new closest facilities, $0 \le \theta_i \le 1$. The following decision variables are defined:

$$y_j = \begin{cases} 1, & \text{if a new facility is opened in } j, \\ 0, & \text{otherwise,} \end{cases}$$

$$x_i = \begin{cases} 1, & \text{if the new facilities capture demand point } i, \\ 0, & \text{otherwise,} \end{cases}$$

$$z_i = \begin{cases} 1, & \text{if demand point } i \text{ is divided,} \\ 0, & \text{otherwise.} \end{cases}$$

then the problem is formulated as follows:

$$(P_2) \begin{cases} \max \sum_{i \in I} w_i x_i + \sum_{i \in I} \theta_i w_i z_i \\ s.t. \ x_i \le \sum_{j \in N_i^<} y_j, \quad i \in I^* \\ \quad z_i \le \sum_{j \in N_i^=} y_j, \quad i \in I^* \\ \quad x_i + z_i \le 1, \quad i \in I^* \\ \quad \sum_{j \in J} y_j = s \\ \quad x_i \ge 0, z_i \ge 0 \\ \quad y_j \in \{0,1\}. \end{cases}$$

In the MAXPROFIT problem, facilities take charge of transportation and deliver the product to customers. Each facility offers a specific price at each demand point, which has to be greater or equal than the minimum selling price plus the transportation cost. Let $p_{\text{net}} = p_{\min} - p_{\text{prod}} \ge 0$, where $p_{\min}$ and p_{prod} are supposed not to depend on site location. As result of price competition, the optimal price a new facility j can offer at demand point i is the equilibrium price, which is given by $p_{\min} + tD_i$ if $d_{ij} \le D_i$ (otherwise no demand from i is captured by facility j). With equilibrium prices, only the closest facility to a demand point i can offer the lowest price in i. Then, the same rule as in MAXCAP is used for tie breaking when two or more facilities are the closest to a demand point. The following decision variables are defined:

$$y_j = \begin{cases} 1, & \text{if a new facility is opened in } j, \\ 0, & \text{otherwise,} \end{cases}$$
$$x_{ij} = \text{proportion of demand at } i \text{ served from site } j,$$
$$z_i = \begin{cases} 1, & \text{if demand point } i \text{ is divided,} \\ 0, & \text{otherwise} \end{cases}$$

then the problem is formulated as follows:

$$(P_3)\begin{cases} \max \sum_{i\in I^*} \sum_{j\in N_i^<} [p_{\text{net}} + t(D_i - d_{ij})] w_i x_{ij} + p_{\text{net}} \sum_{i\in I^*} \theta_i w_i z_i \\ s.t. \sum_{j\in N_i^<} x_{ij} + z_i \le 1, \quad i \in I^* \\ \quad x_{ij} \le y_j, \quad i \in I^*, j \in N_i^< \\ \quad z_i \le \sum_{j\in N_i^=} y_j, \quad i \in I^* \\ \quad \sum_{j\in J} y_j = s \\ \quad x_{ij} \ge 0, z_i \ge 0 \\ \quad y_j \in \{0,1\}. \end{cases}$$

Problems (P_1), (P_2), and (P_3) can be exactly solved by standard integer linear programing optimizers, but only for problems of moderated size, which is due to the optimizer cannot manage the corresponding matrices when the cardinalities of I and/or J are very large. In fact, (P_1) and (P_2) have been proved to be NP-Hard (see (16,17)) and (P_3) is also NP-Hard since it becomes (P_2) by taking $p_{\text{net}} = 1, t = 0$ and $x_i = \sum_{j\in N_i^<} x_{ij}$.

2.2 Heuristic algorithms

Due to their complexity, some heuristic solution techniques have been developed in order to find nearly optimal solutions to large-scale problems with a reasonable computational effort. For finding good solutions to these problems, and many other combinatorial optimization problems, they are formulated as follows:

$$\max\{\Pi(S) : |S| = s, S \subset J\}.$$

Then an important task is the evaluation of the objective function for each possible set S.

Given $S \subset J$, we denote by $d_i(S)$ the distance between the demand point i and the closest location in S, $d_i(S) = \min\{d_{ij} : j \in S\}$. If we consider the sets $I_S^1 = \{i : d_i(S) < D_i\}$ and $I_S^2 = \{i : d_i(S) = D_i\}$, then the objective functions of the mentioned location problems are as follows:

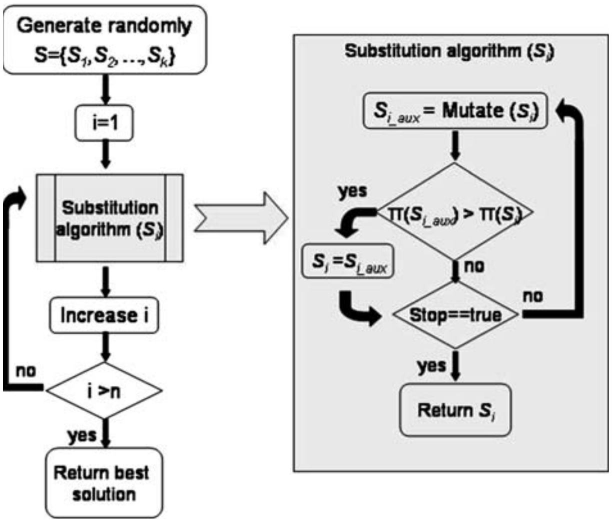


Fig. 1 Structure of MSH algorithm

$$
\begin{aligned}
s-\text{MEDIAN}: &\quad \Pi_1(S) = -\sum_{i\in I} w_i d_i(S),\\
\text{MAXCAP}: &\quad \Pi_2(S) = \sum_{i\in I_S^1} w_i + \sum_{i\in I_S^2} \theta_i w_i,\\
\text{MAXPROFIT}: &\quad \Pi_3(S) = p_{net}\left(\sum_{i\in I_S^1} w_i + \sum_{i\in I_S^2} \theta_i w_i\right) + t\sum_{i\in I_S^1}(D_i - d_i(S))w_i.
\end{aligned}
$$

The most popular heuristic to solve this type of problem is MSH, which is a multistarting facility-exchange procedure that attempts to improve the objective function value at each iteration. The structure of this algorithm is shown in Fig. 1, where the operator *mutate*(S) replaces one facility in S by one facility in $J \setminus S$ following a predetermined order. This procedure ends when all possible exchanges have been done for a given S without improving the objective function. There also exist other more sophisticated heuristics (see for instance (18–20)), but most of them are specific for each type of problem and are not considered here.

3 The GASUB algorithm

In this section the basic concepts, the algorithm, and the setting of the parameters are outlined.

3.1 Problem encoding

A point (individual in terms of genetic algorithms) consists of a single string that is a collection of m bits. The position of a bit in the string coincides with the index of the associated facility. Because of the set S of selected facilities is predetermined for every

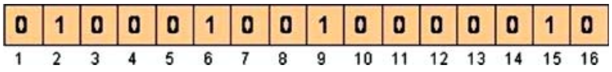


Fig. 2 Example of chromosome

problem, the number of bits to 1 value must be fixed to the number of new facilities (cardinal of S). We must consider this constraint when generating any search point.

Figure 2 shows an example of a solution to a problem where we have to select four new locations from 16 possible locations. The solution or chromosome has four genes to 1 value and 12 to 0. In this example, genes set to 1 value are at positions 2, 6, 9, and 15. It means that the chosen facilities are those with these identifiers.

3.2 Basic concepts

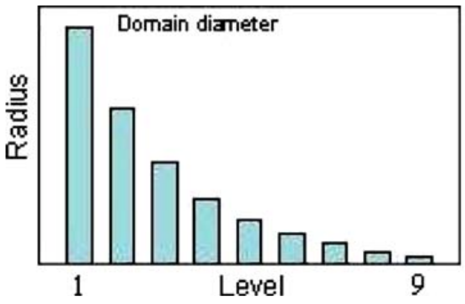
A key notion in GASUB is that of a subpopulation. A subpopulation would be equivalent to a single individual, which is defined by a center, a fitness function and a radius value. The center is a solution and the radius indicates the attraction area of this subpopulation. This definition assumes a *distance* defined over the search space. For our combinatorial problem we define the Hamming distance. As a consequence of the constraint of the problem, where the number of chosen facilities and hence the number of bits (or genes) to 1 value is fixed, the Hamming distance between any two feasible points (individuals) must be always multiple of 2.

The radius of a subpopulation is not arbitrary; it is taken from a list of decreasing radii that follows a *cooling schedule* (see Fig. 3). The first element of this list is the diameter of the search space. If the radius of a subpopulation is the ith element of the list, then the level of the subpopulation is said to be i. Given the largest radius and the smallest one (r_1 and r_{levels}, respectively) the radii in the list are expressed by the exponential function:

$$r_i = r_1 \left(\frac{r_{levels}}{r_1} \right)^{\frac{i-1}{levels-1}}, \quad (i = 1, \ldots, levels).$$

The parameter *levels* indicates the maximal number of levels in the algorithm, i.e., the number of different 'cooling' stages. Every level i (i.e., for levels from [1, *levels*]) has a radius value (r_i) and two maxima on the number of function evaluations (f.e.) namely new_i (maximum f.e. allowed when creating new subpopulations) and n_i (maximum f.e. allowed when mutating individuals).

Fig. 3 Radius values for the levels based on an exponentially decreasing function



During the optimization process, a list of subpopulations is kept by GASUB and this *subp—list* defines the whole population.

3.3 Input parameters

In GASUB, the most important parameters are those defined at each level: the radii (r_i) and the numbers of function evaluations for subpopulations creation (new$_i$) and optimization (n_i). These parameters are computed from some user-given parameters that are easier to understand:

levals: The maximal number of function evaluations the user allows for the whole optimization process. It could be called as *Whole Budget*. Note that the actual number of function evaluations may be less than this value.
levels: The maximum number of levels, i.e. the number of cooling stages.
max_subp_num: The maximum length of the *subp − list*.
r_{levels}: The radius associated with the maximum level, i.e. *levels*.

3.4 The Algorithm

The GASUB algorithm has the following structure:

```
Begin GASUB
    Initializing population
    Mutation($n_1$)
    for $i = 1$ to levels
        Determine $r_i$, $new_i$, $n_i$
        Generation and crossover($new_i$/length(subp_list))
        Selection($r_i$, max_subp_num)
        Mutation($n_i$/max_subp_num)
        Selection($r_i$, max_subp_num)
    end for
End GASUB
```

In the following, the main procedures of the algorithm are described:

Initializing population: A new subpopulation list consisting of a single subpopulation with a random center at first level is created. The center must have as many genes to 1 value as the number of new facilities are being chosen. The associated radius is the diameter of the search space. This subpopulation is the first element of the *subp_list* and it will be kept during the whole optimization, though its center would be replaced.

Generation and Crossover: For every subpopulation in the *subp_list*, new random individuals are generated in its attraction area, and for every pair of new individuals the objective function is evaluated at the middle of the *section* connecting the pair (see Fig. 4). All generated individuals must satisfy the constraint of having a fixed number of genes to 1 value. If the fitness value of the middle point is better than the fitness value of the center of the subpopulation, then this individual will be the new center, keeping the same level value. If the value in the middle point is worse than the values of the pair, then the members of the pair are inserted in the *subp—list*. Every newly inserted subpopulation is assigned the actual level value (i).

As a result of this procedure the *subp_list* will eventually contains several subpopulations with different levels (hence different radii). The motivation behind

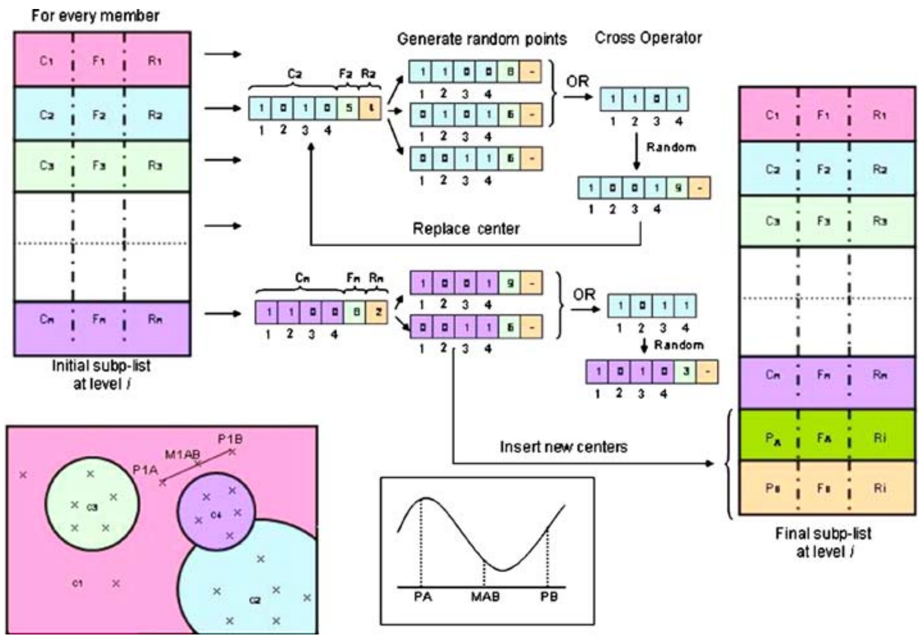


Fig. 4 Generation and crossover

this method is to create subpopulations that are on different ‘hills’ so ensuring that there is a valley between the new subpopulations.

Selection: This procedure has two mechanisms, the first tries to fuse subpopulations that are too close and the second selects the subpopulations that will be maintained in the *subp − list*.

> *Fuse_subpopulations:* If the centers of any pair of subpopulations from the *subp − list* are closer to each other than the given radius, the two subpopulations are fused. The center of the new subpopulation will be the one with the better function value while the level will be the minimum of the levels of the fused subpopulations (so the radius will be the largest one).
> *Select_subpopulations:* It deletes subpopulations to reduce the list length to the given value. Higher level subpopulations are deleted first; therefore subpopulations with larger radii are always kept. For this reason one subpopulation at level 1 whose radius is equal to the diameter of the search domain always exists, making it possible to escape from local optima.

Mutation: It applies consecutive mutations to every center of every subpopulation. A mutation means an interchange of one facility, so one gene of the center is changed from 0 to 1 and other gene is changed from 1 to 0, according to our problem encoding.
When mutating an individual, if its function value is better than the value of the center, then this new individual will replace the center. At level *i*, the maximum number of consecutive mutations applied to each center is $n_i/\max\_subp\_num$.

Note that in fact, that GASUB may terminate simply because it has executed all its levels and can be smaller than the input parameter evals. This behavior is qualitatively different from genetic algorithms which typically run until a maximum number of function evaluations. The number of function evaluations can be reduced if the number of subpopulations found during the optimization process depends on the problem and may be smaller than the maximum allowed that is determined by the input parameter *max_subp_num*. Thus, the final number of function evaluations depends on the complexity of the objective function that can determine the number of subpopulations.

4 Computational experiments

The computational experiments are aimed at evaluating GASUB as multimodal global optimization algorithm that is able to solve different locations problems (s-MEDIAN, MAXCAP, and MAXPROFIT). In this way results should demonstrate that GASUB finds more than a single global optimum, if there are more than one for a particular problem. In Sect. 4.1 the problems have been previously solved using Xpress-MP in order to known the exact solution to each problem. Also the algorithm MSH has been used to compare the efficiency, effectiveness and computational time of both heuristics. All the computational results have been obtained under Linux on a Pentium IV with 3 GHz CPU and 2GB memory. The algorithms were implemented in C++. Due to the stochastic nature of MSH and GASUB, every experiment has been executed 10 times with each algorithm and results showed in tables are average values.

As for setting input parameters in GASUB algorithm, several preliminary experiments varying the input parameters have been done to find a robust parameter setting. From those experiments it could be said that a robust parameter setting consists of a large enough number of levels (l), a small minimum radius (r_l), a sufficient maximum number of subpopulations (M), and a large value of the number of function evaluations (N). Since the parameter N should be greater when the problem is harder, the following experimental expression for setting N has been used: $N = 213419,118 \cdot s - 421691,176$ for $s \geq 2$, where s is the number of new facilities. The remaining input parameter were set to $r_l = 2$, $l = 10$ and $M = 10$. Taking in account that the maximal number of subpopulations has been set to 10, the maximal number of global optima that GASUB is able to find is 10. This maximal number of global optima has been found by the algorithm for some of the tested problems.

4.1 Comparison of the algorithms

In our computational experiments three sets of points containing 1,046, 1,273, and 1,671 cities in Spain as demand points, and their corresponding populations as demand have been used. The population of these cities is over 4,000, 3,000, and 2,000 inhabitants in each set, respectively. The geographic coordinates and population of each city was obtained from *http://www.terra.es/personal/GPS.2000* and *http://www.ine.es*, respectively. Distances were taken as the Euclidean distances between cities and each city demand was taken proportional to its population. All cities were chosen as location candidates (the set J) in all test problems. It is important to remark that results for other additional problems containing 4,072 demand points have not been included because Xpress-MP was not able to solve them using the computer mentioned above (see (22)). However, the heuristic algorithms did not have any problem in solve them because they have less memory requirements.

Table 1 Results for the s-MEDIAN problem

n	1046				1671			
	XP	GASUB		MSH	XP	GASUB		MSH
s	T	T	O	T	T	T	O	T
2	1525.0	7.4	1	9.0	10032.6	13.0	1	24.4
4	1396.1	32.7	1	41.6	8671.6	58.9	1	121.2
6	661.6	88.2	1	100.4	4198.4	151.1	1	271.8
8	439.0	157.0	1	168.7	2663.6	265.5	2	458.3
10	374.1	234.1	1	258.2	1794.0	403.6	1	715.3

In order to have the same input parameter for computational testing with the three problems, we have taken $p_{\min} = p_{\text{prod}}$. Then an overall view on the performance of MSH and GASUB can be obtained by varying the number of new facilities ($s = 2, 4, 6, 8, 10$) and the number of pre-existing facilities ($q = 2, 4, 6, 8, 10$). All test problems were optimally solved by using Xpress-MP and their optimal values were used to evaluate the performance of both heuristics.

The following three Tables 1, 2, and 3 show results obtained for the s-MEDIAN, MAXCAP, and MAXPROFIT problems, respectively. Each row of a table specifies the results obtained for a different value of s and q. The results showed in tables are grouped in two big columns, for the sets of points containing 1,046 and 1,671 cities, respectively. Results for 1,273 cities are not included in the tables for sake of space.

For each experiment, tables show the computational time required by Xpress-MP (XP), which always finds a single global optimum. Results showed for GASUB are the average value of the computational time (T) obtained from ten runs and the average number of optima (O) found by the algorithm. For problems where the number of found optima is always one, the corresponding column has been omitted. Due to the fact that GASUB always found at least a global optimum, the percentage of success in finding an optimal solution (100%) is not indicated. For the MSH algorithm, tables show the average computational time (T) obtained from ten runs and the percentage of success (S) in finding an optimal solution. This algorithm only finds a single optimum, so the columns associated have been omitted from tables.

Table 1 shows results for the s-MEDIAN problem. For this problem all the algorithms found the global optimum and the percentage of success is 100% for every case. It can be seen that the problem with 1,671 cities is harder and the algorithms need more time to solve it than for the problem of 1,046 cities.

For a particular problem, results show that Xpress-MP is the most time consumer while GASUB is the least one. The computational time of MSH are slightly bigger than the time of GASUB. It can be seen that the computational time for Xpress-MP decreases as the number of new facilities (s) increases while the time for heuristics increases with s. Run times of Xpress-MP might be explained by the fact that the number of variables in this problem decreases as long as the value of s increases. With respect to the number of optima, it is interesting to remark that GASUB found two global optima when $s = 8$ and the number of cities was 1,273 and 1,671.

Table 2 shows results for the MAXCAP problem. It can be seen that the computational time for Xpress-MP has been reduced drastically in such a way that now is quite smaller than for both heuristics. This is a consequence of the formulation of this problem as ILP. It means that the number of variables is much less than for the s-MEDIAN problem. The computational times for the heuristics are

Table 2 Results for the MAXCAP problem

n		1046					1671				
		XP	GASUB		MSH		XP	GASUB		MSH	
q	s	T	T	O	T	S	T	T	O	T	S
2	2	2.1	8.0	1	9.7	90	5.3	13.9	1	29.9	100
	4	1.9	34.1	10	32.5	100	4.5	59.9	2	96.9	80
	6	1.8	88.0	10	75.1	80	4.4	149.6	2	207.8	30
	8	1.8	153.8	10	113.4	100	4.3	260.0	4	317.9	100
	10	1.8	237.0	10	126.4	100	4.3	396.3	10	316.9	100
4	2	1.7	7.9	1	9.5	100	5.4	13.9	1	25.3	100
	4	1.8	33.4	1	45.8	100	6.0	58.7	1	114.6	80
	6	2.0	89.3	1	104.6	90	7.9	150.6	1	263.3	70
	8	1.6	156.4	1	160.7	30	5.7	263.8	1	447.4	10
	10	2.4	241.5	2	264.2	20	5.9	402.6	2	653.4	10
6	2	2.4	7.9	1	10.3	100	6.2	13.8	1	28.2	100
	4	2.5	33.8	1	43.2	100	6.0	60.0	1	125.2	100
	6	2.0	90.3	1	106.7	60	9.8	154.3	1	297.9	20
	8	2.1	161.7	2	173.1	100	8.7	269.7	1	519.7	100
	10	1.9	245.8	2	264.0	100	7.7	410.2	2	745.7	100
8	2	1.5	7.9	1	9.0	100	5.3	13.7	1	26.4	100
	4	2.1	33.7	1	40.9	80	5.6	60.0	1	126.4	100
	6	1.7	89.9	1	107.9	90	6.3	153.4	1	293.0	60
	8	1.6	158.0	2	170.0	70	6.3	270.9	2	449.5	90
	10	1.6	246.3	1	259.8	30	6.0	408.8	2	733.0	10
10	2	1.5	7.9	1	8.7	100	4.1	13.8	1	26.5	100
	4	1.4	34.3	1	40.5	10	5.3	61.1	1	117.8	100
	6	1.7	92.7	1	100.8	80	3.9	154.0	1	272.8	100
	8	1.5	162.4	1	176.7	10	4.4	280.3	2	475.8	50
	10	1.6	251.8	1	279.9	40	4.9	421.5	1	711.1	70

not so different than times for the s-MEDIAN problem, and they also follow the same tendency of growing with s. Comparing both heuristics it can be seen that when the number of optima is not too big, GASUB needs less time than MSH, though when the number of optima is high (10) then MSH finds a solution in less time than *GASUB*. It is interesting to remark that GASUB finds in several cases more than a single global optima, mainly for the cases with $q = 2$, where the algorithm is able to find ten optima. Note that maximum number of optima coincides with the maximum number of subpopulations that has been fixed to 10 in the experiments. Finally, it must be noticed that though computational times for MSH are similar to times for GASUB, the values of the percentage of success are bellow 100% in several cases due to the fact that MSH gets trapped in local optima.

Table 3 shows results for the MAXPROFIT problem. For this problem, GASUB is only able to find a single global optimum for each case. The computational times for GASUB and MSH are similar thought slightly smaller for GASUB. For this problem *MSH* always finds the global optimum except for the case $q = 4$ and $s = 10$, where 90% of success was reached. Again, the computational times for Xpress-MP decreases when s increases (with exceptions, not very significant, for $q = 10$ and $1,046$ nodes;

Table 3 Results for the *MAXPROFIT* problem

n		1046				1671			
		XP	GASUB	MSH		XP	GASUB	MSH	
q	s	T	T	T	S	T	T	T	S
2	2	568.1	8.0	9.4	100	3268.0	14.1	28.7	100
	4	307.2	34.1	42.4	100	2086.2	60.7	123.3	100
	6	256.8	91.8	106.4	100	1332.5	155.9	278.4	100
	8	207.8	161.9	166.4	100	1052.1	272.0	437.6	100
	10	168.5	241.1	255.5	100	978.3	404.3	697.8	100
4	2	65.3	7.9	9.2	100	386.8	13.9	27.4	100
	4	60.5	34.3	43.2	100	238.4	61.1	117.0	100
	6	56.5	89.3	99.1	100	319.0	154.2	269.3	100
	8	43.1	161.3	175.6	100	256.3	267.7	451.1	100
	10	29.4	244.4	263.3	**90**	199.0	412.3	741.2	100
6	2	81.2	8.0	9.0	100	469.6	14.1	26.7	100
	4	73.6	34.6	40.5	100	348.8	60.7	119.1	100
	6	59.9	90.3	96.9	100	317.2	153.3	274.8	100
	8	50.5	160.4	171.1	100	250.2	266.4	463.0	100
	10	44.5	245.6	253.9	100	260.5	414.5	709.5	100
8	2	31.7	7.9	9.8	100	177.1	13.9	25.4	100
	4	31.1	34.3	42.0	100	166.0	60.9	123.6	100
	6	27.5	91.7	96.7	100	160.3	155.9	271.5	100
	8	26.0	162.3	174.3	100	170.6	273.1	451.1	100
	10	23.0	244.9	253.9	100	122.0	417.1	734.4	100
10	2	28.7	8.0	9.6	100	137.5	13.9	27.7	100
	4	24.7	34.1	39.2	100	122.4	60.6	113.7	100
	6	27.5	90.9	103.5	100	142.9	153.2	257.8	100
	8	24.1	158.2	169.5	100	123.8	267.4	476.0	100
	10	28.6	245.1	241.4	100	125.2	406.4	748.2	100

$q = 4, 8, 10$, and $1,671$ nodes), which is now due to the number of variables of its formulation as ILP decreases as long as s increases. The opposite happens to the heuristics, in such a way that for smaller values of s the heuristics need less computational time than *X*press-MP, but for greater values of s the heuristics need more time.

4.2 Testing GASUB with a set of known test problems

After verifying that GASUB obtains 100% of success in finding the global solution for the previous problems, and that it is also able to find more than one global solution for problems with have several optima, the following set of experiments are aimed to test GASUB with a set of known test problems.

These problems are s-MEDIAN problems whose function values for the optimal solutions are known. In particular we have chosen the Alberta s-MEDIAN test problems (24) and the 40 Beasley s-MEDIAN test problems (23). This choice has been determined by the fact that both sets of test problems have several problems with more than a single global optimum. All problems from the Alberta set are characterized by working with 316 nodes, and the number of facilities to localize (s) varies from 5 to 100. The problems of the Beasley set have different amount of nodes (from 100 to 900 nodes) and the number of facilities to localize (s) ranges from 5 to 200. Our experiments have shown that GASUB always succeed in finding the global

Table 4 Results of GASUB for the Alberta s-MEDIAN problems

s	Opt	Av(T)
5	1	2.3
10	1	100.4
20	2	413.2
30	2	686.1
40	2	1133.1
50	2	1654.8
60	2	1815.5
70	2	2100.6
80	2	2482.0
90	2	2826.0
100	2	2856.0

Table 5 Results of GASUB for the Beasley s-MEDIAN problems

Problem	s	Opt	Av(T)	Problem	s	Opt	Av(T)
1 (100)	5	1	1.8	21 (500)	5	1	18.9
2 (100)	10	2	5.3	22 (500)	10	1	70.8
3 (100)	10	2	5.1	23 (500)	50	2	849.0
4 (100)	20	9	19.9	24 (500)	100	3	1763.8
5 (100)	33	9	45.7	25 (500)	167	6	3445.6
6 (200)	5	1	4.9	26 (600)	5	1	25.3
7 (200)	10	2	12.8	27 (600)	10	1	88.7
8 (200)	20	4	34.6	28 (600)	60	2	1311.0
9 (200)	40	7	96.8	29 (600)	120	1	3509.0
10 (200)	67	10	249.0	30 (600)	200	1	5099.0
11 (300)	5	1	9.4	31 (700)	5	1	31.9
12 (300)	10	1	24.8	32 (700)	10	1	102.4
13 (300)	30	5	214.2	33 (700)	70	2	2237.0
14 (300)	60	9	458.1	34 (700)	140	1	5084.0
15 (300)	100	5	767.6	35 (800)	5	1	38.0
16 (400)	5	1	14.9	36 (800)	10	1	113.7
17 (400)	10	1	50.9	37 (800)	80	1	6843.0
18 (400)	40	4	488.8	38 (900)	5	1	44.9
19 (400)	80	4	994.6	39 (900)	10	1	123.6
20 (400)	133	10	2018.0	40 (900)	90	1	8198.0

solution for all problems from both sets. The results for the Alberta and Beasley sets of test problems are shown in Tables 4 and 5, respectively. As the values of optimal function values are public (see http://www.bus.ualberta.ca/eerkut/testproblems/ and http://people.brun- el.ac.uk/ mastjjb/jeb/orlib/pmedinfo.html), we have not specified them in our tables.

In Table 4, column s indicates the number of new facilities to localize for the problem, column Opt shows the number of global optima obtained when solving the problem, while column Av(T) shows the average computational time in seconds needed to solve the problem. It can be seen that most of the problems have two optima, so there are two combinations of new facilities that obtain the optimal objective function. The computational time increases as s increases (the complexity of the problem increases).

In Table 5, the column *Problem* shows the number of the problem and the number of nodes defined for the problem. It can be seen that for 19 out of 40 problems the number of global optima found by GASUB is greater than 1. In some cases the number of found global optima was 10, which is limited by the input parameter max *_subp_*num, that was set to 10. It means that if the input parameter is changed to a bigger number, it could be possible that GASUB could find more than ten optima. In general, it seems that there exits an increment of the number of global optima found when the number of new facilities increases, in such a way that when s is 5 the number of global optima is always 1 while this number is higher than 3 when s is greater than 10 and the number of nodes is smaller than 500. When the number of nodes is greater than 500, GASUB only finds a global optimum for most of the problems. With respect to the computational time, Table 5 shows that it increases when s and the number of nodes increase i.e., when the complexity of the problem increases.

5 Conclusions and future research

We have shown some basic discrete location problems which can be solved, exactly and approximately, by the same type of algorithms. Then we have proposed a genetic-like heuristic, GASUB, which is able to find a predetermined number of global optima (parameter max*_subp_num*) if they exist. This algorithm has been compared with an optimizer, *X*press-MP, and a well known heuristic used for many location problems, MSH. For this, 15 instances of s-MEDIAN and 75 instances of both MAXCAP and MAXPROFIT have been solved by the three algorithms. Computational experiments show that optimal locations can also be obtained by the heuristics. The GASUB found global optima in all the instances (100% of success) while MSH got trapped in a local optimum in several cases, mainly when solving the MAXCAP problem (50% of success). An additional advantage of GASUB is that this algorithm can generate more than one global optimum (in 27 out of the 75 MAXCAP and 2 out of the 15 s-MEDIAN solved problems), the contrary happens with MSH and *X*press-MP which obtain only one optimum. This characteristic of GASUB is confirmed by solving the known Alberta and Beasley s-MEDIAN test problems, for which this algorithm found several global optima in 28 out of the 55 test problems, obtaining ten global optima (the maximum allowed) in 2 of them.

With respect to the computational cost, it is interesting to remark the different behaviors of the algorithms, mainly for the s-MEDIAN and MAXPROFIT problems. On one hand, the computational time for Xpress-MP decreases when s increases because the number of variables of their formulation as ILP decreases as long as s increases; and on the other hand, the computational time for both heuristics increase significantly when the s value increases, which is explained by the fact that the number of location combination exponentially increases when the parameter s increases. These opposite behaviors explain the fact that for some problems *X*press-MP is faster than the heuristics, and in other cases is the slowest. Anyway, it is remarkable that in general GASUB always is faster than MSH and it also obtains always all the global optima.

In short, the proposed algorithm is able to generate a pre-determined number of global optima to discrete location problems, if they exist; it can solve moderately large problems; it improves the widely used heuristic MSH; and it can compete with the optimizer *X*press-MP when the number of facilities to be located is small.

As future research, a parallelization of GASUB will be investigated in order to reduce computational times when solving large-scale problems.

Acknowledgements This research has been supported by the Ministry of Science and Technology of Spain under the research projects SEJ2005-06273/ECON and TIN2005-00447, in part financed by the European Regional Development Fund (ERDF). We are very grateful with *Dash optimization* for providing a Xpress-MP license for testing.

References

1. Francis, R.L., Lowe, T.J., Tamir, A.: Demand point aggregation for location models. In Drezner Z., Hamacher, H. (eds.) Facility Location: Application and Theory, pp. 207–232. Springer, Berlin (2002)
2. Eiselt, H.A., Laporte, G., Thisse, J.F.: Competitive location models: a framework and bibliography. Transport. Sci. **27** 44–54 (1993)
3. Hamacher H.W., Nickel, S.: Classification of location models. Location Sci. **13**, 229–242 (1998)
4. ReVelle, C.S., Eiselt, H.A. Location analysis: a synthesis and survey. EJOR **165**(1), 1–19 (2005)
5. Daskin, M.S. Network and Discrete location: Models, Algorithms and Applications. Wiley, New York (1995)
6. Mirchandani, P.B.: The p-median problem and generalizations. In: Mirchandani, P.B. Francis, R.L. (eds.), Discrete Location Theory, pp. 55–117. Wiley-Interscience, New York (1990)
7. Serra, D., ReVelle, C.: Competitive location in discrete space. In: Drezner, Z. (ed.), Facility Location: A Survey of Applications and Methods, pp. 367–386. Springer, Berlin (1995)
8. García, M.D., Fernández, P., Pelegrín, B.: On price competition in location-price models with spatially separated markets. TOP **12**(2), 351–374 (2004)
9. Lederer, P. J., Hurter, A.P.: Competition of firms: discriminatory pricing and location. *Econometrica*, **54**(3), 623–640 (1986)
10. Lederer, P.J., Thisse, J.F.: Competitive location on networks under delivered pricing. Opera. Res. Lett. **9**(91), 147–153 (1990)
11. Dorta-González, P., Santos-Peñate, D.R., Suárez Vega, R.: Spatial competition in networks with discriminatory pricing. Pap. Regi. Sci. **84**(2), 271–280 (2005)
12. Fernández, P., Pelegrín, B., García, M.D., Peeters, P.: A discrete long-term location-price problem under the assumption of discriminatory pricing: Formulations and parametric analysis. Eur. J. Opera. Rese. 2006. DOI 10.1016/j.ejor.2005.03.075
13. Berman, O., Krass, D.: Locating multiple competitive facilities: spatial interaction models with variable expenditures. Ann. Oper. Res. **11**, 197–225 (2002)
14. Suárez-Vega, R., Santos-Peñate, D.R., Dorta-González, P.: Discretization and resolution of the $(r \mid X_p)$-medianoid problem involving quality criteria. TOP **12**, 111–134 (2004)
15. Ortigosa, P.M., García, I., Jelasity, M.: Reliability and performance of UEGO, a clustering-based global optimizer. Jo. Global Optim. **19**(3), 265–289 (2001)
16. Cornuejols, G., Fisher, M.L., Nemhauser, G.L.: Locations of bank accounts to optimize float: an analytical study of exact and approximate algorithms. Manage. Sci. **23**, 789–810 (1997)
17. Hakimi, S.L.: On locating new facilities in a competitive environment. Eur. J. Oper. Res. **12**, 29–35 (1983)
18. Colomé, R., Serra, D.: Consumer choice and optimal location models : Formulations and heuristics. Pap. Reg. Sci. **80**, 439–464 (2001)
19. Moreno, J.A., Roda García, J.L., Marcos, J.M. Moreno-Vega, M. A parallel genetic algorithm for the discrete p-median problem. Studies Location. Analy. **7**, 131–141 (1994)
20. Rosing K.E., Revelle, C.: Heuristic concentration: two stage solution construction. Eur. J. Oper. Res. **97**, 75–86 (1997)
21. Xpress-MP. Dash optimization, 2004
22. Pelegrín, B., Fernández, P., Redondo, J.L., García, I., Ortigosa, P.M.: Solving a competitive facility location problem by stochastic algorithms. In: Proceedings of the Tenth International Symposium on Locational Decisions, *ISOLDE X*, pp 217–218. Sevilla, Spain (2005)
23. Alp, O., Drezner, Z., Erkut, E.: An efficient genetic algorithm for the p-median problem. Ann. Oper. Res. **122**(1), 21–42 (2003)
24. Beasley, J.E.: Or-library: distributing test problems by electronic mail. J. Oper. Res. Soc. **41**(11), 1069–1072 (1990)