Solving the Multiple Competitive Facilities Location and Design Problem on the Plane

Juana López Redondo juani@ace.ual.es
Department of Computer Architecture and Electronics, University of Almería, Almería, Spain

José Fernández josefdez@um.es
Department of Statistics and Operations Research, University of Murcia, Murcia, Spain

Inmaculada García inma@ace.ual.es
Department of Computer Architecture and Electronics, University of Almería, Almería, Spain

Pilar M. Ortigosa pilar@ace.ual.es
Department of Computer Architecture and Electronics, University of Almería, Almería, Spain

Abstract

A continuous location problem in which a firm wants to set up two or more new facilities in a competitive environment is considered. Other facilities offering the same product or service already exist in the area. Both the locations and the qualities of the new facilities are to be found so as to maximize the profit obtained by the firm. This is a global optimization problem, with many local optima. In this paper we analyze several approaches to solve it, namely, three multi-start local search heuristics, a multi-start simulated annealing algorithm and two variants of an evolutionary algorithm. Through a comprehensive computational study it is shown that the evolutionary algorithms are the heuristics which provide the best solutions. Furthermore, using a set of problems for which the optimal solutions are known, only the evolutionary algorithms were able to find the optimal solutions for all the instances. The evolutionary strategies presented in this paper can be easily adapted to handle other continuous location problems.

Keywords

Evolutionary algorithm, multimodal optimization, multistart, simulated annealing, continuous location, computational study.

1 Introduction

Location science deals with the location of one or more facilities in a way that optimizes a certain objective (minimization of transportation costs, minimization of social costs, maximization of the market share, etc.) For an introduction to the topic see (Drezner, 1995b; Drezner and Hamacher, 2002; Francis et al., 1992) . In continuous facility location problems the set of possible locations is the plane or a region of the plane. Continuous location theory is very vast, to the point that it has now its own entry (90B85) in the *Mathematics Subject Classification* used by *Mathematical Reviews* and *Zentralblatt für Mathematic*. Whereas some continuous location models are easy to solve (like, for

instance, the classical Weber problem (Weber, 1909)), other more realistic models lead to global optimization problems which are very hard to solve. Some of those difficult problems are the location of undesirable facilities (see Erkut and Neuman (1989); Fernández et al. (2000)), competitive location models (see Drezner (1995a); Fernández et al. (2006)) or the well known p-center problem (see Drezner (1984)), to name a few. These hard to solve problems are sometimes used as test problems to check the efficiency of global optimization methods.

Several global optimization techniques have been proposed to cope with these difficult problems. Among the deterministic methods, the branch-and-bound algorithms have proved to be the most successful, examples of these algorithms are the Big Square Small Square (BSSS) method (Hansen et al., 1985), the Generalized BSSS (Plastria, 1992), the Big Triangle Small Triangle (BTST) (Drezner and Suzuki, 2004) or interval branch-and-bound methods (Fernández et al., 2000; Fernández and Pelegrín, 2001; Fernández et al., 2006). They all are general methods able to solve different location models.

However, those deterministic algorithms are unable to solve some problems: either they are too slow or the computer runs out of memory. Furthermore, with the exception of the interval methods, they have been designed to deal with single facility location problems, i.e., problems in which a single facility is to be located. Thus, heuristic procedures have to be developed to solve those other more difficult problems. Usually, researchers propose ad-hoc heuristics for the specific problem they are handling, although well known meta-heuristics, such as tabu search (Glover and Laguna, 1997), simulated annealing (Kirkpatrick et al., 1983), variable neighborhood search (Hansen and Mladenović, 2001) or ant systems (Dorigo and Di Caro, 1999) have also been applied to different location problems (see Benati and Laporte (1964); Drezner et al. (2002); Yang and Yang (2005)). Genetic algorithm based approaches (Goldberg, 1989; Holland, 1975) are not an exception, and although rather scarce, they have also been applied to solve a few location problems, specially in *discrete* space (see for instance Aytug and Saydam (2002); Jamarillo et al. (2002)). However, the use of genetic or evolutionary algorithms in *continuous* location problems can be qualified as an anecdote: to the extend of our knowledge, apart from our companion paper (Redondo et al., pear), the only published references are (Bhandarkar and Zhang, 1999; Salhi and Gamal, 2003; Houck et al., 1996), and the three deal with the same problem, namely, the uncapacitated location-allocation problem.

In this paper we deal with a multiple competitive facilities location problem presented in (Fernández et al., 2006; Tóth et al., 2006). Competitive location deals with the problem of locating facilities to provide a service (or goods) to the customers (or consumers) of a given geographical area where other facilities offering the same service are already present (or will enter to the market in the near future). Thus, the new facilities will have to *compete* for the market with the existing facilities. Many competitive location models are available in the literature, see for instance the survey papers (Eiselt et al., 1993; Kilkenny and Thisse, 1999; Plastria, 2001) and the references therein. In (Fernández et al., 2006; Tóth et al., 2006), a company wants to enlarge its presence in a given geographical region by opening new facilities. Its objective is the maximization of its *profit*. In the model the *demand* is concentrated at n demand points, whose locations and *buying power* are known. The location and quality of the existing facilities are also known. It is considered that the demand points split their buying power among all the facilities proportionally to the *attraction* they feel for them. The attraction (or utility) function of a customer towards a given facility depends on the distance between the customer and the facility, as well as on other characteristics of the facility

which determine its *quality*. The locations and the qualities of the new facilities are the variables of the problem. The problem is highly nonlinear, with many local optima.

In Fernández et al. (2006) two procedures for solving the single facility case are proposed. The first one is a simple multi-start algorithm in which a local search procedure (namely, a Weiszfeld-like algorithm) is repeated from different starting points. Since the method cannot guarantee that the *global* solution has been found, an interval branch-and-bound algorithm is also proposed. This second procedure can solve the problem with reliability, but it is time-consuming. Recently, in Redondo et al. (pear) we used an evolutionary algorithm, called UEGO (*Universal Evolutionary Global Optimizer*), to solve that single facility competitive location model. In this approach, UEGO itself applies the same Weiszfeld-like algorithm as local optimizer. A comprehensive computational study showed that, (i) using the same computational resources and CPU time, the solutions obtained by UEGO are better than those obtained by the multistart heuristic, for both the objective value and the solution point, and (ii) letting the multistart heuristic use more CPU time than UEGO does not guarantee that the multistart heuristic will find better solutions. Furthermore, with a suitable parameter setting, UEGO is able *always* to obtain the global optimum, taking less time than the reliable interval branch-and-bound method. Additionally, UEGO can solve much larger problems than the interval method.

In Tóth et al. (2006), the case where two facilities are to be located is considered. An interval branch-and-bound method is proposed to solve the problem exactly, but the running time makes the approach inadequate (the required time for solving one problem varies from 6 to 140 hours). In fact, the interval method could not solve problems where more than two facilities have to be located. On the other hand, in Drezner et al. (2002) a related problem is considered, in which the qualities of the facilities are fixed and given in advance (they are not variables of the problem). In that paper, five heuristic procedures are analyzed and the authors conclude that a simulated annealing algorithm (which uses a generalized Weiszfeld algorithm as local search procedure) provides the best solutions (for problems in which up to ten facilities are to be located). In particular, the simulated annealing algorithm is better than two multi-start procedures proposed in Drezner (1998) for that problem.

In this paper we investigate several procedures for solving the multiple competitive facilities location problem introduced in Tóth et al. (2006). In particular, we have extended and adapted the multi-start procedures proposed in Drezner (1998) and the simulated annealing algorithm in Drezner et al. (2002) to cope with our problem, and we have proposed a new way of generalizing the Weiszfeld-like algorithm in Fernández et al. (2006) to cope with the location of more than one facility. Additionally we have adapted the evolutionary algorithm UEGO introduced in Ortigosa et al. (2001) to cope with the multiple facilities location problem.

UEGO is a hybrid multimodal evolutionary algorithm that could also be identified as a memetic clustering algorithm (see Speer et al. (2004)), in the sense that it introduces a local optimizer into the evolution process. In this way, at every generation UEGO performs a local optimizer operation on each individual or subpopulation, and these locally optimal solutions replace the caller individuals. This approach is commonly called *Lamarckian evolution* where the local search can be thought of as being analogous to a kind of learning that occurs during the lifetime of an individual (see Whitley et al. (1994)). Several hybrid evolutionary algorithms have been proposed in the bibliography, where different kinds of optimizers have been used in genetic or evolutionary algorithms. The optimizer could be not only a local search heuristic method, but

also a deterministic (either exact or numerical) optimization method (Brimberg et al., 2000; He et al., 2000). In previous works we have successfully used the stochastic hill climber SASS (Solis and Wets, 1981) as local optimizer. Some of the real applications solved by UEGO using SASS have been the detection of deformable objects in digital images (González-Linares et al., 2000) and the image translational alignment in electron microscopy (Redondo et al., 2004). For the single facility version of the problem considered in this paper we used the Weiszfeld-like algorithm introduced in Fernández et al. (2006) as the local optimizer (see Redondo et al. (pear)). Here, for the multiple facilities case, we have considered different generalizations of the Weiszfeld-like algorithm.

As the computational studies will show, UEGO is the algorithm which provides the best results. Having such a procedure is important for solving the problem at hand, but also as a tool for coping with other more complex related problems, such as the leader-follower (or Stackelberg) problem (Hakimi, 1990). Furthermore, UEGO can easily be adapted to handle many and diverse (location) problems. We hope that this study will foster the use of evolutionary algorithms at solving *continuous* location problems.

The paper is organized as follows. The location model is presented in Section 2. The extensions of the multi-start procedures in Drezner (1998) and the simulated annealing algorithm in Drezner et al. (2002), as well as the new generalized Weiszfeld-like algorithm are given in Section 3. It is in Section 4 where we present the evolutionary algorithm UEGO adapted to the multiple facilities location problem. In Section 5 we present a computational study to analyze the performance of the different algorithms. The paper ends in Section 6 with some conclusions and directions for future research.

2 The multiple competitive facilities location and design problem

A chain wants to locate p new facilities in a given area of the plane, where there already exist m facilities offering the same good or product. The first k of those m facilities belong to the chain, $0 \leq k < m$. The demand, supposed to be inelastic, is concentrated at n demand points, whose locations and buying power are known. The location and quality of the existing facilities are also known.

The following notation will be used throughout:

Indices

i	index of demand points, $i = 1, \ldots, n$.
l	index of existing facilities, $l = 1, \ldots, m$.
j	index of new facilities, $j = 1, \ldots, p$.

Variables

z_j	location of the j-th new facility, $z_j = (x_j, y_j)$.
α_j	quality of the j-th new facility ($\alpha_j > 0$).
nf_j	variables of the j-th facility, $nf_j = (z_j, \alpha_j)$.
$\overline{nf}$	variables of the problem, $\overline{nf} = (nf_1, \ldots, nf_p)$.

Data

p_i	location of the i-th demand point.
w_i	demand (or buying power) at p_i.
f_l	location of the l-th existing facility.
$d^i_{\min}$	minimum distance from p_i at which the new facilities can be located.
d_{il}	distance between p_i and f_l.
α_{il}	quality of f_l as perceived by p_i.
$g_i(\cdot)$	a non-negative non-decreasing function.
$\alpha_{il}/g_i(d_{il})$	attraction that p_i feels for f_l.
γ_i	weight for the quality of the new facilities as perceived by demand point p_i.
S	region of the plane where the new facilities can be located.
$q_{\min}$	minimum allowed quality for the new facilites.
$q_{\max}$	maximum allowed quality for the new facilites.

Miscellaneous

d_{iz_j}	distance between p_i and z_j.
$\gamma_i\alpha_j/g_i(d_{iz_j})$	attraction that p_i feels for z_j.

We may assume that $g_i(d_{il}) > 0 \ \forall i, l$, because any demand point i for which $g_i(d_{il}) = 0$ for some l would be totally lost to the new facilities, so it may simply be left out of the model.

Following Huff's suggestion (Huff, 1964), we consider that the patronizing behavior of customers is probabilistic, that is, demand points split their buying power among all the facilities proportionally to the attraction they feel for them.

With the previous notation, the market share attracted by the p new facilities is

$$\sum_{i=1}^{n} \omega_i \frac{\displaystyle\sum_{j=1}^{p} \frac{\gamma_i\alpha_j}{g_i(d_{iz_j})}}{\displaystyle\sum_{j=1}^{p} \frac{\gamma_i\alpha_j}{g_i(d_{iz_j})} + \sum_{l=1}^{m} \frac{\alpha_{il}}{g_i(d_{il})}}$$

and the total market share attracted by the chain is

$$M(\overline{nf}) = \sum_{i=1}^{n} \omega_i \frac{\displaystyle\sum_{j=1}^{p} \frac{\gamma_i\alpha_j}{g_i(d_{iz_j})} + \sum_{l=1}^{k} \frac{\alpha_{il}}{g_i(d_{il})}}{\displaystyle\sum_{j=1}^{p} \frac{\gamma_i\alpha_j}{g_i(d_{iz_j})} + \sum_{l=1}^{m} \frac{\alpha_{il}}{g_i(d_{il})}}$$

The problem to be solved is then

$$\begin{cases} \max & \Pi(\overline{nf}) = F(M(\overline{nf})) - \sum_{j=1}^{p} G(nf_j) \\ \text{s.t.} & d_{iz_j} \geq d^i_{\min},\ i = 1, \ldots, n, j = 1, \ldots, p \\ & \alpha_j \in [q_{\min}, q_{\max}],\ j = 1, \ldots, p \\ & z_j \in S \subset \mathbb{R}^2,\ j = 1, \ldots, p \end{cases} \tag{1}$$

where $\Pi(\overline{nf})$ is the profit obtained by the chain, obtained as the total income minus the costs. This profit disregards the operating costs of the existing facilities of the own chain, since these are considered to be constant. $F(M)$ is a strictly increasing differentiable function which transforms the market share M into expected sales. $G(nf_j)$ is

a differentiable function which gives the operating cost of a facility located at z_j with quality α_j.

The function F will often be linear, $F(M) = c \cdot M$, where c is the income per unit of good sold. Of course, other functions can be more suitable depending on the real problem considered. The function $G(nf_j)$ should increase as z_j approaches one of the demand points, since it is rather likely that around those locations the operational cost of the facility will be higher (due to the value of land and premises, which will make the cost of buying or renting the location higher). On the other hand, G should be a nondecreasing and convex function in the variable α_j, since the more quality we expect from of the facility, the higher the costs will be, at an increasing rate. We will assume G to be separable, i.e. of the form $G(nf_j) = G_1(z_j) + G_2(\alpha_j)$. Possible expressions for G_1 may be

$$G_1(z_j) = \sum_{i=1}^{n} \Phi_i(d_{iz_j})$$

with

$$\Phi_i(d_{iz_j}) = w_i/((d_{iz_j})^{\phi_{i0}} + \phi_{i1}), \;\; \phi_{i0}, \phi_{i1} > 0,$$

or

$$\Phi_i(d_{iz_j}) = w_i/(e^{\frac{d_{iz_j}}{\phi_{i0}}} - 1 + \phi_{i1}), \;\; \phi_{i0}, \phi_{i1} > 0.$$

A few typical forms for G_2 might be

$$G_2(\alpha_j) = (\alpha_j/\xi_0)^{\xi_1}, \xi_0 > 0, \xi_1 \geq 1,$$

or

$$G_2(\alpha_j) = e^{\frac{\alpha_j}{\xi_0} + \xi_1} - e^{\xi_1}, \;\; \xi_0 > 0, \xi_1 \in \mathbb{R}.$$

The model is rather general and can be specialized for many real situations. Of course, its ability to model general situations comes at the expense of the difficulty in its solution: the problem is neither concave nor convex. For the simpler model in which the qualities of the new facilities are given in advance (they are not variables of the problem), the operating costs of the facilities are disregarded (i.e., $G = 0$), and no constraints are taken into account, the number of local optima in a given problem with $n = 100$, $m = 7$, $k = 0$ were 11, 470 and 23714 for $p = 1$, 2 and 3, respectively, and more than 90000 for $p = 4$ (see Drezner et al. (2002)).

3 Extensions of previous algorithms and a new generalized Weiszfeld-like algorithm

A local method is proposed in Fernández et al. (2006) (see also Redondo et al. (pear)) for solving the single facility case ($p = 1$) of problem (1). The algorithm is a steepest descent type method which takes discrete steps along the search paths and, in the best case, converges to a local optimum. In the method, the derivatives of the objective function are equated to zero and the next iterate is obtained by implicitly solving these equations. In Location Science these types of methods are known as Weiszfeld-like methods, in honor to E. Weiszfeld, who first proposed that strategy (Weiszfeld, 1937).

If we denote by $nf_1 = (z_1, \alpha_1) = (x_1, y_1, \alpha_1)$ the location and quality of the single

new facility to be located, we set

$$\begin{aligned} r_i &= \sum_{l=1}^{m} \frac{\alpha_{il}}{g_i(d_{il})}, \;\; t_i = w_i \sum_{l=k+1}^{m} \frac{\alpha_{il}}{g_i(d_{il})}, \\ H_i(nf_1) &= \frac{\partial \Pi}{\partial d_{iz_1}} = -\frac{\mathrm{d}F}{\mathrm{d}M} \cdot \frac{\alpha_1 \gamma_i t_i g_i'(d_{iz_1})}{(\gamma_i \alpha_1 + r_i g_i(d_{iz_1}))^2} - \frac{\mathrm{d}\Phi_i}{\mathrm{d}d_{iz_1}} \end{aligned} \tag{2}$$

and d_{iz_1} is a distance function such that

$$\frac{\partial d_{iz_1}}{\partial x_1} = x_1 A_{i1}(z_1) - B_{i1}(z_1), \;\; \frac{\partial d_{iz_1}}{\partial y_1} = y_1 A_{i2}(z_1) - B_{i2}(z_1),$$

then the Weiszfeld-like algorithm for solving (1) when $p = 1$ is described in Algorithm 1 (for more details see Fernández et al. (2006)).

Algorithm 1: Weiszfeld-like algorithm WLM1

1 Set iteration counter $r = 0$
2 Initialize $nf_1^{(0)} = (x_1^{(0)}, y_1^{(0)}, \alpha_1^{(0)})$
3 WHILE Stopping criteria are not met DO
4 Update $nf_1^{(r+1)} = (x_1^{(r+1)}, y_1^{(r+1)}, \alpha_1^{(r+1)})$ as follows:

$$x_1^{(r+1)} = \frac{\sum_{i=1}^{n} H_i(nf_1^{(r)}) B_{i1}(z_1^{(r)})}{\sum_{i=1}^{n} H_i(nf_1^{(r)}) A_{i1}(z_1^{(r)})}$$

$$y_1^{(r+1)} = \frac{\sum_{i=1}^{n} H_i(nf_1^{(r)}) B_{i2}(z_1^{(r)})}{\sum_{i=1}^{n} H_i(nf_1^{(r)}) A_{i2}(z_1^{(r)})}$$

and $\alpha_1^{(r+1)}$ as a solution of the equation:

$$\frac{\mathrm{d}F}{\mathrm{d}M} \cdot \sum_{i=1}^{n} \frac{\gamma_i t_i g_i(d_{iz_1^{(r+1)}})}{(\gamma_i \alpha_1 + r_i g_i(d_{iz_1^{(r+1)}}))^2} - \frac{\mathrm{d}G_2}{\mathrm{d}\alpha_1} = 0$$

5 IF $nf_1^{(r+1)}$ is infeasible THEN
6 $nf_1^{(r+1)} \in [nf_1^{(r)}, nf_1^{(r+1)}] \cap \partial S$
7 $r = r + 1$

Two stopping criteria determine the end of Algorithm 1. For the first stopping criterion, the algorithm stops if $\|(x_1^{(r)}, y_1^{(r)}) - (x_1^{(r+1)}, y_1^{(r+1)})\|_2 < \epsilon_1$ and $|\alpha_1^{(r)} - \alpha_1^{(r+1)}| < \epsilon_2$, for given tolerances $\epsilon_1, \epsilon_2 > 0$. The second stopping criterion sets a maximum number of iterations $r_{\max}$; it is needed because the convergence of the algorithm cannot be guaranteed. In Step 6 of Algorithm 1 $nf_1^{(r+1)}$ is set equal to a point in the segment $[nf_1^{(r)}, nf_1^{(r+1)}]$ which is also on the border of the feasible region S.

It is important to note that the Weiszfeld-like algorithm WLM1 is a *local* procedure. Thus, in the best case, the algorithm ends at a local maximum (not necessarily at the global one). Thus, in order to have a good chance of finding the optimal solution, in Fernández et al. (2006) it is suggested to apply the algorithm repeatedly using different starting points in Step 2, and then select the solution that obtains the maximum profit. However, this still does not guarantee that a global optimal solution has been found. In Fernández et al. (2006) it is proposed to use that multi-start technique starting from 100 or 1000 different points.

3.1 Two multistart algorithms

Following Drezner (1998), there are two easy ways of using the Weiszfeld-like algorithm WLM1 for solving the multifacility case ($p > 1$). The rationale behind them both is the same: the new variable values for one facility at a time are found while holding all the other facilities fixed in their current values. Once this is accomplished, the optimized facility is fixed and another facility's variables are optimized. One iteration consists of finding the new values, one at a time, for all p facilities.

In the first approach, which we denote by WLMa, the best locations and qualities for the facilities, one at a time, are found, by applying the complete while loop of WLM1. In the second approach, denoted by WLMb, p locations and qualities for the p new facilities are randomly generated. Then, only one iteration of the while loop of WLM1 is applied, in turn, for each new facility.

Algorithm 2: Algorithm WLMa (resp. WLMb)

1 Set iteration counter $r = 0$
2 Set the current best objective value $\tilde{\Pi} = -\infty$
3 FOR $t = 1$ to 200 DO
4 Generate a random feasible vector $\overline{nf}^{(0t)}$
5 IF $\Pi(\overline{nf}^{(0t)}) > \tilde{\Pi}$ THEN
6 $\tilde{\Pi} = \Pi(\overline{nf}^{(0t)})$ and $\overline{nf}^{(0)} = \overline{nf}^{(0t)}$
7 WHILE Stopping criteria are not met DO
8 FOR $j = 1$ to p DO
9 Relocate the j-th facility $nf_j^{(r)}$ applying Step 3 of Algorithm 1 (resp. one iteration of the while loop in Step 3 of Algorithm 1) holding the other $p - 1$ facilities fixed in their values
10 $r = r + 1$

In Step 3 of Algorithm 2, we generate and evaluate random points in order to obtain a relatively good starting point. After extensive experiments we found that 200 random points is sufficient for these algorithms. The first stopping criterion of Algorithm 2 is based on the distance between two vectors from consecutive iterations. Two consecutive iteration vectors $\overline{nf}^{(r)}$ and $\overline{nf}^{(r+1)}$ are considered closed enough if $\|(x_j^{(r)}, y_j^{(r)}) - (x_j^{(r+1)}, y_j^{(r+1)})\|_2 < \epsilon_1$ and $|\alpha_j^{(r)} - \alpha_j^{(r+1)}| < \epsilon_2$ for all $j = 1 \ldots, p$, for given tolerances $\epsilon_1, \epsilon_2 > 0$. The second stopping rule sets a limit $r_{\max}$ on the maximum number of iterations. Notice that Algorithms WLMa and WLMb only differ in Step 9.

As with WLM1, it is recommended to execute these algorithms many times to have more chances at finding the optimal solution. The corresponding multi-start algorithms will be denoted mWLMa and mWLMb, respectively.

3.2 A simulated annealing algorithm

We now propose an extension of the heuristic H-5 in Drezner et al. (2002) which is able to handle the multiple facilities location problem. Initially (Steps 1 to 11), the optimal solution is restricted to a set of points rather than to the whole space, and a simulated annealing procedure is applied. A rectangular grid of possible locations and qualities for the facilities is selected. For each point of the grid (i.e., for a possible location and quality of a new facility) we define the set of neighboring points. In our rectangular grid, each point (which is not on the boundary of the grid) has 26 neighbors: eight with the same quality (four in the East, West, North and South directions, and four in diagonal directions), nine with a higher (in addition the eight described before, now we can also use the same location for the facility and change only the quality) and nine with a lower quality. Details are given in Algorithm 3. We use three parameters for the cooling schedules: the starting temperature T_0, the rate at which the temperature is lowered ρ ($\rho < 1$), and the limit on the number of iterations $r_{\max}$. The actual values of parameters T_0 and ρ were those in Drezner et al. (2002), namely, $T_0 = 1$ and $\rho = 1 - 5/r_{\max}$. However the parameter $r_{\max}$ was set to $r_{\max} = 5000 \cdot p$ instead of $r_{\max} = 1000 \cdot p$. The reason for this increasse is that in our problem the quality should be computed as a part of the solution (it is not fixed, as happened in the problem described in Drezner et al. (2002)). As a consequence, the grid used is three-dimensional (and not bi-dimensional). Since the range of the quality has been sampled 5 times, we work with 5 bi-dimensional grids as described in Drezner et al. (2002).

Algorithm 3: Algorithm SA

1 Select a starting random configuration on the grid
2 Set iteration counter $r = 0$
3 Select a starting temperature T_0 and a reduction rate for the temperature, ρ
4 WHILE $r < r_{\max}$ DO
5 Select a facility and one of its neighbors at random (this defines a random move), and evaluate the change in the objective function when we exchange the selected facility for its neighbor
6 IF the random move improves the value of the objective function THEN
7 Accept the move
8 ELSE
9 Compute $\delta = \Delta\Pi/T_r$, where $\Delta\Pi$ is the objective function value decrease
10 Accept the move with probability $e^{-\delta}$ (if the move is not accepted, the current solution is not changed)
11 $T_{r+1} = \rho T_r$, $r = r + 1$
12 Apply Step 8 of WLMb starting with the best solution encountered so far

After the simulated annealing procedure, WLMb is used to fine-tune the solutions (Step 12). The multi-start algorithm which repeats SA will be denoted by mSA. For the simpler problem considered in Drezner (1998) (with qualities given beforehand, and no constraints) mWLMb proved to be superior to mWLMa (see Drezner (1998)), and mSA superior in turn to mWLMb (and to other 3 different heuristics, see Drezner et al. (2002)).

3.3 A new generalized Weiszfeld-like algorithm

Algorithms WLMa and WLMb solve the multifacility problem using the single facility Weiszfeld-like algorithm WLM1. In this subsection we describe a Weiszfeld-like method specifically designed for the multifacility problem.

The total market share captured by the chain can be rewritten as

$$M(\overline{nf}) = \sum_{i=1}^{n} w_i - \sum_{i=1}^{n} \omega_i \frac{\displaystyle\sum_{l=k+1}^{m} \frac{\alpha_{il}}{g_i(d_{il})}}{\displaystyle\sum_{j=1}^{p} \frac{\gamma_i \alpha_j}{g_i(d_{iz_j})} + \sum_{l=1}^{m} \frac{\alpha_{il}}{g_i(d_{il})}}$$

and setting $W = \sum_{i=1}^{n} w_i$ and using (2), we finally have

$$M(\overline{nf}) = W - \sum_{i=1}^{n} \frac{t_i}{\displaystyle\sum_{j=1}^{p} \frac{\gamma_i \alpha_j}{g_i(d_{iz_j})} + r_i}.$$

In what follows, we assume that

$$G(nf_j) = \sum_{i=1}^{n} \Phi_i(d_{iz_j}) - G_2(\alpha_j).$$

A necessary condition for a vector to be a local (or global) maximum of the objective function

$$\Pi(\overline{nf}) = F(M(\overline{nf})) - \textstyle\sum_{j=1}^{p} \sum_{i=1}^{n} \Phi_i(d_{iz_j}) - \sum_{j=1}^{p} G_2(\alpha_j)$$

is that the partial derivatives of Π at that vector must vanish.

- With regard to the variable x_s we have that

$$\frac{\partial \Pi}{\partial x_s} = 0 \Longleftrightarrow \frac{\mathrm{d}F}{\mathrm{d}M} \cdot \frac{\partial M}{\partial x_s} - \sum_{i=1}^{n} \frac{\mathrm{d}\Phi_i}{\mathrm{d}d_{iz_s}} \frac{\partial d_{iz_s}}{\partial x_s} = 0.$$

Doing some calculations the previous expression is equivalent to

$$\frac{\mathrm{d}_F}{\mathrm{d}_M} \cdot \sum_{i=1}^{n} \frac{-\alpha_s \gamma_i t_i g_i'(d_{iz_s})}{\left(\displaystyle\sum_{j=1}^{p} \frac{\gamma_i \alpha_j}{g_i(d_{iz_j})} + r_i\right)^2 (g_i(d_{iz_s}))^2} \frac{\partial d_{iz_s}}{\partial x_s} - \sum_{i=1}^{n} \frac{\mathrm{d}\Phi_i}{\mathrm{d}d_{iz_s}} \frac{\partial d_{iz_s}}{\partial x_s} = 0.$$

which can be rewritten as

$$\sum_{i=1}^{n} H_{is}(\overline{nf}) \frac{\partial d_{iz_s}}{\partial x_s} = 0$$

where

$$H_{is}(\overline{nf}) = -\frac{\mathrm{d}F}{\mathrm{d}M} \frac{\alpha_s \gamma_i t_i g_i'(d_{iz_s})}{\left(\displaystyle\sum_{j=1}^{p} \frac{\gamma_i \alpha_j}{g_i(d_{iz_j})} + r_i\right)^2 (g_i(d_{iz_s}))^2} - \frac{\mathrm{d}\Phi_i}{\mathrm{d}d_{iz_s}}.$$

Furthermore, if d_{iz_s} is a distance function such that

$$\frac{\partial d_{iz_s}}{\partial x_s} = x_s A_{i1s}(z_s) - B_{i1s}(z_s), \tag{3}$$

where $A_{i1s}(z_s)$ and $B_{i1s}(z_s)$ are functions of z_s, then

$$\frac{\partial \Pi}{\partial x_s} = 0 \Longleftrightarrow x_s = \frac{\sum_{i=1}^{n} H_{is}(\overline{nf}) B_{i1s}(z_s)}{\sum_{i=1}^{n} H_{is}(\overline{nf}) A_{i1s}(z_s)}.$$

- Analogously, if

$$\frac{\partial d_{iz_s}}{\partial y_s} = y_s A_{i2s}(z_s) - B_{i2s}(z_s), \tag{4}$$

then

$$\frac{\partial \Pi}{\partial y_s} = 0 \Longleftrightarrow y_s = \frac{\sum_{i=1}^{n} H_{is}(\overline{nf}) B_{i2s}(z_s)}{\sum_{i=1}^{n} H_{is}(\overline{nf}) A_{i2s}(z_s)}.$$

- Finally,

$$\frac{\partial \Pi}{\partial \alpha_s} = 0 \Longleftrightarrow \frac{\mathrm{d}F}{\mathrm{d}M} \cdot \sum_{i=1}^{n} \frac{\gamma_i t_i}{\left(\sum_{j=1}^{p} \frac{\gamma_i \alpha_j}{g_i(d_{iz_j})} + r_i \right)^2 g_i(d_{iz_s})} - \frac{\mathrm{d}G_2}{\mathrm{d}\alpha_s} = 0. \tag{5}$$

Notice that (5), when we fix $z_j \ \forall j$ and $\alpha_j \ \forall j \neq s$, has just one variable, α_s. Thus we can solve it by using any algorithm for solving equations of a single variable. Furthermore, Π, as a function of the α_s variable alone, is concave: regardless of the strictly increasing differentiable function $F(\cdot)$ considered, the second derivative of Π with regard to α_s is negative.

Among the distance functions that satisfy the conditions (3) and (4) we have the inflated Euclidean distance or its rescaled version, the l_{2b} norm, given by

$$d_{iz_s} = \sqrt{b_1(x_s - p_{i1})^2 + b_2(y_s - p_{i2})^2},$$

where $b_1, b_2 > 0$ are given parameters (see Fernández et al. (2002)). In this case

$$A_{i1s} = \frac{b_1}{d_{iz_s}}, \quad B_{i1s} = \frac{p_{i1} b_1}{d_{iz_s}},$$
$$A_{i2s} = \frac{b_2}{d_{iz_s}}, \quad B_{i2s} = \frac{p_{i2} b_2}{d_{iz_s}}.$$

Thus, a new generalized Weiszfeld-like algorithm can be constructed (Algorithm 4, WLMc).

The stopping rules are the same as in algorithm WLMa and WLMb. Again, we will use WLMc in a multi-start strategy, which will be denoted by mWLMc.

Algorithm 4: Algorithm WLMc

1 Set iteration counter $r = 0$
2 Set the current best objective value $\tilde{\Pi} = -\infty$
3 FOR $t = 1$ to 200 DO
4 Generate a random feasible vector $\overline{nf}^{(0t)}$
5 IF $\Pi(\overline{nf}^{(0t)}) > \tilde{\Pi}$ THEN
6 $\tilde{\Pi} = \Pi(\overline{nf}^{(0t)})$ and $\overline{nf}^{(0)} = \overline{nf}^{(0t)}$
7 WHILE Stopping criteria are not met DO
8 FOR $j = 1$ to p DO
9 Update $nf_1^{(r+1)} = (x_1^{(r+1)}, y_1^{(r+1)}, \alpha_1^{(r+1)})$ as follows:

$$x_j^{(r+1)} = \frac{\sum_{i=1}^{n} H_{ij}(\overline{nf}^{(r)}) B_{i1j}(z_j^{(r)})}{\sum_{i=1}^{n} H_{ij}(\overline{nf}^{(r)}) A_{i1j}(z_j^{(r)})}$$

$$y_j^{(r+1)} = \frac{\sum_{i=1}^{n} H_{ij}(\overline{nf}^{(r)}) B_{i2j}(z_j^{(r)})}{\sum_{i=1}^{n} H_{ij}(\overline{nf}^{(r)}) A_{i2j}(z_j^{(r)})},$$

and $\alpha_j^{(r+1)}$ is obtained as a solution of the equation:

$$\frac{\mathrm{d}F}{\mathrm{d}M} \cdot \sum_{i=1}^{n} \frac{\gamma_i t_i}{\left(\sum_{s=1}^{p} \frac{\gamma_i \alpha_s}{g_i(d_{iz_s^{(r)}})} + r_i \right)^2 g_i(d_{iz_j^{(r)}})} - \frac{\mathrm{d}G_2}{\mathrm{d}\alpha_j} = 0,$$

10 IF $nf_1^{(r+1)}$ is infeasible THEN
11 $nf_1^{(r+1)} \in [nf_1^{(r)}, nf_1^{(r+1)}] \cap \partial S$
12 $r = r + 1$

4 Universal Evolutionary Global Optimizer (UEGO)

UEGO is a multimodal algorithm which is able both to solve multimodal optimization problems where the objective function has multiple local optima and to discover the structure of these optima as well as the global optimum. In a multimodal domain, each peak can be thought of as a niche. The analogy with nature is that within the environment there are different subspaces, niches, that can support different types of life (species or organisms). Niching or clustering methods are techniques that promote the formation and maintenance of subpopulations in the Evolutionary Algorithms (EA). A niching method must be able to form and maintain multiple, diverse, final solutions, either of identical fitness or of varying fitness. These mechanisms allow an EA to identify multiple optima in a multi modal domain, and not just a single global optimum. UEGO is a clustering method that offers a solution to the so-called niche radius problem

which is a common problem of many simple niching techniques such as *fitness sharing* (Deb, 1989), *simple iteration* or the *sequential niching* (Beasley et al., 1993). This problem is related to functions that have multiple local optima and whose optima are unevenly spread throughout the search space. With such functions the *niche radius* cannot be set correctly since if it is too small the search becomes ineffective and if it is too large those local optima that are too close to each other cannot be distinguished. The solution of UEGO involves a 'cooling' technique which enables the search to focus on the promising regions of the space, starting off with a relatively large radius that decreases as the search proceeds.

Another important characteristic of UEGO is that it is not necessary to know the number of niches or clusters in advance. This is not something trivial, and although some algorithms do not do the total number of clusters either (Ma et al., 2006), many popular niching algorithms need to know it. In UEGO every cluster is intended to occupy a local maximizer of the fitness function, without knowing the total number of local maximizers in the fitness landscape. This means that when the algorithm starts it does not know how many clusters there will be at the end. For this purpose, UEGO uses a non-overlapping set of clusters which define sub-domains of the search space. As the algorithm progresses, the search process can be directed toward smaller regions by creating new sets of non-overlapping clusters defining smaller sub-domains. This process is a kind of cooling method similar to simulated annealing. A particular cluster is not a fixed part of the search domain; it can move through the space as the search proceeds. UEGO is abstract in the sense that the 'cluster-management' and the cooling mechanism have been logically separated from the actual local optimization algorithm. Therefore it is possible to implement any kind of optimizer to work inside a cluster.

4.1 Algorithmic description of UEGO

As mentioned above, the 'cluster-management' is one of the core parts of UEGO. It consists of procedures for creating, fusing and eliminating clusters during the whole optimization process. The concept of cluster, which is fundamental in UEGO, is renamed *species*. A species can be thought of as a window on the whole search space (see Figure 1). This window is defined by its center and a radius. The center is a solution, and the radius is a positive number. In particular, for the multiple facilities location problem, a species is an array of the form $(\overline{nf}, \Pi(\overline{nf}), R)$ (we also store information about the objective value at the center of the species). Taking into account that the center is a solution, a species could be equivalent to the concept of individual in a standard evolutionary algorithm. However, a species has an attraction area, defined by its radius, that covers a region of the search space and hence multiple solutions, so a species would be equivalent to a subpopulation or cluster in an evolutionary algorithm based on subpopulations (clustering algorithm).

The main role of the window is to define the search region for the local optimizer. If the value of a new solution found by the optimizer is better than that of the old center, the new solution becomes the center and the window is moved while it keeps the same radius value (see Figure 1).

As it has been previously mentioned, in UEGO every species (cluster, subpopulation) is intended to determine a local maximizer of the fitness function, without knowledge about the total number of local maximizers in the fitness landscape. It means that when the algorithm starts it does not known how many species there will be at the end. This implies that the number of species cannot be fixed in advance, and the introduction of the concepts of creation and fusion of species is necessary. At the beginning, a

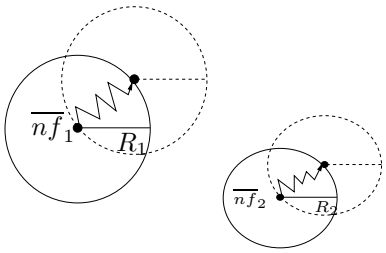

Figure 1: Optimization procedure

single species exists (the root species), and as the algorithm evolves and applies genetic operators new species can be created, and some of the existing species can be fused. Given two parents (species), a new species is created when it is likely that the parents are on different hills.

Another special characteristic of UEGO is the radius of the species, which is neither constant along the execution of UEGO nor the same for each species. This radius is a monotonous function that decreases as the index level (or cycles or generations) increases. The parameter *levels* indicates the maximum number of levels in the algorithm, i.e. the number of different 'cooling' stages. At each level the function radius does not change. The radius function is defined in such a way that it coincides with the initial domain landscape at the first level, and it converges to zero when the number of levels tends to infinite.

The radius function is used to control the speed of cooling. In fact, it gives the temperature of the system in a given level. The radius of a species created at level i, is given by the following exponential function, where R_L and R_1 are the given (input parameter) smallest and largest radii, respectively:

$$R_i = R_1 \left(\frac{R_L}{R_1} \right)^{\frac{i-1}{L-1}} \qquad (i = 2, \ldots, L).$$

Every time a new species is created, it will be associated with a radius, whose value depends on the current level. In this way, species which have been created on different levels, have different radii. The species created at the highest levels have the smallest radius. In this way, species with small radii behave as if they were cooler, they discover a relatively small area, their motion in the space is slower but they are able to differentiate between local optima that are relatively close to each other.

During the optimization process, a list of species is kept by UEGO. This concept, species-list, would be equivalent to the term population in an evolutionary algorithm. UEGO is in fact a method for managing this *species-list* (i.e. creating, deleting and optimizing species). The maximum length of the species list is given by the input parameter *max_spec_num* (maximum population size).

As previously mentioned, this algorithm not only creates new species, but also fuses some of them. The fusion depends on the level at which the algorithm is each time. Each level i determines a different radius (R_i). When the algorithm is in a fusion stage, it unites species of the species list whose centers are closer than the distance defined by R_{level}. In this way, for low levels most of species can be fused, and for high levels (when the radius is smaller) only a few species can be fused, and then the deceptive points can be detected. If two species are fused, the center of the new species

is the center with maximum objective function value, and the assigned level will be the minimum level of both fused species. In view of the species list, the species with the lower level absorbs the other. It is interesting to notice that the fusion only implies the computation of distances between the centers of all species.

A global description of UEGO is given in Algorithm 5

Algorithm 5: Algorithm UEGO

```
1 Set iteration counter r = 0
2 Optimize_species(n_1)
3 FOR i = 2 to levels
4    Determine R_i, new_i, n_i
5    Create_species(creat_evals)        # creat_evals= new_i/length(species_list)
6    Fuse_species(R_i)
7    Shorten_species_list
8    Optimize_species(opt_evals)        # opt_evals= n_i/max_spec_num
9    Fuse_species(R_i)
```

Every level i (with $i \in [1,$*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 species) and n_i (maximum f.e. allowed when optimizing individual species). In the following, the different key stages in the algorithm are described:

- *Init_species_list:* A new species list consisting of one species with a random center at level 1 is created. Taking into account that the radius associated at level 1 is the diameter of the search space, this species always covers the whole search space.

- *Create_species(*$create\_evals$*):* For every species on the list, random trial points in the 'window' of the species are created, and for every pair of trial points the objective function is evaluated at the middle of the *segment* connecting the pair and at the trial points (see Figure 2). The parameter $create\_evals$ indicates the maximum number of function evaluations that a species can make in this procedure. This parameter is computed dividing the number of function evaluations required for the creation mechanism at level i (new_i) by the number of existing species (length($species\_list$).

  If the objective function value at the midpoint is worse than the values at the trial points, then the members of the pair are inserted in the species list. Otherwise, the midpoint is inserted. The motivation behind this method is to create species that are on different 'hills' so ensuring that there is a valley between the new species. Every new inserted species is assigned the current level value (i). As a result of this procedure the species list will eventually contain several species with different levels (hence different radii).

  The parameter of this procedure (*evals*) is an upper bound on the number of function evaluations.

  In terms of evolutionary computation, this procedure can be interpreted as a mechanism to create offspring, where a species generate new species that are within its attraction area, though later on the new species can move toward new regions.

- *Fuse_species(radius):* If the centers of any pair of species from the species list are closer to each other than the given radius, the two species are fused (see Figure 3).

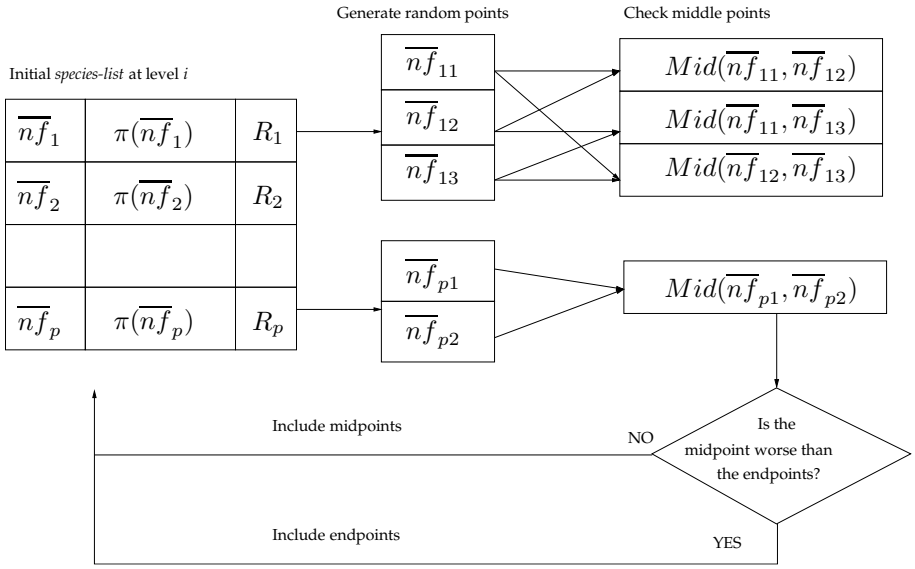

Figure 2: Creation procedure

The center of the new species will be the one with the best function value while the level will be the minimum of the levels of the original species (so the radius will be the largest one).

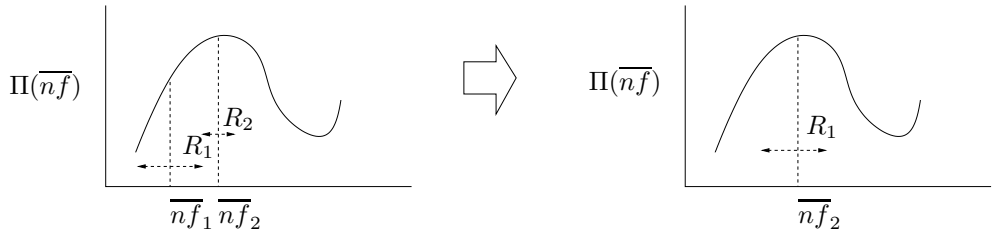

Figure 3: Fusion procedure

- *Shorten_species_list:* It deletes species to reduce the list length to *max_spec_num* value. Higher level species are deleted first, therefore species with larger radii are always kept. In particular, one species at level 1 (whose radius is equal to the diameter of the search domain) always exists, making it possible to escape from local optima.

  In terms of evolutionary computation, this procedure can be interpreted as a selection mechanism, where species with bigger radii are chosen first.

- *Optimize_species(opt_evals):* It executes the optimizer for every species with a given number of evaluations (*budget_per_species*) (see Figure 1). This budget is computed dividing the maximum number of function evaluations (n_i) allowed by the optimization procedure at level i, by the maximum number of species or maximum population size (max_spec_num). This means that if the current number of species (length($species\_list$)) at level i is smaller than the maximum allowed, the num-

ber of function evaluations is smaller than max_spec_num. Additionally, the optimizer called by a species may use fewer function evaluations than the maximum passed as argument (opt_evals).

Note that UEGO terminates when it has executed all of its levels. Therefore the final number of function evaluations depends on the complexity of the problem at hand.

4.2 The Weiszfeld-like algorithms within UEGO

For solving the multiple facilities location problem we propose the use of the Weiszfeld-like algorithms WLMa or WLMb as local optimizer during the *Optimize_species* process (see Step 8 of Algorithm 5), where UEGO tries to optimize every species from its species list. During this process, every species calls the corresponding Weiszfeld-like algorithm once, using the center of the caller species as starting point. The Weiszfeld-like algorithms finish when one of the stopping criteria of the algorithms is satisfied. If during an execution of the Weiszfeld-like algorithm a new point with a better objective function is found, then this new point becomes the center of the species.

As a consequence, at the end of the optimization process, the species from the species list can move toward the locations of the local optima. The implementations which use WLMa and WLMb as local search methods will be called UEGOa and UEGOb, respectively.

4.3 UEGO parameter setting

In UEGO the most important parameters are those defined at each level: the radii (R_i) and the maximum number of function evaluations for species creation (new_i) and optimization (n_i). These parameters are computed from some user-given parameters that are easier to understand:

- *evals* (N): The maximum number of function evaluations for the whole optimization process. Note that the actual number of function evaluations executed by UEGO is usually less than this value.
- *levels* (L): The maximum number of levels, i.e., the number of cooling stages.
- *max_spec_num* (M): The maximum length of the species list or the maximum population size allowed.
- *min_R* (R_L): The radius that is associated with the maximum level, i.e., *levels*.

In Ortigosa et al. (2001) an analysis of the effects of the different parameters of UEGO was made in order to obtain a robust parameter setting. From the experiments made in that work, 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 species (M) and a large value of evals (N) in order to get a minimum budget per species which is sufficient in the optimization process. As an example of robust parameter setting for real problems defined in $[0,1]^n$ domains, the setting $L = 10$, $R_L = 0.03$, $M = 50$ or 100 and $N = 1000000$ was proposed in Ortigosa et al. (2001). In that paper, some guidelines to fine-tune the parameters depending on the problem to be solved were also proposed.

In order to find a robust parameter setting for our competitive location and design problem we followed those guidelines. Taking into account that in our computational studies the location variables are defined in the interval $[0,10]^{2p}$ and the qualities in $[0.5,5]^p$, we decided to increase the parameter R_L to 0.25.

We know that UEGO does not become trapped in local optima only if the number of levels is high enough, following that the cooling process allows the algorithm to escape from local optima. Thus, we also increased the number of levels (L).

As for the parameters N and M, we know that the number of function evaluations is limited by N and the number of detected species is limited by M. Taking into account that the number of local optima is very high for our multifacility problem, mainly when the number of new facilities to be located is high, we have decided to tune the parameter M to 750 and the parameter N to 20000000.

Therefore, after extensive experiments, we found that a good parameter setting for both UEGOa and UEGOb is $L = 15$, $R_L = 0.25$, $M = 750$ and $N = 20000000$.

5 Computational studies

All the computational results in this paper have been obtained under Linux on a Xeon IV with 2.4GHz CPU and 1GB memory. The algorithms have been implemented in C++.

5.1 The test problems

In order to have an overall view on the performance of the algorithms, we have generated different types of problems, varying the number n of demand points, the number m of existing facilities and the number k of those facilities belonging to the chain. The actual settings (n, m, k) employed in the different computational studies will be detailed later on.

For every setting, the problems were generated by randomly choosing the parameters of the problems uniformly within the following intervals:

- $p_i, f_l \in S = ([0, 10], [0, 10])$,
- $\omega_i \in [1, 10]$,
- $\gamma_i \in [0.75, 1.25]$,
- $\alpha_{il} \in [0.5, 5]$,
- $c \in [1, 2]$, the parameter for $F(M(\cdot)) = c \cdot M(\cdot)$,
- $G(nf_j) = \sum_{i=1}^{n} \Phi_i(d_{iz_j}) + G_2(\alpha_j)$ where
  - $\Phi_i(d_{iz_j}) = w_i \frac{1}{(d_{iz_j})^{\phi_{i0}+\phi_{i1}}}$ with $\phi_{i0} = \phi_0 = 2$ and $\phi_{i1} \in [0.5, 2]$.
  - $G_2(\alpha_j) = e^{\frac{\alpha_j}{\xi_0}+\xi_1} - e^{\xi_1}$ with $\xi_0 \in [7, 9]$ and $\xi_1 \in [4, 4.5]$.
- $d_{iz_j} = \sqrt{b_1(x_j - p_{i1})^2 + b_2(y_j - p_{i2})^2}$, with $b_1, b_2 \in [1, 2]$.

The searching space for every problem was

$$z_j = (x_j, y_j) \in S = ([0, 10], [0, 10]), \quad \alpha_j \in [0.5, 5]$$

for all $j = 1, \ldots, p$. The tolerances used in the algorithms were set to $\epsilon_1 = \epsilon_2 = 10^{-3}$. The maximum iteration counter $r_{\max}$ was set to 400 for all the Weiszfeld-like methods except for WLMa and WLMb when they are used as local optimizer within UEGOa and UEGOb, respectively. In that case, $r_{\max}$ was set to the budget per species opt_eval. In the algorithm SA $r_{\max}$ was set to $5000 \cdot p$ (where p is the number of facilities to be located).

5.2 Comparing the reliability and performance of the algorithms when $p = 2$

To check the reliability of the different algorithms, to be understood as the percentage of success at finding the global optimum, we need test problems for which we know their optimal values and their optimal solutions. In Tóth et al. (2006) several problems where $p = 2$ new facilities had to be located were solved exactly with an interval branch-and-bound method. Due to the complexity of the problem, no computational results are reported in Tóth et al. (2006) for problems with $p > 2$. In fact, for the problems with $p = 2$ the running times of the interval B&B method vary from 6 to more than 140 hours. The interval B&B method produces a very narrow interval containing the global optimal value of the problem and a list of 6-dimensional intervals vectors where any maximizer point must lie.

We have solved each of the fifteen problems considered in this subsection with the following heuristics: WLMa, WLMb, WLMc, the multistart algorithms mWLMa, mWLMb, mWLMc and mSA, and the evolutionary algorithms UEGOa and UEGOb. The number of times that the multistart algorithms were allowed to repeat their basic algorithms was chosen so that the CPU times used by the multistart algorithms were, on average (when considering the fifteen problems), similar to the CPU times used by the UEGO algorithms. Since each run of a heuristic may provide a different solution, each heuristic was run five times for each problem. For each algorithm execution, we obtain the approximate optimal value Obj_{ap} provided by the heuristic, the point P_{ap} at which that value is attained and the CPU time employed by the heuristic. We also check whether the heuristic has successfully found the optimal solution. We say so when both Obj_{ap} and P_{ap} are included in the corresponding intervals provided by the interval B&B method. With that information, for a given heuristic and a given problem, we compute the averages, in the five runs, of the objective value 'Av(Obj)' and running time 'Av(T)' (in seconds), and the number of times that the algorithm has attained, in the five runs, the optimum solution, in percentage, 'Av(S)'.

In Table 1 we give the values obtained for the fifteen problems when mWLMa and UEGOa are used. In the first column we can see the setting of each problem. The values in the last line of the table give the corresponding average of the averages.

As it can be seen from the table, whereas the CPU time used by mWLMa increases with the pair (n, m), the time of UEGOa is more erratic. This is because UEGO adapts itself to the difficulty of the problem at hand, whereas the multistart algorithm does not take the difficulty of the problem into account, but just applies WLMa a given number of times (650 in this study). In fact, as we can see, UEGOa reaches the global optimum with 100% of success for all the problems, whereas only four problems are solved by mWLMa with 100% of success. Considering the fifteen problem together, UEGOa has a 100% of success, and mWLMa only 50% (i.e., only in half of the 75 runs mWLMa obtains the optimal solution). Notice also that, except for the third problem, in which both algorithms attain a 100% success, UEGOa always obtains a better Av(Obj) value.

To have a general overview of the results, it is better to see only the summarizing values of the last line in Table 1. In Table 2 we can see those values for all the algorithms we have tested. After the name of the multistart algorithms we give, in brackets, the number of times that the corresponding basic algorithm has been repeated.

As we can see, WLMa and WLMb have a similar Av(T) value, but WLMa obtains better solutions than WLMb. On the other hand, the new Wesizfeld-like algorithm WLMc obtains worse results. This was surprising, since WLMc takes all the new facilities into account simultaneously, instead of one at a time, as the other two algorithms do, and this non-myopic view could have been an advantage for it. In any case, none

Table 1: Detailed results with mWLMa and UEGOa when $p = 2$.

Problem	mWLMa(650)			UEGOa		
(n,m,k)	Av(Obj)	Av(T)	Av(S)	Av(Obj)	Av(T)	Av(S)
(21,5,2)	307.7767	42.0	20	308.6051	21.8	100
(21,5,3)	440.2696	41.6	100	440.2762	21.2	100
(50,2,0)	206.8271	87.2	100	206.8269	76.4	100
(50,2,1)	219.1414	82.2	100	219.1480	93.8	100
(50,5,0)	149.3007	104.6	40	149.4327	50.0	100
(50,5,2)	253.5393	105.2	80	253.6061	52.4	100
(50,10,0)	88.9117	152.6	60	88.9365	67.4	100
(50,10,4)	178.9957	148.8	60	179.0204	80.0	100
(71,5,2)	299.6643	138.6	100	299.7357	61.0	100
(100,2,0)	442.0476	152.4	0	443.8871	80.2	100
(100,2,1)	540.1376	146.6	40	540.1934	131.2	100
(100,5,0)	258.6377	206.0	0	259.5785	99.8	100
(100,5,2)	614.3975	203.2	40	614.5680	657.6	100
(100,10,0)	258.4436	305.6	20	261.8812	129.6	100
(100,10,4)	399.8011	296.0	0	400.1539	479.0	100
Average	310.5226	147.5	50	311.0567	140.0	100

Table 2: Summarizing results when $p = 2$.

Alg	Av(Obj)	Av(T)	Av(S)
WLMa	304.7244	0.97	1
WLMb	294.1313	0.98	0
WLMc	281.2690	1.49	0
mWLMa(650)	310.5226	147.50	50
mWLMb(650)	304.3774	158.41	16
mWLMc(650)	306.0342	429.53	3
mSA(16)	282.7083	152.88	0
UEGOa	311.0567	140.09	100
UEGOb	311.0444	123.72	100

of the three Weiszfeld-like methods is able to succeed at finding the optimum solution greater than 1%. This confirms the need to use a multistart strategy with those algorithms to have a better chance of achieving the optimum solution. In particular, we have repeated WLMa and WLMb 650 times (algorithms mWLMa and mWLMb), so that their CPU times are similar to those of the UEGO algorithms. As expected, mWLMa is superior to mWLMb (see their Av(Obj) values), and has a 50% success rate at finding the optimum solution, as compared to the more discrete 16% of mWLMb. And this, using 6.88% less time. This contradicts the results in (Drezner, 1998; Drezner et al., 2002), where for the simpler problem considered in those papers, the procedure corresponding to mWLMb was superior to the one corresponding to mWLMa. As for mWLMc, even when it is allowed to run four times longer than mWLMa and mWLMb, it obtains worse results than mWLMa and its success is just 3%.

Concerning mSA, the multistart simulated annealing strategy that generalizes the procedure that in Drezner et al. (2002) proved to be the best heuristic at solving the simpler problem considered in that paper, here it obtains very bad results. The reason

for this is not due to the additional quality variables of the problem, but to the presence of constraints. The forbidden regions around the existing demand points make some of the moves in the rectangular grid unfeasible, provoking the inefficiency of the simulated annealing strategy.

As for UEGO algorithms, they both are competitive and obtain 100% success at finding the optimum solution, and using less time than the multistart algorithms. Thus, the evolutionary algorithms are clearly the most reliable and efficient algorithms.

5.3 Comparing the performance of the algorithms for $p > 2$

We do not have the exact solution of any problem in which more than two facilities are to be located. Thus, in this case, we cannot study the reliability of the algorithms at finding the global optimum. However, we can still compare them so as to know which one obtains the best solutions. To this aim, we have solved the fifteen problems of the previous subsection but considering now that we want to locate $p = 3, \ldots, 10$ new facilities. The results have been summarized in Figures 4 and 5 (for completeness, we have also included the results when $p = 2$ facilities have to be located). We have solved the problems only with the four algorithms giving the best results for the case $p = 2$, namely, mWLMa, mWLMb, UEGOa and UEGOb. We varied the number of times that WLMa and WLMb were repeated in mWLMa and mWLMb for each p, so that the CPU times used by the multistart algorithms were, on average (considering the fifteen problems), similar to the CPU times used by the UEGO algorithms. The actual number of times that the Weiszfeld-like algorithms were repeated are given in Figure 4, after the values of p, within brackets. Since each run of a heuristic may provide a different solution, each heuristic has been run five times for each problem. In each run, we obtain the approximate optimal value Obj_{ap} provided by the heuristic, the point P_{ap} at which that value is attained and the CPU time used by the heuristic. With that information, for a given heuristic and a given problem, we compute the averages, in the five runs, of the objective value 'Av(Obj)' and running time 'Av(T)'. The values displayed in Figure 5 are the averages, in the fifteen problems, of those average times, whereas in Figure 4 we give the relative differences in objective function value of UEGOb, mWLMa and mWLMb with regard to UEGOa, which was the algorithm which always gave the best average results. These relative differences between the values of the objective functions have been computed by the expression

$$\text{DifRel(Obj)} = \frac{\text{Av(Obj}_{\text{UEGOa}}) - \text{Av(Obj)}}{\text{Av(Obj}_{\text{UEGOa}})} \cdot 100$$

As we can see, on average, mWLMa always obtains better objective function values than mWLMb, and using less time. However, both UEGOa and UEGOb outperform the multistart strategies for all the values of p, again confirming the superiority of the evolutionary strategies. In particular, UEGOa gives the best results, with better objective values than those offered by UEGOb for all the values of p, and in most of the cases using less time. Although on average the differences in the objective values provided by mWLMa and UEGOa are not too great, for all the values of p there are instances in which the relative difference was higher than 1.65%, and in some cases it reached a considerable 2.74%. It is also interesting to note that the CPU time of the UEGO algorithms increases linearly with p.

In the comparison of the algorithms we have talked about the objective values, but it is interesting to know whether the solutions obtained by the multistart algorithms are

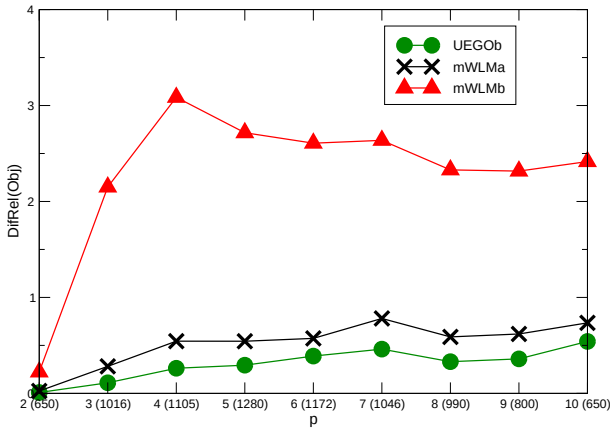

Figure 4: Rela ıction of p.

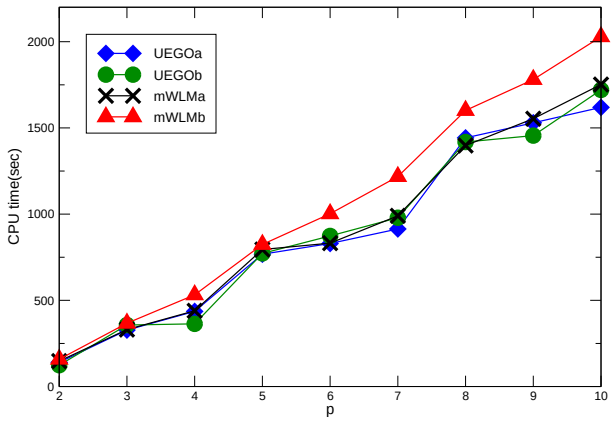

Figure 5: CPU time of the algorithms as a function of p.

close to those obtained by UEGOa. To this aim, we need first a function which measures how close two solutions are.

Suppose that we solve a given multiple competitive facilities location and design problem in which we want to locate p new facilities, with two different algorithms, $Alg^{(1)}$ and $Alg^{2)}$. Let us denote by $nf^{(t)} = (nf_1^{(t)}, \ldots, nf_p^{(t)})$ the solution obtained by $Alg^{(t)}$, with $nf_j^{(t)} \in \mathbb{R}^3, t = 1, 2, j = 1, \ldots, p$. Then, we propose to measure the distance between $nf^{(1)}$ and $nf^{(2)}$ as:

$$d_M(nf^{(1)}, nf^{(2)}) = \frac{1}{2}\left(\sum_{i=1}^{p} \min_{j=1,\ldots,p} \|nf_i^{(1)} - nf_j^{(2)}\|_2 + \sum_{j=1}^{p} \min_{i=1,\ldots,p} \|nf_i^{(1)} - nf_j^{(2)}\|_2\right)$$

where $\|\cdot\|_2$ stands for the Euclidean norm. This measure can be seen as a modification

of the classical Hausdorff distance,

$$d_H(nf^{(1)}, nf^{(2)}) = \max\left\{\max_{i=1,\ldots,p}\min_{j=1,\ldots,p}\|nf_i^{(1)} - nf_j^{(2)}\|_2,\ \max_{j=1,\ldots,p}\min_{i=1,\ldots,p}\|nf_i^{(1)} - nf_j^{(2)}\|_2\right\}$$

to take the distance between each point and its closest one into account (and not only the maximum of those distances over the set of points, as the Hausdorff distance does). For instance, in the left picture of Figure 6, if the triangles represent the locations for $p = 3$ facilities offered by $Alg^{(1)}$ and the squares the locations offered by $Alg^{(2)}$, then $d_M(nf^{(1)}, nf^{(2)}) = d_H(nf^{(1)}, nf^{(2)}) = 1$, whereas in the right picture $d_M(nf^{(1)}, nf^{(2)}) = 0.5$ and $d_H(nf^{(1)}, nf^{(2)}) = 1$ (in both examples we assume that all the facilities have the same quality).

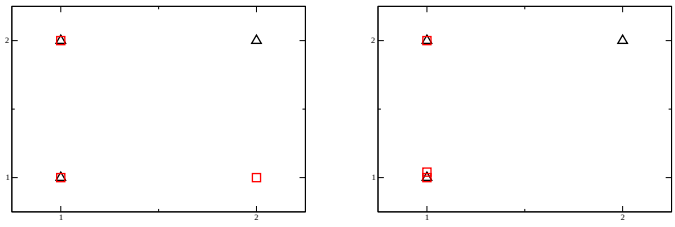

Figure 6: The triangles represent $nf^{(1)}$ (the locations of $p = 3$ new facilities) and the squares $nf^{(2)}$ (other locations for the facilities). In the left picture $d_M(nf^{(1)}, nf^{(2)}) = 1$, whereas in the right one $d_M(nf^{(1)}, nf^{(2)}) = 0.5$

Table 3: Differences in solution between mWLMa and UEGOa.

p	Av(d_M)	Max(d_M)
1	0.00327	0.47982
2	0.27784	1.97440
3	1.05789	7.50578
4	2.70268	10.60503
5	3.56291	10.32289
6	4.43084	10.29280
7	5.99879	13.56942
8	5.98995	12.35032
9	6.62278	11.38778
10	6.70409	18.18882

In Table 3 we can see the difference between the solutions obtained by mWLMa and UEGOa in the fifteen problems considered. For each value of p, we give the average difference in the fifteen problems, Av(d_M), as well as the maximum of those differences, Max(d_M). As we can see, even for small values of p the difference in the solutions is great. This clearly shows the high degree of nonlinearity of the problem, and the fact that there may be local optima whose objective values are close to the optimal value. Furthermore, for each value of p there is always at least one problem for which the difference in the solutions provided by both algorithms is much higher than Av(d_M) (see Max(d_M)). The corresponding differences between mWLMb and UEGOa are even higher.

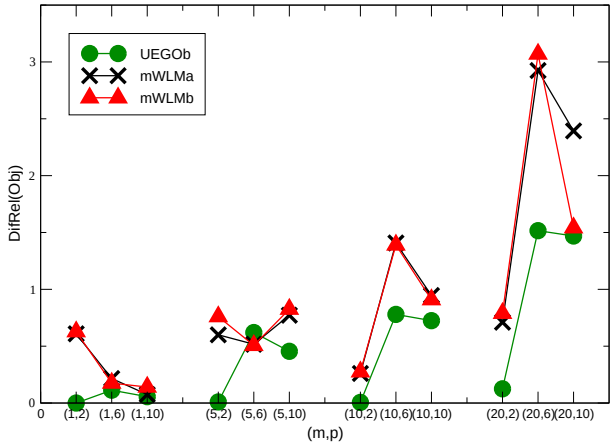

Figure 7: Sensitivity to the number of existing facilities: DifRel(Obj).

5.3.1 Sensitivity to the number of existing facilities

In a new study we have researched whether the performance of the algorithms depends on the number of existing facilities m. To this aim, for m =1, 5, 10 and 20, we have randomly generated five problems. Those twenty problems have been solved assuming that we want to locate $p = 2$, 6 or 10 facilities (in all the problems we have set $n = 100$ and $k = 0$). Each problem has been solved once, with UEGOa, UEGOb, mWLMa and mWLMb. The results are summarized in Figures 7 and 8. In Figure 7 we have plotted for each pair (m, p) the average of the relative differences in the objective function value of UEGOb, mWLMa and mWLMb with regard to UEGOa (which was the algorithm that always gave the best results) in the five problems of each setting. In Figure 8 we plot, for each pair (m, p), the mean time employed by each algorithm at solving the five problems of each setting.

As we can see in Figure 7, the performance of the algorithms is the same regardless of the value of m: UEGOa is the algorithm which provides the best results, followed by UEGOb. The multistart algorithms mWLMa and mWLMb provide solutions of similar quality (with mWLMa slightly better than mWLMb), and they are worse than the ones provided by the evolutionary algorithms. Notice also that the higher the value of m, the greater the relative difference in objective function tends to be.

As for the computational times (see Figure 8), UEGOa is again the fastest one, followed by UEGOb, mWLMa and mWLMb, in that order. Interestingly, the time does not depend too much on the parameter m (it depends mainly on the number p of new facilities to be located).

5.3.2 Sensitivity to the quality of the existing facilities

In our last experiment we have studied to what extent the performance of the algorithms depends on the quality of the existing facilities. In particular, we have randomly generated five problems by setting first $\alpha_{il} = 1$ for all i, l, other five problems with $\alpha_{il} = 3$ for all i, l, and five problems more with $\alpha_{il} = 5$ for all i, l. In all the prob-

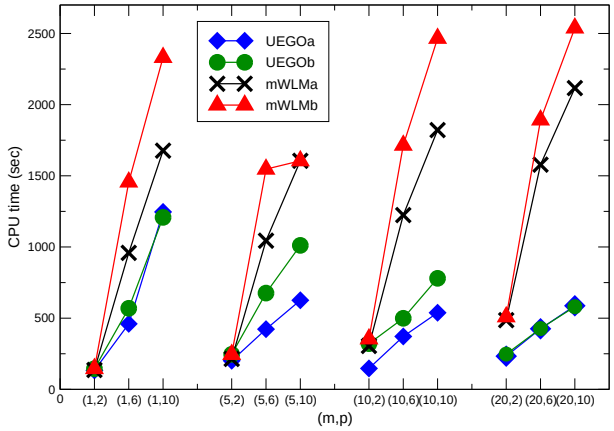

Figure 8: Sensitivity to the number of existing facilities: CPU times.

lems we have set $\gamma_i = 1$ for all i, $n = 100$, $m = 5$ and $k = 0$, and we have solved those fifteen problems for $p = 2$, 6 and 10. As in the previous study, each problem has been solved once, with UEGOa, UEGOb, mWLMa and mWLMb. The results are summarized in Figures 9 and 10. As in the previous subsection, in Figure 9 we have plotted for each pair (α_{il}, p) the average of the relative differences in objective of UEGOb, mWLMa and mWLMb with regard to UEGOa (which was the algorithm that gave always the best results) in the five problems of each setting. In Figure 10 we plot for each pair (α_{il}, p), the mean time used by each algorithm at solving the five problems of each setting.

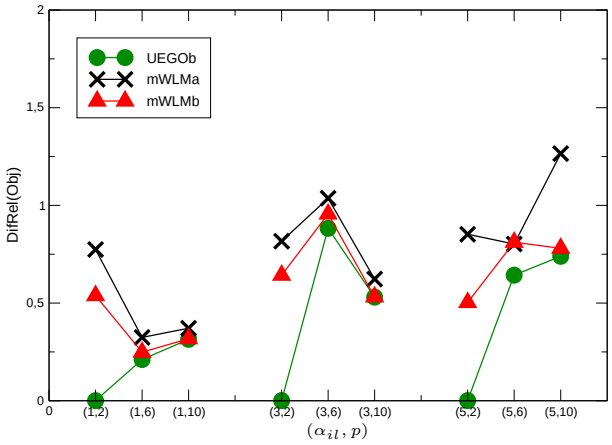

Figure 9: Sensitivity to the quality of the existing facilities: DifRel(Obj).

As we can see in Figure 9, the performance of the algorithms does not depend

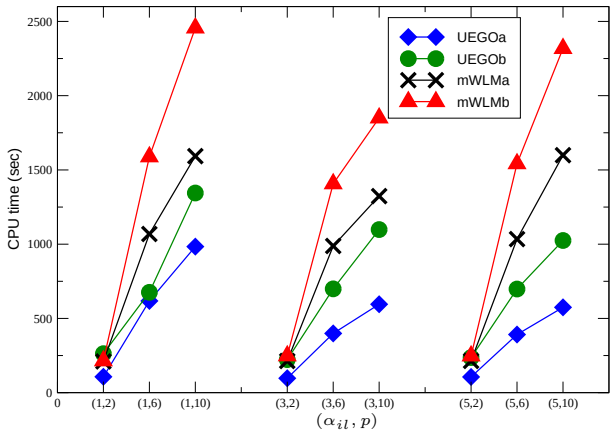

Figure 10: Sensitivity to the quality of the existing facilities: CPU times.

on the quality of the existing facilities. Again, UEGOa is the algorithm which provides the best results, followed by UEGOb, mWLMa and mWLMb, in that order. The same ordering can be seen when considering the computational times, with UEGOa the fastest one. The time does not depends on the quality on the existing facilities.

6 Conclusions and future work

In this paper we have considered a multiple competitive facilities location and design problem on the plane. The model is an extension of the single facility problem presented in Fernández et al. (2006) (see also Tóth et al. (2006)). To cope with it, several algorithms have been proposed. They all use Weiszfeld-like algorithms as local search procedures (steepest descent type methods). We have worked with three such algorithms. In two of them, WLMa and WLMb, the new variables for one facility at a time are found while holding the other facilities fixed in their current values. The third one, WLMc, is a new method which directs the search without reducing it to the single facility case.

The Weiszfeld-like procedures have been used within multi-start algorithms, within a simulated annealing algorithm (mSA), and within an evolutionary algorithm (UEGO). mSA is an extension of the simulated annealing algorithm presented in Drezner et al. (2002) to solve a related (simpler) problem, and although in Drezner et al. (2002) it proved to be the best strategy of the five analyzed in that article for the simpler problem, the performance of mSA in our model is very bad (mainly due to the presence of constraints).

As for the multi-start strategies, mWLMa is the best one, closely followed by mWLMb (interestingly, in the simpler model introduced in Drezner (1998) it was the procedure analogous to mWLMb which gave the best results). On the other hand, mWLMc does not seem to be suitable for the problem at hand.

However, neither the simulated annealing algorithm nor the multi-start strategies are good methods at finding the global optimum. In a set of fifteen problems in which

two facilities have to be located and for which the optimal solutions are known, only the evolutionary algorithm UEGO is able to obtain the optimal solution with a 100% of success. Furthermore, in a comprehensive computational study on a set of problems in which up to ten facilities have to be located, the evolutionary algorithm UEGO obtained better results than the other algorithms. In particular, the implementation UEGOa of UEGO in which WLMa is used as local optimizer is the algorithm providing the best results. The differences between the solutions provided by UEGOa and the multi-start algorithms are big, which proves the high nonlinearity of the problem. On the other hand, the CPU time of UEGO increases linearly with the number of facilities to be located.

To our knowledge, with the exception of (Bhandarkar and Zhang, 1999; Salhi and Gamal, 2003; Houck et al., 1996) (in which the uncapacitated location-allocation problem is considered), this is the first time that evolutionary algorithms are applied to a continuous location problem. And it is important to highlight that UEGO is a general method, which can be specialized to solve many other continuous location problems, not only the one considered here.

Having a reliable method such as UEGO is also important to carry out an economic study of the model. For instance, analyzing the objective values of the problems solved in Subsection 5.3 as a function of p, we can see that the higher the p, the greater the profit, but with a decreasing rate, which means that there is a number of facilities above which it is no longer profitable to set up more facilities. In the future we plan to analyze more economic implications of the model. Another line for future research is to study the so-called Stackelberg problem, in which a chain (the leader) wants to locate r new facilities, and after that, a second chain (the follower) reacts by locating p new facilities. The problem to be solved by the leader is where to locate its facilities so as to maximize its profit after the establishment of the follower's new facilities.

Acknowledgment

The authors are indebted to Boglárka Tóth, who has provided us with the WLM1 code and the solutions obtained by the interval branch-and-bound method for the problems with $p = 2$ (see Tóth et al. (2006)); her help is gratefully acknowledged.

References

Aytug, H. and Saydam, C. (2002). Solving large-scale maximum expected covering location problems by genetic algorithms: A comparative study. *European Journal of Operational Ressearch*, 141:480–494.

Beasley, D., Bull, D., and Martin, R. (1993). A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2):101–125.

Benati, S. and Laporte, G. (1964). Tabu search algorithms for the $(r|x_p)$-medianoid and $(r|p)$-centroid problems. *Location Science*, 2(4):193–204.

Bhandarkar, S. and Zhang, H. (1999). Image segmentation using evolutionary computation. *IEEE Trans. Evolutionary Computation*, 3(1):1–21.

Brimberg, J., Hansen, P., Mlandinović, N., and Taillard, E. (2000). Improvement and comparison of heuristics for solving the uncapacitated multisource Weber problem. *IEEE Trans. Evolutionary Computation*, 48(3):444–460.

Deb, K. (1989). Genetic algorithms in multimodal function optimization. TCGA report no. 89002, The University of Alabama, Dept. of Engineering mechanics.

Dorigo, M. and Di Caro, G. (1999). The ant colony optimization meta-heuristic. In Corne, D., Dorigo, M., and Glover, F., editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, London.

Drezner, T. (1995a). Competitive location in the plane. *Facility Location: A survey of applications and methods, Springer*, pages 285–300.

Drezner, T. (1998). Location of multiple retail facilities with limited budget constraints in continuous space. *Journal of Retailing and Consumer Services*, 142:173–184.

Drezner, T., Drezner, Z., and Salhi, S. (2002). Solving the multiple competitive facilities location problem. *European Journal of Operational Research*, 142:138–151.

Drezner, Z. (1984). The p-center problem: Heuristic and optimal algorithms. *Journal of the Operational Research Society*, 35:741–748.

Drezner, Z. (1995b). *Facility Location: A Survey of Applications and Methods*. Springer, Berlin.

Drezner, Z. and Hamacher, H. (2002). *Facility location. Applications and theory*. Springer, Berlin.

Drezner, Z. and Suzuki, A. (2004). The big triangle small triangle method for the solution of non-convex facility location problems. *Operations Research*, 52:128–135.

Eiselt, H., Laporte, G., and Thisse, J. (1993). Competitive location models: A framework and bibliography. *Transportation Science*, 27:44–54.

Erkut, E. and Neuman, S. (1989). Analytical models for locating undesirable facilities. *European Journal of Operational Research*, 40:275–291.

Fernández, J., Fernández, P., and Pelegrín, B. (2000). A continuous location model for siting a non-noxious undesirable facility within a geographical region. *European Journal of Operational Research*, 121:259–274.

Fernández, J., Fernández, P., and Pelegrín, B. (2002). Estimating actual distances by norm functions: A comparison between the $l_{k,p,\theta}$-norm and the $l_{b_1,b_2,\theta}$-norm and a study about the selection of the data set. *Computers and Operations Research*, 29:609–623.

Fernández, J. and Pelegrín, B. (2001). Using interval analysis for solving planar single-facility location problems: New discarding tests. *Journal of Global Optimization*, 19:61–81.

Fernández, J., Pelegrín, B., Plastria, F., and Tóth, B. (2006). Solving a Huff-like competitive location and design model for profit maximization in the plane. *European Journal of Operational Research*, DOI:10.1016/j.ejor.2006.02.005, to appear.

Francis, R., McGinnis, L., and White, J. (1992). *Facility layout location: An analytical approach*. Prentice Hall, Englewood Cliffs.

Glover, F. and Laguna, M. (1997). *Tabu search*. Kluwer Academic Publisher.

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley.

González-Linares, J., Guil, N., Zapata, E., Ortigosa, P., and García, I. (2000). Deformable shapes detection by stochastic optimization. In *Proceedings of the International Conference on Image Processing*, pages 780–783, Vancouver (Canada).

Hakimi, S. (1990). Locations with spatial interactions: Competitive locations and games. In Francis, R. and Mirchandani, P., editors, *Discrete location theory*. Wiley/Interscience.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467.

Hansen, P., Peeters, D., Richard, D., and Thisse, J. (1985). The minisum and minimax location problems revisited. *Operations Research*, 33:1251–1265.

He, J., Xu, J., and Yao, X. (2000). Solving equations by hybrid evolutionary computation techniques. *IEEE Transactions on Evolutionary Computation*, 4(3):295–304.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Houck, C., Joines, J., and Kay, M. (1996). Comparison of genetic algorithms, random restart and two-opt switching for solving large location-allocation problems. *Computers and Operations Research*, 23:587–596.

Huff, D. (1964). Defining and estimating a trading area. *Journal of Marketing*, 28(3):34–38.

Jamarillo, J., Bhadury, J., and Batta, R. (2002). On the use of genetic algorithms to solve location problems. *Computers and Operations Research*, 29:761–779.

Kilkenny, M. and Thisse, J. (1999). Economics of location: A selective survey. *Computers and Operations Research*, 26:1369–1394.

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Ma, P., Chan, K., Yao, X., and Chiu, D. (2006). An evolutionary clustering algorithm for gene expression microarray data analysis. *IEEE Transactions on Evolutionary Computation*, 10(3):296–314.

Ortigosa, P., García, I., and Jelasity, M. (2001). Reliability and performance of UEGO, a clustering-based global optimizer. *Journal of Global Optimization*, 9:265–289.

Plastria, F. (1992). GBSSS: The generalized big square small square method for planar single-facility location. *European Journal of Operational Research*, 62:163–174.

Plastria, F. (2001). Static competitive facility location: An overview of optimisation approaches. *European Journal of Operational Research*, 129:461–470.

Redondo, J., Fernández, J., García, I., and Ortigosa, P. (to appear). A robust and efficient algorithm for planar competitive location problems. *Annals of Operations Research*.

Redondo, J., Ortigosa, P., García, I., and Fernández, J. (2004). Image registration in electron microscopy. A stochastic optimization approach. *Lecture Notes in Computer Science, Proceedings of the International Conference on Image Analysis and Recognition, ICIAR 2004*, 3212(2):141–149.

Salhi, S. and Gamal, M. (2003). A genetic algorithm based approach for the uncapacitated continuous location-allocation problem. *Annals of Operations Research*, 123:203–222.

Solis, F. and Wets, R. (1981). Minimization by random search techniques. *Mathematics of Operations Research*, 6:19–30.

Speer, N., Spieth, C., and Zell, A. (2004). A memetic co-clustering algorithm for gene expression profiles and biological annotation. In *Proceedings of the IEEE 2004 Congress on Evolutionary Computation, CEC 2004*, volume 2, pages 1631–1638. IEEE Press.

Tóth, B., Fernández, J., Pelegrín, B., and Plastria, F. (2006). Sequential versus simultaneous approach in the location and design of two new facilities using planar Huff-like models. Under second revision in *Computers and Operations Research.* Available at `http://www.um.es/geloca/gio/2fac.pdf`.

Weber, A. (1909). *Über den Standort der Industrien 1.* Teil: Reine theorie des standordes, Tübingen, Germany.

Weiszfeld, E. (1937). Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal*, 43:355–386.

Whitley, D., Gordon, V., and Mathias, K. (1994). Lamarckian evolution, the baldwin effect and function optimization. In Davidor, Y., Schwefel, H.-P., and Männer, R., editors, *Parallel Problem Solving from Nature – PPSN III*, pages 6–15, Berlin. Springer.

Yang, J. and Yang, C. (2005). The retail stores' competitive location problem with retail regional saturation. In *Services Systems and Services Management, 2005. Proceedings of ICSSSM '05. 2005 International Conference on*, volume 2, pages 1511–1516.