Comput Optim Appl (2010) 45: 111–141
DOI 10.1007/s10589-008-9170-0

Heuristics for the facility location and design (1|1)-centroid problem on the plane

J.L. Redondo · J. Fernández · I. García · P.M. Ortigosa

Received: 14 March 2007 / Revised: 24 January 2008 / Published online: 20 February 2008

Abstract A chain (the leader) wants to set up a single new facility in a planar market where similar facilities of a competitor (the follower), and possibly of its own chain, are already present. The follower will react by locating another single facility after the leader locates its own facility. Fixed demand points split their demand probabilistically over all facilities in the market in proportion to their attraction to each facility, determined by the different perceived qualities of the facilities and the distances to them, through a gravitational model. Both the location and the quality (design) of the new leader's facility are to be found. The aim is to maximize the profit obtained by the leader following the follower's entry. Four heuristics are proposed for this hard-to-solve global optimization problem, namely, a grid search procedure, an alternating method and two evolutionary algorithms. Computational experiments show that the evolutionary algorithm called UEGO_cent.SASS provides the best results.

Keywords Continuous location · Competition · Centroid problem · Stackelberg problem · Evolutionary algorithms

This paper has been sponsored by the Ministry of Education and Science of Spain under the research projects SEJ2005-06273/ECON and TIN2005-00447, in part financed by the European Regional Development Fund (ERDF).

J.L. Redondo · I. García · P.M. Ortigosa
Department of Computer Architecture and Electronics, University of Almería, Almería, Spain

J. Fernández (✉)
Facultad de Matemáticas, Universidad de Murcia, 30100 Espinardo, Murcia, Spain
e-mail: josefdez@um.es

J. Fernández
Department of Statistics and Operations Research, University of Murcia, Murcia, Spain

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.) See [9, 10, 14, 21] for an introduction to the topic. According to their feasible set, location problems are classified as discrete, on networks or continuous, and the mathematical formulations and solution methods of the problems vary substantially depending on the type of problem. In this paper we deal with continuous problems. Continuous location theory is very vast. Since the pioneering works of Weber [29] in 1909 and Weiszfeld [30] in 1937, the literature on that topic has been increasing constantly, specially since the 1970s, to the point that it has now its own entry (90B85) in the *Mathematics Subject Classification* used by *Mathematical Reviews* and *Zentralblatt für Mathematik*.

In particular, we consider here a *competitive facility location* (and *design*) problem. 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 competing facilities offering the same service are already present (or will enter the market in the near future). Many competitive location models are available in the literature, as can be seen, for instance, in survey papers [12, 19, 24]. They vary in the ingredients which form the model. For instance, we may want to locate just a single facility or more than one new facility. The demand (usually supposed to be concentrated in a discrete set of points, called *demand points*) can be either *inelastic* or *elastic*, depending on whether the goods are essential or inessential. The *patronizing behavior* of the customers can be either *deterministic*, when the full demand of the customer is served by the facility to which he/she is *attracted* most (leading to Hotelling-type models) or *probabilistic*, when the customer splits his/her demand among all the existing facilities (leading to Huff-type models). The *attraction* (*or utility*) *function* of a customer towards a given facility, which usually depends on the distance between the customer and the facility, as well as on other characteristics of the facility which determine its *quality*, is also a key factor to be specified. The *market share* captured by the facilities depends on all those factors. Furthermore, the competition may be *static*, which means that the competitors are already in the market, the owner of the new facility knows their characteristics and no reaction is expected from them, or *with foresight*, in which the competitors are assumed to react after the new facility enters. Furthermore, if the competitors can change their decisions, then we have a *dynamic model*, in which the existence of *equilibrium* situations is of major concern.

The scenario considered in this paper is that of a *duopoly*. A chain (the leader) wants to set up a single new facility in a planar market where similar facilities of a competitor (the follower), and possibly of its own chain, are already present. Fixed demand points split their demand (supposed to be inelastic) probabilistically over all facilities in the market in proportion to their attraction to each facility, determined by the different perceived qualities of the facilities and the distances to them, through a gravitational or Huff-type model. Both the location and the quality (design) of the new facility are to be found. Several types of constraints and costs are also considered. After the location of the leader's facility, the competitor will react by locating another new facility at the place that maximizes its own profit. The objective of the leader is to find the location and the quality of its facility that maximizes its profit (to be understood as the income resulting from the market share captured by the chain minus its operational costs), following the location of the facility of the follower.

These types of problems are known as Stackelberg problems in economic literature and as Simpson's problems in voting theory. In Location literature they were introduced by Hakimi [16]. He introduced the terms *medianoid* for the follower problem, and *centroid* for the leader problem. More precisely, an $(r|X_p)$ medianoid problem refers to the follower's problem of locating r new facilities in the presence of p leader's facilities located at a set of points X_p. And an $(r|p)$ centroid problem refers to the leader's problem of locating p new facilities, knowing that the follower will react positioning r new facilities by solving an $(r|X_p)$ medianoid problem.

The literature on centroid problems is scarce (see [11] for a review on the topic until 1996), and to our knowledge, among the existing papers only four of them deal with continuous problems. This is mostly due to the complexity of that type of bilevel programming problems. Drezner [8] solved the (1|1) centroid problem for the Hotelling model and Euclidean distances exactly, through a geometric-based approach. Bhadury *et al.* [2] also considered the $(r|p)$ centroid problem for the Hotelling model with Euclidean distances, and gave an alternating heuristic to cope with it. Drezner and Drezner [5] considered the Huff model, and proposed three heuristic approaches for handling the (1|1) centroid problem (see also [6]).

All the papers dealing with the centroid problem make use of procedures for solving the corresponding medianoid problem. However, the medianoid problem is also a hard-to-solve global optimization problem (as most competitive location problems are). Things become even more complicated when we consider the quality of the facility as an additional variable of the problem, such as we do in this paper. Little research has been done on this kind of problem with simultaneous decisions on location and quality in continuous space. Under a deterministic (or Hotelling) customer behavior, the $(1|X_1)$ medianoid problem has been studied in [25], and for a probabilistic, or Huff, customer behavior in [13]. In [4], a closely related multifacility problem with probabilistic customer behavior is analyzed.

Deterministic methods to solve some of the medianoid problems exactly can be found in the literature, see for instance [7] for a branch-and-bound method to solve the $(1|X_1)$ medianoid problem with Huff patronizing behavior when the quality of the new facility is supposed to be given in advance, and [13] for an interval branch-and-bound method when the quality is also a variable of the problem. However, the interval B&B method in [13] is not suited for solving the centroid problem, since it is time-consuming, and can make the procedure inoperative (usually, to solve a single centroid problem many medianoid subproblems have to be solved). Hence, heuristic (but reliable) methods are needed to be used within the algorithms for solving our centroid problem. A multistart Weiszfeld-like method was presented in [3] to solve the $(1|X_1)$ medianoid problem with Huff patronizing behavior when the quality of the new facility is supposed to be fixed, and it was extended in [13] to handle the case where the quality is also variable and there are constraints in the problem. Recently, in [26], the evolutionary algorithm UEGO (*Universal Evolutionary Global Optimizer*), described in [23], was proposed for the $(1|X_1)$ medianoid problem, and it proved to be more reliable than the multistart heuristic introduced in [13]. Other metaheuristics, such as tabu search, have also been proposed to cope with medianoid problems when the feasible space is a network (see [1]).

In this paper we study several procedures for solving a constrained $(1|1)$ centroid problem with Huff patronizing behavior, in which the design of the facility is considered a third variable. In Sect. 2 the facility location and design problem is formally introduced. The heuristics used to solve the corresponding medianoid problem are presented in Sect. 3. In the next section we explain several heuristics specifically devised to cope with the centroid problem, namely, a grid search procedure, an alternating procedure and two adaptations of the evolutionary algorithm UEGO (see [23]). In Sect. 5 we present some computational studies to compare the performance of these algorithms. As we will show, an adaptation of UEGO, which uses the stochastic hill climber SASS (*Single Agent Stochastic Search*) [28] as local optimizer, is the algorithm which provides the best results. It is important to highlight that UEGO can be easily adapted to handle many and diverse (location) problems. The paper ends in Sect. 6 with some conclusions and guidelines for future research.

2 A Huff-like (1|1)-centroid problem with decisions in both location and design

A chain, the *leader*, wants to locate a single new facility in a given area of the plane, where m facilities offering the same goods or product already exist. The first k of those m facilities belong to the chain, and the other $m-k$ to a competitor chain, the *follower*. The leader knows that the follower, as a reaction, will subsequently position a new facility too. The demand, supposed to be inelastic, is concentrated at n demand points, whose locations p_i and purchasing power w_i are known. The location f_j and quality of the existing facilities is also known.

The following notation will be used throughout this paper:

Indices

i	index of demand points, $i = 1, \ldots, n$.
j	index of existing facilities, $j = 1, \ldots, m$ (the first k of those m facilities belong to the leader's chain, and the rest to the follower's).

Variables

$z_1 = (x_1, y_1)$	location of the new leader's facility.
α_1	quality of the new leader's facility.
$nf_1 = (z_1, \alpha_1)$	variables of the new leader's facility.
$z_2 = (x_2, y_2)$	location of the new follower's facility.
α_2	quality of the new follower's facility.
$nf_2 = (z_2, \alpha_2)$	variables of the new follower's facility.

Data

p_i	location of the i-th demand point.
w_i	demand (or purchasing power) at p_i.
f_j	location of the j-th existing facility.
ϵ_i	minimum distance from p_i at which the new facilities can be located.

d_{ij}	distance between p_i and f_j.
α_{ij}	quality of f_j as perceived by p_i.
$g_i(\cdot)$	a non-negative non-decreasing function.
$\alpha_{ij}/g_i(d_{ij})$	attraction that p_i feels for f_j.
γ_i	weight for the quality of the new facilities as perceived by demand point p_i.
S_1	location space where the leader will locate its new facility.
S_2	location space where the follower will locate its new facility.
$q_1^{\min}$	minimum allowed quality for the new leader's facility.
$q_1^{\max}$	maximum allowed quality for the new leader's facility.
$q_2^{\min}$	minimum allowed quality for the new follower's facility.
$q_2^{\max}$	maximum allowed quality for the new follower's facility.

Miscellaneous

d_{iz_l}	distance between p_i and $z_l, l = 1, 2$.
$\gamma_i\alpha_l/g_i(d_{iz_l})$	attraction that p_i feels for $nf_l, l = 1, 2$.
$M_1(nf_1, nf_2)$	market share obtained by the leader after the location of the new facilities.
$M_2(nf_1, nf_2)$	market share obtained by the follower after the location of the new facilities.
$\Pi_1(nf_1, nf_2)$	profit obtained by the leader after the location of the new facilities.
$\Pi_2(nf_1, nf_2)$	profit obtained by the follower after the location of the new facilities.

The use of a general non-negative and non-decreasing function, g_i, in the attraction functions generalizes the proposals found in literature, such as $g_i(d) = d^{\lambda_i}$ (see [3, 18]) or $g_i(d) = e^{\lambda_i d}$ (see [17]), with $\lambda_i > 0$, a given parameter which might differ between demand points. Note that when $g_i(d) = 0$, which may only happen for some g_i functions if $d = 0$, i.e., the facility coincides with the demand point, the attraction becomes $+\infty$, which in the current model would mean that all demand will be attracted by that facility only. Therefore we may assume that $g_i(d_{ij}) > 0\ \forall i, j$, because any demand point i for which some $g_i(d_{ij}) = 0$ would be totally lost to the new facilities, so it may simply be left out of the model.

In the spirit of Huff [18], we consider that the patronizing behavior of customers is probabilistic, that is, demand points split their purchasing power among the facilities proportionally to the attraction they feel for them. Using these assumptions, the market share attracted by the leader's chain after the location of the leader and the follower's new facilities is

$$M_1(nf_1, nf_2) = \sum_{i=1}^{n} \omega_i \frac{\frac{\gamma_i\alpha_1}{g_i(d_{iz_1})} + \sum_{j=1}^{k} \frac{\gamma_i\alpha_j}{g_i(d_{ij})}}{\frac{\gamma_i\alpha_1}{g_i(d_{iz_1})} + \frac{\gamma_i\alpha_2}{g_i(d_{iz_2})} + \sum_{j=1}^{m} \frac{\alpha_{ij}}{g_i(d_{ij})}}$$

and the corresponding market share attracted by the follower's chain is

$$M_2(nf_1, nf_2) = \sum_{i=1}^{n} \omega_i \frac{\frac{\gamma_i\alpha_2}{g_i(d_{iz_2})} + \sum_{j=k+1}^{m} \frac{\gamma_i\alpha_j}{g_i(d_{ij})}}{\frac{\gamma_i\alpha_1}{g_i(d_{iz_1})} + \frac{\gamma_i\alpha_2}{g_i(d_{iz_2})} + \sum_{j=1}^{m} \frac{\alpha_{ij}}{g_i(d_{ij})}}.$$

Given nf_1, the problem of the follower is the $(1|nf_1)$ medianoid problem:

$$(FP(nf_1)) \quad \begin{cases} \max & \Pi_2(nf_1, nf_2) = F_2(M_2(nf_1, nf_2)) - G_2(nf_2) \\ \text{s.t.} & z_2 \in S_2, \\ & d_{iz_2} \geq \epsilon_i, \quad i = 1, \ldots, n, \\ & \alpha_2 \in [q_2^{\min}, q_2^{\max}], \end{cases}$$

whose objective is the maximization of the profit obtained by the follower (once the leader has set up its new facility at nf_1), to be understood as the difference between the revenues obtained from the captured market share minus the operating costs of the new facility (see [13]). Note that we may ignore the operating costs of the existing facilities of the follower, since these are considered to be constant. $F_2(\cdot)$ is a strictly increasing function which transforms the market share into expected sales and $G_2(nf_2)$ is a function which gives the operating cost for the follower of a facility located at z_2 with quality α_2. Since it is rather likely that the closer to the demand points the follower is, the higher the operating costs of the facility will be due to the property value (this will make the cost of buying or renting the location higher), then the function $G_2(nf_2)$ should increase as z_2 approaches any demand point. On the other hand, the more quality we require of the facility, the higher the costs will be, at an increasing rate, so G_2 should be a nondecreasing and convex function in the variable α_2.

Let us denote by $nf_2^*(nf_1)$ an optimal solution of $(FP(nf_1))$. The problem for the leader is the $(1|1)$ centroid problem:

$$(LP) \quad \begin{cases} \max & \Pi_1(nf_1, nf_2^*(nf_1)) = F_1(M_1(nf_1, nf_2^*(nf_1))) - G_1(nf_1) \\ \text{s.t.} & z_1 \in S_1, \\ & d_{iz_1} \geq \epsilon_i, \quad i = 1, \ldots, n, \\ & \alpha_1 \in [q_1^{\min}, q_1^{\max}], \end{cases}$$

where F_1 and G_1 are the corresponding expected sales and operating costs functions, respectively, for the leader's chain.

In our computational studies we made the following choices:

- Functions F_l, $l = 1, 2$, are linear, $F_l(M) = c_l \cdot M$, where c_l is the income per unit of goods sold.
- Usually, the operating costs of a new facility consist of the sum of the locational costs and the costs related to reaching a given level of quality. Therefore functions $G_l, l = 1, 2$, are assumed to be separable, i.e. $G_l(nf_l) = G_l^a(z_l) + G_l^b(\alpha_l)$. In particular, we have considered $G_l^a(z_l) = \sum_{i=1}^n \Phi_l^i(d_{iz_l})$, with $\Phi_l^i(d_{iz_l}) = w_i/((d_{iz_l})^{\phi_l^{i0}} + \phi_l^{i1})$, $\phi_l^{i0}, \phi_l^{i1} > 0$ and $G_l^b(\alpha_l) = \exp(\alpha/\xi_l^0 + \xi_l^1) - \exp(\xi_l^1)$, with $\xi_l^0 > 0$ and $\xi_l^1 \in \mathbb{R}$ as given values.

A more detailed explanation of these functions, as well as other possible expressions for $G_l(nf_l)$ can be found in [13]. Of course, other functions might be more suitable depending on the real problem considered, and for each real application the most appropriate F_l and G_l functions should be discovered.

As we can see, the leader problem (LP) is much more difficult to solve than the follower problem $(FP(nf_1))$. Notice, for instance, that to evaluate its objective function

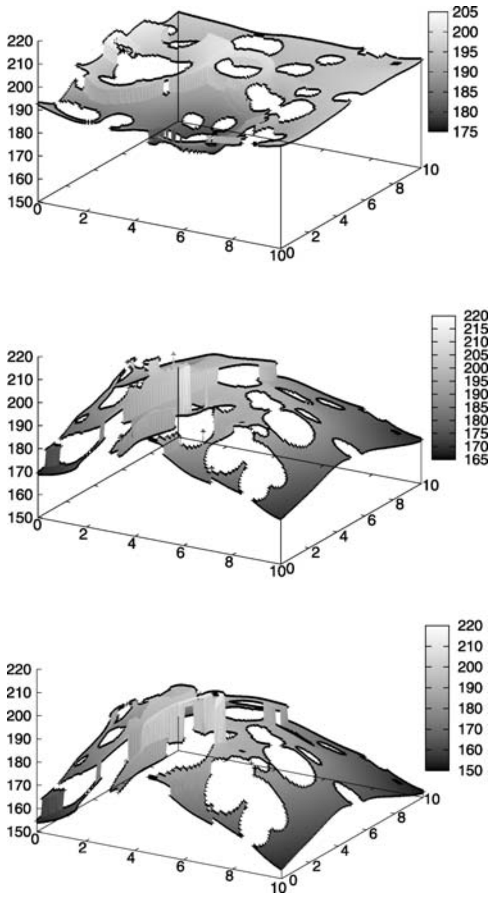


Fig. 1 Plot of Π_1 when $\alpha_1 = 0.5$ (*top*), 3.75 (*middle*) and 5.0 (*bottom*) for a problem with setting ($n = 50$, $m = 8$, $k = 4$)

Π_1 at a given point nf_1, we have to first solve the corresponding medianoid problem $(FP(nf_1))$ to obtain $nf_2^*(nf_1)$. Furthermore, in order to compute the objective value of Π_1 at nf_1 accurately, the follower problem $(FP(nf_1))$ has to be precisely solved, since otherwise, the error of the approximate value can be considerable, even if it is optimal given the non-optimal follower's solution (see Sect. 4.3.3).

To the extent of our knowledge, when considering Huff patronizing behavior, the centroid problem has only been addressed in [5], where a simplified version of the problem considered here was studied (in [5] the qualities of the new facilities were assumed to be known in advance, the objective was the maximization of the market share, and no constraints were allowed). In Fig. 1, for a given problem with $n = 50$, $m = 8$ and $k = 4$, the shape of the objective function Π_1 for three fixed values of α_1 is shown. As we can see, for a fixed value of α_1 the objective function is very mountainous, with many local maxima and minima, and depending on the value of α_1, the shape changes considerably. This can give us an idea of the complexity of the problem at hand, and it also shows the importance of considering quality as an additional variable of the problem.

In the following sections we present heuristics for solving leader problem (*LP*), but first, in the next section, we discuss how to solve the follower problem.

3 Solving the medianoid problem

3.1 WLM: a Weiszfeld-like algorithm

A local method is proposed in [13] (see also [26]) for solving the follower problem. The algorithm is a steepest descent type method which takes discrete steps along the search directions and, usually, converges to a local optimum. In this method, the derivatives of the objective function are equated to zero and the next iterate is obtained by implicitly solving these equations. In location literature these types of methods are known as Weiszfeld-like methods, in honor of E. Weiszfeld, who first proposed that strategy [30].

In the medianoid problem, the follower wants to locate a new facility, knowing the location and the quality of all the facilities of the competitor (the leader). However, throughout the paper, we will also be sometimes interested in solving the opposite problem, in which the leader wants to locate a new facility, assuming that the location and the quality of all the facilities of the competitor (the follower) are fixed. In this case, the leader also has to solve a medianoid problem, but, in which the roles of the leader and the follower are interchanged. We will call this problem a *reverse medianoid* problem.

To take both cases into account, in Appendix A we describe the Weiszfeld-like algorithm (WLM) to solve the medianoid problem, regardless the chain acting as follower. The input parameter of WLM is the set of facilities currently considered.

It is important to note that the Weiszfeld-like algorithm WLM is a *local* procedure. Thus, usually, 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 [13] it is recommended to apply the algorithm repeatedly using (100 or 1000) different starting points, and then select the solution that obtains the maximum profit. However, this still does not guarantee a global optimal solution.

Next, we present another heuristic procedure, UEGO, introduced in [26], which is rather robust, in the sense that it usually finds the global optimum. Furthermore, its running times are competitive, for small to medium size problems, with the multistart technique starting from 1000 points, and it can handle big size problems without any difficulties.

3.2 UEGO_med: UEGO for the medianoid problem

UEGO is a general evolutionary algorithm designed to solve many kinds of multimodal global optimization problems. Some of the real applications that were solved using UEGO were the detection of deformable objects in digital images [15] and the image translational alignment in electron microscopy [27].

In multimodal optimization, the optimizer should discover the structure of the local optima to reach the global optimum. Time should thus be spent in discovering new and promising regions rather than exploring the same region multiple times. UEGO uses non-overlapping sets of clusters which define sub-domains of the search space. As the algorithm progresses, the search process can be directed towards 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 [20]. A particular cluster is not a fixed part of the search domain; it can move through the space as the search proceeds. The non-overlapping property of the set of clusters is maintained. UEGO is abstract in the sense that the 'cluster-management' and the cooling mechanism have been logically separated from the optimization algorithm. Therefore, any kind of optimizer can be used as the optimizer to work inside a cluster. A more detailed description of UEGO can be found in Appendix B.

When dealing with the medianoid problem, WLM is used as local optimizer within UEGO. The resulting algorithm, introduced in [26], will be called UEGO_med throughout this paper. WLM is used during the *Optimize_species* process where UEGO_med tries to optimize every species from its species list. In this procedure, every species calls the Weiszfeld-like algorithm once, using the center of the caller species as the initial point. The Weiszfeld-like algorithm finishes when one of the stopping criteria of WLM is satisfied. If, during the 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 towards the locations of the local optima.

UEGO_med will be used to solve medianoid problems as well as reverse medianoid problems. The first input parameter of UEGO_med allows to distinguish the type of problem being solved: *Follower* means that a medianoid problem is being considered while *Leader* stands for a reverse medianoid problem. The m pre-existing facilities are always taken into account by UEGO_med, but in addition to them, sometimes an additional facility is also considered in the (reverse) medianoid problems. That additional facility, if any, is passed as the second input parameter to UEGO_med. For instance, UEGO_med($Follower, nf_1$) means that the procedure is used to solve the medianoid problem in which nf_1 is also taken into account; UEGO_med($Leader, nf_2$) means that the procedure is used to solve a reverse medianoid problem in which nf_2 is also taken into account; and UEGO_med($Leader, \emptyset$) means that the procedure is used to solve a reverse medianoid problem in which only the m pre-existing facilities are taken into account.

4 Solving the centroid problem

In this section we present four heuristics devised to cope with the leader problem, namely, a grid search procedure, an alternating procedure and two adaptations of the evolutionary algorithm UEGO.

4.1 GS: a grid search procedure

The first method that we have used to cope with the centroid problem is a simple Grid Search procedure (GS). A grid of points that cover the leader's searching region is generated. For each point nf_1 of the grid we first check its feasibility. If it is feasible, we evaluate the objective function $\Pi_1(nf_1, \text{UEGO\_med}(Follower, nf_1))$. Notice that in order to do it, we first have to solve the corresponding medianoid problem by calling UEGO_med to obtain an optimal solution for the follower. When all the feasible points of the grid have been evaluated, a second finer grid is constructed in the vicinity of the point of the first grid having the best objective value. In our first grid, the length of the step between two adjacent points was 0.25 units in each coordinate ($1/40$ the length of the search space), and in the second grid, 0.02 units ($1/500$ the length of the search space). Of course, depending on the size of the search region the grid size may vary; the rule to follow is that the first grid has to be small enough to densely cover the whole region but without doing the process impractical, and the second grid has to be small enough to achieve a reasonable precision.

4.2 AlternatMed: an alternating leader-follower medianoid procedure

The second approach is based on some results in the literature that suggest that a repeated alternate optimization process may lead to a Nash equilibrium, that is, to a solution in which neither the leader nor the follower has an incentive to change its current decision. In continuous competitive location, this principle has already been used in [2], where the $(r|p)$ centroid problem is considered with a Hotelling-type patronizing behavior (see also [22]). In pseudocode form, the alternating leader-follower medianoid procedure (AlternatMed) is described by Algorithm 1.

Two stopping criteria have been considered in Algorithm 1. The first stopping rule which considers the distance between two iteration vectors was implemented as in WLM (see Appendix A), and similarly the second stopping criterion sets a maximum on the number of iterations, $ic_{\max}$. As an initialization step, Line 2 of Algorithm 1 solves the reverse medianoid problem considering the original m existing facilities. Then, at every iteration, a medianoid and a reverse medianoid problem are solved.

A similar algorithm can be devised starting the process by first locating the follower's facility (solving the medianoid problem taking the original m existing facilities into account). In fact, in Step 2, we could set $nf_1^{(0)}$ equal to a random feasible point. However, in a series of preliminary computational studies all the variants produced the same final solutions and required a similar CPU time.

In general, it is unknown whether this type of alternating process always converges to a Nash equilibrium. It is also unknown whether a Nash equilibrium (if any) is an optimal solution of the Stackelberg problem. For our competitive location problem,

Algorithm 1: Algorithm AlternatMed

```
1 Set iteration counter $ic = 0$
2 $nf_1^{(ic)} = \text{UEGO\_med}(Leader, \emptyset)$
3 WHILE Stopping criteria are not met DO
4     $nf_2^{(ic)} = \text{UEGO\_med}(Follower, nf_1^{(ic)})$
5     $nf_1^{(ic+1)} = \text{UEGO\_med}(Leader, nf_2^{(ic)})$
6     $ic = ic + 1$
```

the alternating process may not converge: the leader's solution may alternate between two different feasible solutions as the follower's solution does the same. That is why we also stop AlternatMed if the number of iterations is greater than a given value $ic_{\max}$.

Furthermore, as we will see in Sect. 5, the solutions offered by AlternatMed are not necessarily optimal solutions to the leader problem (*LP*). In the computational studies, the problems for which AlternatMed did not stop according to the first criterion, we set $ic_{\max}$ equal to a value in which the CPU time employed by AlternatMed was similar to that of the UEGO type algorithms described next.

4.3 UEGO_cent: an evolutionary procedure to solve the centroid problem

In view of the excellent results obtained by UEGO when applied to the medianoid problem (algorithm UEGO_med, see [26]), in this section we will describe two algorithms for solving the centroid problem based on the UEGO structure. These algorithms will be called UEGO_cent.WLM and UEGO_cent.SASS. In view of the difficulty of the centroid problem, we have had to make several modifications in UEGO to adapt it to the problem at hand. Next, we describe these modifications:

Species definition A species will be represented by a vector in the form (nf_1, nf_2, r), where nf_1 refers to the leader point, nf_2 to the follower point, and r to the radius of the species. There exists a relationship between nf_1 and nf_2: nf_2 is the solution of the medianoid problem when taking the original m existing facilities and nf_1 into account, i.e. $nf_2 = \text{UEGO\_med}(\mathit{Follower}, nf_1)$.

Species creation A species-list is maintained by the algorithm. Initially, a feasible point nf_1 is randomly chosen, and from it, the first species is constructed. Later on, in a given iteration, we do what follows. For every species in the species-list, we randomly generate feasible points for the leader's facility within the radius of attraction of the species. Then, for each pair of those points, we compute the midpoint of the segment connecting them. If the *approximate* objective value (fitness value) for the leader problem at the midpoint is greater than at the extreme points, then the midpoint becomes a candidate point to generate a new species. Otherwise, if the value of the objective function at both extreme points is greater than at the midpoint then both extreme points are inserted in the sublist of candidate points. The approximate objective values for the leader problem at the points mentioned above are computed considering as follower's facility the one inherited from the species from which they were generated (i.e., we do not solve the corresponding medianoid problem associated to each new point to obtain the correct follower's facility). After this process, for every species in the species-list we have a sublist of 'candidate' points to generate new species. Notice that this process is different from the one used in UEGO_med, since now the center of the species is never replaced by any other point, because we cannot compare the objective value at the midpoints or at the endpoints to the objective value at the center of the species, since the objective value at those points is only an *approximation*.

Unfortunately, this process may generate a large number of candidate points. In order to reduce it, the candidate points are fused as described in the next paragraph.

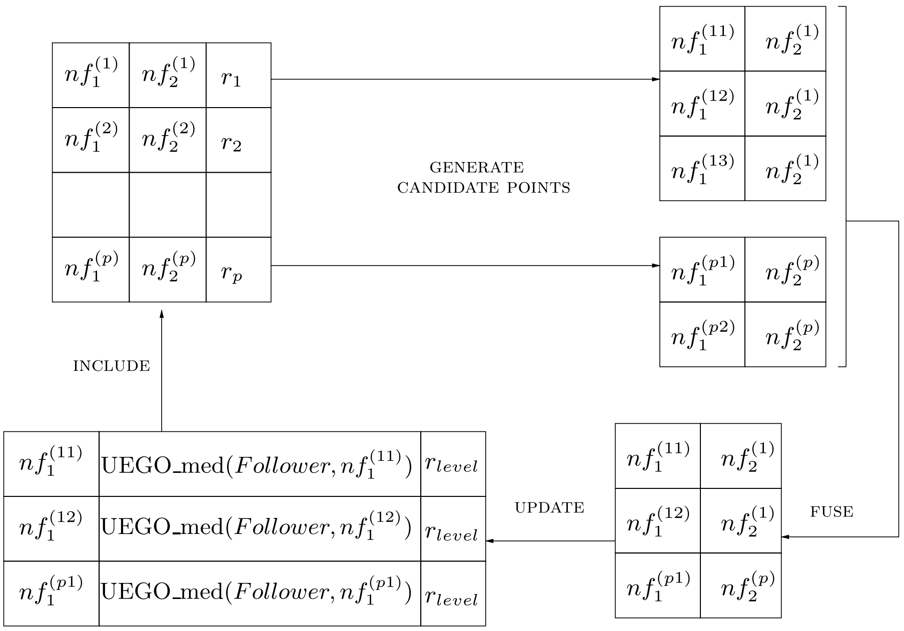

$nf_1^{(1)}$ $nf_2^{(1)}$ r_1
$nf_1^{(2)}$ $nf_2^{(2)}$ r_2
$nf_1^{(p)}$ $nf_2^{(p)}$ r_p
GENERATE CANDIDATE POINTS
$nf_1^{(11)}$ $nf_2^{(1)}$
$nf_1^{(12)}$ $nf_2^{(1)}$
$nf_1^{(13)}$ $nf_2^{(1)}$
$nf_1^{(p1)}$ $nf_2^{(p)}$
$nf_1^{(p2)}$ $nf_2^{(p)}$
INCLUDE
FUSE
UPDATE
$nf_1^{(11)}$ UEGO_med$(Follower, nf_1^{(11)})$ r_{level}
$nf_1^{(12)}$ UEGO_med$(Follower, nf_1^{(12)})$ r_{level}
$nf_1^{(p1)}$ UEGO_med$(Follower, nf_1^{(p1)})$ r_{level}
$nf_1^{(11)}$ $nf_2^{(1)}$
$nf_1^{(12)}$ $nf_2^{(1)}$
$nf_1^{(p1)}$ $nf_2^{(p)}$

Fig. 2 Species creation

After that, for each candidate point in this reduced list we compute its corresponding follower's facility (applying UEGO_med) and then evaluate the correct objective value for the leader's facility. The new species (with the corresponding radius according to the iteration) are inserted in the species-list (see Fig. 2).

Fuse process If the leader's facilities of two species from the species-list are closer to each other than the given radius r_i, then the two species are fused. The new leader's facility will be the one with the best objective value, and the follower's facility will be the corresponding one, while the level will be the minimum of the levels of the original species.

Optimization process For every species in the list a local optimization process is applied. Notice that a local optimizer usually assumes that the configuration of the problem during the optimization process does not change. However, for the centroid problem this is not true, since every time that the leader's facility changes, so does the follower's facility. Thus, the value of the objective function of the leader's problem may change if the new configuration is taken into account. This means that the new follower's facility should be computed every time that the leader's facility changes. However, obtaining the exact new follower's facility using UEGO_med would make the process very time-consuming. That is why we have designed two variants of a new local optimization procedure. The general structure of this procedure is described in Algorithm 2 (LeaderOpt), later on we will provide details of the variants of this procedure.

In Line 2 of Algorithm LeaderOpt, $\mathrm{LO}+\mathrm{WLM}$ is a local search which tries to obtain a better value of the choice of the leader (nf_1) based on the current choice of

Algorithm 2: Algorithm LeaderOpt

```
1 Let $(nf_1, nf_2, r)$ be the species to be optimized
2 $opt\_nf_1 = \mathrm{LO} + \mathrm{WLM}(nf_1, nf_2, r)$
3 $opt\_nf_2 = \mathrm{UEGO\_med}(Follower, opt\_nf_1)$
4 IF $opt\_nf_1 = nf_1$ THEN
5     IF $\Pi_2(nf_1, nf_2) > \Pi_2(nf_1, opt\_nf_2)$ THEN $opt\_nf_2 = nf_2$
6     Update the original species to $(nf_1, opt\_nf_2, r)$
7 ELSE IF $\Pi_1(opt\_nf_1, opt\_nf_2) > \Pi_1(nf_1, nf_2)$ THEN
8     Update the original species to $(opt\_nf_1, opt\_nf_2, r)$
```

the follower (nf_2). It also takes into account the radius (r) of the species which is being optimized. We have designed two variants of the local optimizer LO + WLM, called WLM + WLM and SASS + WLM, and these form the only difference between UEGO_cent.WLM and UEGO_cent.SASS.

4.3.1 WLM + WLM *variant of* LeaderOpt

WLM + WLM procedure uses WLM as a local optimizer for updating both the leader and the follower's facilities. In pseudo-code form, WLM + WLM is described in Algorithm 3. Two stopping rules are considered; the first one takes the distance between two iteration vectors ($nf_1^{(ic-1)}$ and $nf_1^{(ic)}$) into account and it was implemented as in WLM (see Appendix A). The second one is based on the maximum number of iterations $ic_{\max}$. Notice that whereas Steps 3 and 4 are just one iteration of WLM applied to the reverse medianoid problem, Step 5 calls the complete algorithm WLM to solve the corresponding medianoid problem, where *SF* is the set of the m existing facilities. Next, we explain the rationale for those steps.

WLM is an algorithm that uses the locations and qualities of all the existing facilities as input parameters to calculate the coordinates of a new facility in an iterative and deterministic way. Those locations and qualities are assumed to be fixed, and should be as precise as possible in order to obtain a good solution. In WLM + WLM, at each iteration, the leader's facility is modified and hence, the corresponding follower's facility changes accordingly. As a consequence, applying the whole WLM algorithm to the leader's facility without modifying the follower's facility will lead to an incorrect solution. That is why, at each iteration, after modifying the leader's solution, algorithm WLM must be run to solve the corresponding medianoid problem. Notice that the solution provided by WLM to the medianoid problem may be a local (and not a global) one. This means that the information about the follower's location is not always precise, and therefore the calculation of the solution of the leader at the next iteration, $nf_1^{(ic+1)}$, may not be precise either.

Also notice that in WLM the solution point at each iteration is always updated without evaluating the objective function at the new point. This is because in the deterministic way of calculating the coordinates, the new point has a better objective value *for the current setting* than the previous one (WLM is a type of *steepest descent* method). In particular, this means that the leader's facility is updated at each

Algorithm 3: Algorithm WLM + WLM(nf_1, nf_2, r)

```
1 Set iteration counter $ic = 0$; $nf_1^{(0)} = nf_1$ and $nf_2^{(0)} = nf_2$
2 WHILE Stopping criteria are not met DO
3     Compute the next iterate $nf_1^{(ic+1)}$ by equations (A.1) to (A.3) (see
          Appendix A)
4     IF $nf_1^{(ic+1)}$ infeasible THEN $nf_1^{(ic+1)}$ is computed as a point in the segment
          $[nf_1^{(ic)}, nf_1^{(ic+1)}]$ at the border of the feasible region
5     $nf_2^{(ic+1)} = \mathrm{WLM}(SF \cup nf_1^{(ic+1)})$
6     $ic = ic + 1$
7 RETURN $nf_1^{(ic)}$
```

iteration and that the final solution is different from the initial one $nf_1^{(0)}$. For this reason, the condition in Step 4 of Algorithm 2 is never met with this local search. However, although in WLM + WLM, at iteration ic, the calculation of the leader's facility $nf_1^{(ic+1)}$ makes use of the value of the follower's facility from the previous iteration, $nf_2^{(ic)}$, the algorithm does not take the follower's facility $nf_2^{(ic+1)}$ into account to decide whether to replace $nf_1^{(ic)}$ by $nf_1^{(ic+1)}$ or not (comparing their objective values $\Pi_1(nf_1^{(ic+1)}, nf_2^{(ic+1)})$ and $\Pi_1(nf_1^{(ic)}, nf_2^{(ic)})$). It is the Algorithm LeaderOpt, the common local search used in UEGO_cent.WLM and in UEGO_cent.SASS, which does it (see Steps 7 and 8 of Algorithm 2), after evaluating the correct position for the follower (in Step 3).

4.3.2 WLM + SASS *variant of* LeaderOpt

The second variant of the local procedure of LeaderOpt, uses the heuristic SASS + WLM to improve the leader's facility. SASS + WLM uses the stochastic hill climber SASS (see [28]), for updating the leader's facility and WLM for updating the follower's.

Algorithm SASS is a derivative-free optimization algorithm that can be applied to maximize an arbitrary function over a bounded subset of $\mathbb{R}^N$, although internally SASS assumes that the range in which each variable is allowed to vary is the interval [0, 1]. Since this is not our case, when necessary we use a function to rescale (normalize) the variable values to the interval [0, 1], and the function *denorm* to invert this process. The way the heuristic SASS + WLM works is described in Algorithm 4.

In SASS + WLM the new points are generated using a Gaussian perturbation $\xi \in \mathbb{R}^3$ over the search point nf_1 and a normalized bias term $b \in \mathbb{R}^3$ to direct the search. In this way, given nf_1, a first trial point, $nf_1 + \xi$ is considered, and if its objective value is better than that at nf_1, then $nf_1 + \xi$ becomes the new search vector. Otherwise, another trial point $nf_1 - \xi$ is considered and a similar updating procedure is performed.

The coefficient values 0.4 and 0.2 in Steps 13 and 18, used for updating the bias term b are retained from Solis and Wets' results [28]. The standard deviation σ specifies the size of the sphere that most likely contains the perturbation vector, whereas

Algorithm 4: Algorithm SASS + WLM(nf_1, nf_2, σ_{ub})

```
1 Set $ic = 1$, $nf_1^{(ic)} = nf_1$, $nf_2^{(ic)} = nf_2$, $b^{(ic)} = 0$, $scnt = 0$, $fcnt = 0$,
  $\sigma^{(0)} = \sigma_{ub}$, $\sigma_{lb} = \max\{\sigma_{ub}/1000, 10^{-5}\}$
2 Fix $ex$, $ct$, $Scnt$, $Fcnt$, $Maxfcnt$, $ic_{\max}$
3 WHILE $ic < ic_{\max}$ and $fcnt < Maxfcnt$ DO
4     $\sigma^{(ic)} = \sigma^{(ic-1)}$
5     IF $scnt > Scnt$ THEN $\sigma^{(ic)} = ex \cdot \sigma^{(ic-1)}$
6     IF $fcnt > Fcnt$ THEN $\sigma^{(ic)} = ct \cdot \sigma^{(ic-1)}$
7     IF $\sigma^{(ic)} < \sigma_{lb}$ THEN $\sigma^{(ic)} = \sigma_{ub}$ and $b^{(ic)} = 0$
8     IF $\sigma^{(ic)} > \sigma_{ub}$ THEN $\sigma^{(ic)} = \sigma_{ub}$
9     Generate a multivariate Gaussian random vector $\xi_{aux}^{(ic)} = N(b^{(ic)}, \sigma^{(ic)} I)$
      and set $\xi^{(ic)} = denorm(\xi_{aux}^{(ic)})$
10    IF $\Pi_1(nf_1^{(ic)} + \xi^{(ic)}, nf_2^{(ic)}) > \Pi_1(nf_1^{(ic)}, nf_2^{(ic)})$ THEN
11        $nf_2^{(ic+1)} = \mathrm{WLM}(SF \cup \{nf_1^{(ic)} + \xi^{(ic)}\})$
12        IF $\Pi_1(nf_1^{(ic)} + \xi^{(ic)}, nf_2^{(ic+1)}) > \Pi_1(nf_1^{(ic)}, nf_2^{(ic)})$ THEN
13            $nf_1^{(ic+1)} = nf_1^{(ic)} + \xi^{(ic)}$; $b^{(ic+1)} = 0.4\xi_{aux}^{(ic)} + 0.2b^{(ic)}$,
              $scnt = scnt + 1$, $fcnt = 0$
14    ELSE
15        IF $\Pi_1(nf_1^{(ic)} - \xi^{(ic)}, nf_2^{(ic)}) > \Pi_1(nf_1^{(ic)}, nf_2^{(ic)})$ THEN
16            $nf_2^{(ic+1)} = \mathrm{WLM}(SF \cup \{nf_1^{(ic)} - \xi^{(ic)}\})$
17            IF $\Pi_1(nf_1^{(ic)} - \xi^{(ic)}, nf_2^{(ic+1)}) > \Pi_1(nf_1^{(ic)}, nf_2^{(ic)})$ THEN
18                $nf_1^{(ic+1)} = nf_1^{(ic)} - \xi^{(ic)}$, $b^{(ic+1)} = b^{(ic)} - 0.4\xi_{aux}^{(ic)}$,
                  $scnt = scnt + 1$, $fcnt = 0$
19        ELSE $nf_1^{(ic+1)} = nf_1^{(ic)}$, $nf_2^{(ic+1)} = nf_2^{(ic)}$, $b^{(ic+1)} = 0.5b^{(ic)}$,
          $fcnt = fcnt + 1$, $scnt = 0$
20    $ic = ic + 1$
22 RETURN $nf_1^{(ic)}$
```

the bias term b locates the center of the sphere based on directions of past successes. The size of the standard deviation of the normalized perturbation ξ_{aux} is controlled by the repeated number of successes, *scnt*, or failures, *fcnt*, of increasing the objective function Π_1. The contraction ct and expansion ex constants, as well as the upper bound σ_{ub} on the standard deviation σ are set by the user.

The stopping rules are determined by the maximum number of iterations ($ic_{\max}$) and by the maximum number of consecutive failures (*Maxfcnt*). Notice that $2 \cdot ic_{\max}$ is a lower bound of the minimum number of function evaluations made by the algorithm.

When solving the centroid problem, it is very important to have an optimal solution for the follower's facility in order to correctly evaluate the leader's objective function Π_1. Otherwise, the leader's solution may be completely wrong both in the location and the value of the objective function. That is why in Steps 11 and 16 of Algorithm SASS + WLM, before taking a decision about a possible updating of the leader solution $nf_1^{(ic)}$, the algorithm WLM is applied to solve the corresponding medianoid problem that takes the original m existing facilities and $nf_1^{(ic)} + \xi^{(ic)}$ or $nf_1^{(ic)} - \xi^{(ic)}$, respectively, into account. Although the follower's solution $nf_2^{(ic)}$ may be only an approximation of the global solution, it is useful to evaluate the function Π_1 at the new leader's choice using it, and then to take a decision about the updating of the leader's facility $nf_1^{(ic)}$ accordingly.

SASS + WLM is called (through LeaderOpt) by UEGO_cent.SASS in the optimization process when optimizing the species list. Bearing in mind that in the optimization process a species has assigned a budget for the number of function evaluations (n_i/M), SASS + WLM sets $ic_{\max}$ equal to n_i/M iterations. As for the species radius, it is mentioned in Appendix B that a species can be seen as a window whose aim is to focus the optimizer, in such a way that any single step taken by the optimizer in a given species is no longer than the radius of the species. Since in SASS + WLM the standard deviation σ specifies the size of the sphere that most likely contains the normalized perturbation vector, its upper bound σ_{ub} should have the same value than the normalized radius of the caller species. That is why the parameter σ_{ub} is also considered an argument of SASS + WLM.

Contrary to WLM + WLM, in SASS + WLM, it may happen that $opt\_nf_1 \equiv nf_1$ (if the algorithm is not able to find a new solution that improves the current solution, the final solution may coincide with the initial one). This usually happens when the initial point nf_1 is a local or global maximum. When the condition of Step 4 in Algorithm 2 holds, two followers' choices nf_2 and opt_nf_2 for the same leader choice nf_1 are considered. Both choices are usually quite similar but not the same. In this case, in order to get the best evaluation of the leader's objective function Π_1, the follower's choice that maximizes Π_2 must be selected.

4.3.3 On the solution of the medianoid problems

For both UEGO_cent.WLM and UEGO_cent.SASS to work properly, it is very important to find the global optimum (and not a local one) of the medianoid problems. If they are not solved *optimally* (even if the solutions are very close to optimality in the value of the objective function but are in significantly different locations) then the objective value for the leader will be completely wrong, overestimated, and this even if the leader's problem is solved optimally given the non-optimal follower's solution. This will make imposible to determine whether a given point is a true local or global maximum. In fact, in preliminary computational studies where we used the local procedure WLM for solving the medianoid problems the solutions obtained by both UEGO_cent.WLM and UEGO_cent.SASS were in some cases wrong. That is why we have used UEGO_med in Step 3 of Algorithm 2 for solving the medianoid problems: in a large set of test problems solved in [26], UEGO_med *always* obtained the optimal solution.

5 Computational studies

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

5.1 The test problems

To study 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 leader's chain. The actual settings (n, m, k) employed are detailed in the first column of Table 10.

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

- $p_i, f_j \in ([0, 10], [0, 10])$,
- $\omega_i \in [1, 10]$,
- $\gamma_i \in [0.75, 1.25]$,
- $\alpha_{ij} \in [0.5, 5]$,
- $c_1 = c_2 \in [1, 2]$, the parameter for $F_l(M_l(*)) = c_l \cdot M_l(*), l = 1, 2$,
- $G_1 = G_2 = G$, with $G(nf_l) = \sum_{i=1}^{n} \Phi^i(d_{iz_l}) + G^b(\alpha_l)$ where
  - $\Phi^i(d_{iz_l}) = w_i \frac{1}{(d_{iz_l})^{\phi^{i0}} + \phi^{i1}}$ with $\phi^{i0} = \phi^0 = 2$ and $\phi^{i1} \in [0.5, 2]$,
  - $G^b(\alpha_l) = e^{\frac{\alpha_l}{\xi^0} + \xi^1} - e^{\xi^1}$ with $\xi^0 \in [7, 9]$ and $\xi^1 \in [4, 4.5]$,
- $d_{iz_l} = \sqrt{b_1(x_l - p_{i1})^2 + b_2(y_l - p_{i2})^2}, l = 1, 2$, with $b_1, b_2 \in [1, 2]$.

The searching space for every problem was

$$z_l = (x_l, y_l) \in S_1 = S_2 = ([0, 10], [0, 10]), \quad l = 1, 2,$$
$$\alpha_l \in [0.5, 5], \quad l = 1, 2.$$

5.2 UEGO parameter setting

In [26] it was found that a good parameter setting for UEGO_med was $N = 10^6$, $M = 150$, $L = 30$ and $r_L = 0.025$. UEGO_med has been used for solving the (reverse) medianoid problems.

However, for solving the centroid problem the parameter setting has been modified by reducing the number of levels to $L = 10$ (the remaining parameters keep the same value as in UEGO_med). This modification has been introduced to reduce the number of iterations and therefore the computational cost. Remember that one evaluation of the objective function of the centroid problem implies the execution of UEGO_med to find the corresponding follower's facility. And the optimization process implies the running of a local optimization algorithm that also needs to optimize the follower's facility. So the centroid problem requires a quite expensive computational cost. The smaller the number of levels is, the smaller the number of iterations, and therefore the smaller the number of times that the optimization process is called. The value $L = 10$ still allows to obtain very good results when solving the centroid problem.

5.3 Comparing the performance of the algorithms

Before presenting some tables summarizing the computational studies, there are some remarkable facts that deserve to be highlighted.

5.3.1 GS *may get trapped on a local maximum*

Grid search methods are commonly thought to be reliable (although rather time-consuming) methods. However, there is no guarantee that a grid search method will result in a global optimum. If the objective function value increases dramatically in a small neighborhood around the global optimum and the grid is not dense enough, the second finer grid can focus around a local optimum. Something similar can happen if a local optimum exists whose objective value is close to the global optimum value and the grid is not dense enough. The risk of failure is even higher in the presence of constraints, as happens in our centroid problem, since it may occur that the global optimum is surrounded (in part) by infeasible areas, and the grid may not have a feasible point near to the global optimum. Of course, the finer the grids, the higher the possibilities for the method to find the optimum, but one never knows how small the distance between two adjacent points in the grid should be, and regardless how small that distance is, it may still happen that the search does not reach the global optimum. For six out of fourteen problems we have solved, GS was not able to find the global optimum.

As an example, in Table 1 we give the results for a problem with $n = 50$, $m = 5$ and $k = 0$. Since each run of AlternatMed, UEGO_cent.WLM and UEGO_cent.SASS may provide a different solution, each of those heuristics has been run five times for each problem. In the column labelled ‘Time’ we give, for each of them, the average time in the five runs (in seconds), in ‘Best solution’ column the best solution found in the five runs, in ‘MaxDist’ column the maximum Euclidean distance between any pair of solutions provided by the algorithm (this gives us an idea of how far the solutions provided by an algorithm in different runs can be), the next three columns give the minimum, the average and the maximum objective value in the five runs, and in ‘Dev’ the standard deviation. GS has been run only once, thus, for that algorithm we give the corresponding values obtained with it.

The best solution obtained by the different algorithms are also depicted (projected onto the two-dimensional locational space) in Fig. 3. In that figure, and also in the

Table 1 Results for a problem with setting ($n = 50$, $m = 5$, $k = 0$)

Algorithm	Time (s)	Best solution			MaxDist	Objective function			Dev
		x_1	y_1	α_1		Min	Av	Max	
GS	247510	2.480	7.360	4.180	–	86.703	86.703	86.703	–
AlternatMed	1407	5.461	6.660	5.000	0.010	77.381	77.670	77.700	0.159
UEGO_cent.WLM	1391	5.430	7.101	4.647	3.410	81.406	84.485	92.759	4.221
UEGO_cent.SASS	1133	5.417	6.906	4.851	0.010	93.652	93.894	94.044	0.147

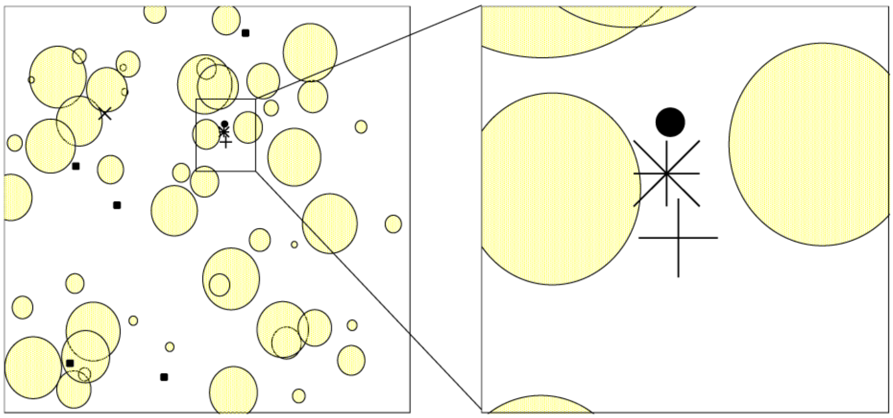


Fig. 3 Example with $n = 50$, $m = 5$, $k = 0$. GS $= \times$, AlternatMed $= +$, UEGO_cent.WLM $= \bullet$ and UEGO_cent.SASS $= *$

Table 2 Results for a problem with setting ($n = 50$, $m = 5$, $k = 1$)

Algorithm	Time (s)	Best solution			MaxDist	Objective function			Dev
		x_1	y_1	α_1		Min	Av	Max	
GS	371445	3.090	7.260	2.440	–	132.296	132.296	132.296	–
AlternatMed	1804	4.038	6.123	3.598	0.050	121.225	121.511	121.761	0.172
UEGO_cent.WLM	1793	4.558	5.943	3.910	0.468	124.482	131.475	136.070	4.969
UEGO_cent.SASS	1719	4.917	5.150	3.418	0.002	143.197	143.488	143.498	0.144

rest of the figures in this section, the black triangles (▲) give the locations of the existing leader's facilities, the black squares (■) correspond to the locations of the existing follower's facilities, the symbol $\times$ gives the solution found by GS, and the plus sign $+$, the bullet $\bullet$ and the start $*$ give the best solution obtained by AlternatMed, UEGO_cent.WLM and UEGO_cent.SASS, respectively, in the five runs. Light gray ovals represent the forbidden areas around the existing demand points, which are at the center of those ovals (the greater the oval, the greater the purchasing power at the demand point). Notice that in Fig. 3 there are no triangles since $k = 0$.

As we can see, in this example the solution provided by GS is far from the area were the optimal solution seems to be. It focuses on a local maximum whose objective value is close to the optimal one. Notice also that quite close points may yield very different objective values (see the best points found by the last three algorithms and their corresponding objective values in the ninth column), showing that the objective function can locally be quite steep.

In Table 2 and Fig. 4 we can see another problem, with setting ($n = 50$, $m = 5$, $k = 1$), where GS fails to find the global optimum, although this time the reason seems to be that the global optimum is between two unfeasible regions and none of the grid points close to it is feasible. This example clearly shows that the grid procedure is not a good strategy when the feasible set is a highly restricted region. In

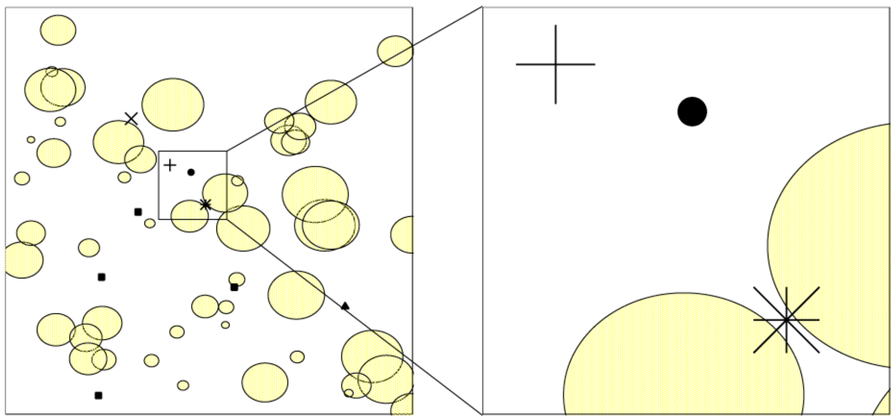


Fig. 4 Example with $n = 50$, $m = 5$, $k = 1$. GS $= \times$, AlternatMed $= +$, UEGO_cent.WLM $= \bullet$ and UEGO_cent.SASS $= *$

Table 3 Results for a problem with setting ($n = 50$, $m = 6$, $k = 3$)

Algorithm	Time (s)	Best solution			MaxDist	Objective function			Dev
		x_1	y_1	α_1		Min	Av	Max	
GS	429389	1.200	4.240	3.770	–	292.337	292.337	292.337	–
AlternatMed	187	1.161	4.222	3.525	0.002	292.412	292.449	292.490	0.027
UEGO_cent.WLM	1309	1.164	4.223	3.528	0.109	292.328	292.413	292.500	0.061
UEGO_cent.SASS	1376	1.161	4.222	3.663	0.102	292.509	292.530	292.554	0.015

fact, it may happen that no feasible grid point exists. One way to try to avoid this is to reduce the grid size, but one never knows how much it should be reduced.

5.3.2 AlternatMed *may not converge and may get trapped on a local maximum*

Although in some problems AlternatMed converges quickly to the global optimum (see for instance the results for a problem with setting ($n = 50$, $m = 6$, $k = 3$) in Table 3), there are instances in which the alternating process does not converge: the leader's solution alternate between two different points as the follower's solution does the same. Consider, for instance, the problem with setting ($n = 50$, $m = 8$, $k = 4$) depicted in Fig. 5 and whose results are summarized in Table 4. In that problem the leader's facility jumps from point (1.899, 4.401, 2.183) to point (5.849, 2.690, 2.847) and the objective varies from 201.904 to 211.660, respectively. Accordingly, the follower's facility jumps from point (5.958, 2.232, 3.017) to point (1.899, 4.401, 2.824). Although in this example the points between which the leader's facility alternates are close to those between which the follower's facility alternates, there are other problems in which the alternating points of the leader and the follower are far from each other. Notice that when AlternatMed alternates between two points, then the objective function value goes up and down accordingly. In these cases, the solution proposed by AlternatMed is the one offering the highest profit.

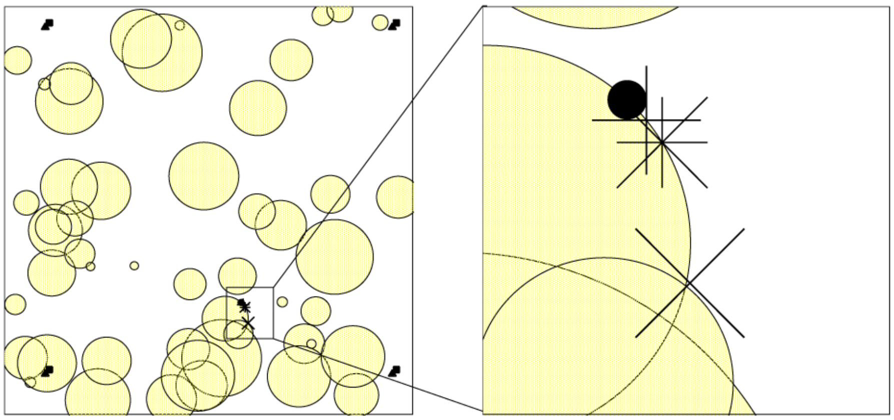


Fig. 5 Example with $n = 50$, $m = 8$, $k = 4$. GS $= \times$, AlternatMed $= +$, UEGO_cent.WLM $= \bullet$ and UEGO_cent.SASS $= *$

Table 4 Results for a problem with setting ($n = 50$, $m = 8$, $k = 4$)

Algorithm	Time (s)	Best solution			MaxDist	Objective function			Dev
		x_1	y_1	α_1		Min	Av	Max	
GS	433918	5.970	2.240	3.750	–	219.718	219.718	219.718	–
AlternatMed	2789	5.849	2.690	2.847	0.030	211.648	211.660	211.682	0.014
UEGO_cent.WLM	2738	5.796	2.747	2.863	3.723	214.648	220.111	222.291	2.854
UEGO_cent.SASS	2954	5.893	2.629	2.864	0.242	221.368	222.775	223.983	1.040

Table 5 Results for a problem with setting ($n = 50$, $m = 6$, $k = 3$)

Algorithm	Time (s)	Best solution			MaxDist	Objective function			Dev
		x_1	y_1	α_1		Min	Av	Max	
GS	371290	4.050	3.050	4.240	–	230.239	230.239	230.239	–
AlternatMed	2496	4.636	3.033	4.300	0.000	229.687	229.688	229.688	0.000
UEGO_cent.WLM	1336	4.638	3.034	4.300	0.006	230.149	230.152	230.154	0.002
UEGO_cent.SASS	2140	4.103	3.055	4.255	0.047	230.238	230.257	230.329	0.036

There are also problems in which AlternatMed converges, but to a local maximum, as can be seen in Table 5, which gives the results for another problem with setting ($n = 50$, $m = 6$, $k = 3$). The results are depicted in Fig. 6.

5.3.3 UEGO_cent.WLM *may get trapped on a local maximum*

Contrary to SASS + WLM, the local optimizer WLM + WLM does not adapt itself well enough to the problem at hand. It does not explore the complete neighborhood around the point to be optimized, but just tries to improve along one direction. Since the configuration of the problem changes every time the leader changes its facility,

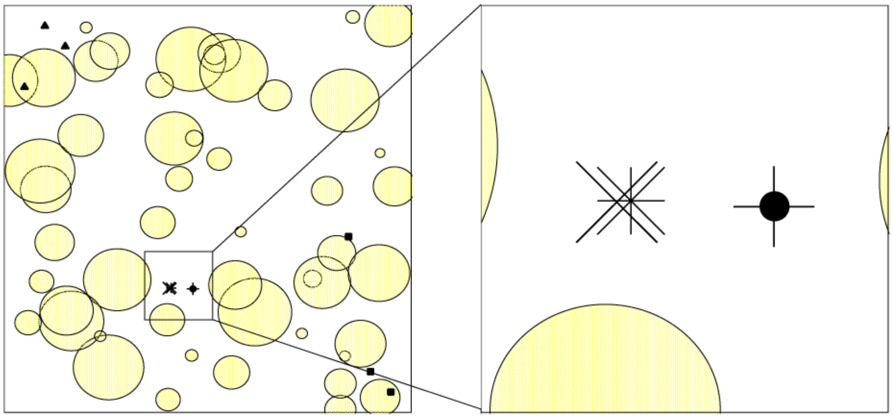


Fig. 6 Example with $n = 50$, $m = 6$, $k = 3$. GS $= \times$, AlternatMed $= +$, UEGO_cent.WLM $= \bullet$ and UEGO_cent.SASS $= *$

Table 6 Results for a problem with setting ($n = 50$, $m = 6$, $k = 3$)

Algorithm	Time (s)	Best solution			MaxDist	Objective function			Dev
		x_1	y_1	α_1		Min	Av	Max	
GS	390625	7.160	2.980	4.140	–	209.831	209.831	209.831	–
AlternatMed	1779	7.161	2.979	4.000	0.050	183.535	183.893	184.234	0.224
UEGO_cent.WLM	1485	4.492	5.055	3.844	4.570	200.801	204.878	208.724	3.227
UEGO_cent.SASS	1818	7.151	3.487	3.123	0.008	212.045	212.170	212.358	0.154

this may provoke the algorithm UEGO_cent.WLM to stop at a local maximum. Some problems in which UEGO_cent.WLM has become trapped on a local maximum are those considered in Tables 1, 2, 5 and the one considered in Table 6.

5.3.4 *Summarizing results*

In Table 7 we can see the average results for the nine problems with $n = 50$ demand points that we have solved. As we can see, UEGO_cent.SASS is the algorithm which provides the best results (in fact, in all the problems, it was the algorithm giving the best results), followed by GS and UEGO_cent.WLM, which provide very similar results. AlternatMed is the algorithm providing the worst results. In both UEGO_cent.SASS and AlternatMed the different runs usually attain the same solution, whereas UEGO_cent.WLM is more erratic, and may provide different solutions in each run (see the values of MaxDist and Dev), thus confirming that it can get trapped in local maxima.

In Table 8 we can see the average results for three problems with $n = 100$ demand points that we have solved, and in Table 9 the average results when considering all the problems with $n = 50$ and 100 demand points, and two more problems with $n = 21$. Similar conclusions can be inferred from those tables. Notice that the increase from $n = 50$ to $n = 100$ makes the problems much harder to solve, as we can see from the

Table 7 Average results for all the problems with $n = 50$

Algorithm	Time (s)	MaxDist	Objective function			Dev
			Min	Av	Max	
GS	358316	–	–	164.528	–	–
AlternatMed	1534	0.027	158.173	158.287	158.367	0.080
UEGO_cent.WLM	1495	1.503	161.338	163.675	165.838	1.782
UEGO_cent.SASS	1728	0.128	166.975	167.225	167.425	0.184

Table 8 Average results for the problems with $n = 100$

Algorithm	Time (s)	MaxDist	Objective function			Dev
			Min	Av	Max	
GS	560019	–	–	160.787	–	–
AlternatMed	4156	0.027	160.587	160.599	160.621	0.014
UEGO_cent.WLM	8344	0.111	160.838	160.868	160.884	0.016
UEGO_cent.SASS	8305	0.031	160.881	160.887	160.896	0.006

Table 9 Average results considering all the problems

Algorithm	Time (s)	MaxDist	Objective function			Dev
			Min	Av	Max	
GS	351576	–	–	208.964	–	–
AlternatMed	1927	0.020	204.163	204.207	204.242	0.033
UEGO_cent.WLM	3484	0.831	207.932	208.970	210.133	0.863
UEGO_cent.SASS	3481	0.111	210.560	210.696	210.830	0.116

CPU times needed to solve them. This seems to suggest that a parallelization of the algorithm may be appropiate when handling larger problems.

5.4 The cost of a myopic decision

To measure how important it is to take the follower's reaction into account, we have conducted a final study in which, for each of the fourteen problems generated for the studies of the previous subsections, we have calculated the leader's profit in case the leader did not take the future facility that the follower will set up into account. With this aim, we have first solved the corresponding reverse medianoid problem taking the original m facilities into account. If we denote its optimal solution with $nf_1^{(myop)}$, then we have solved the corresponding medianoid problem taking the existing m facilities and $nf_1^{(myop)}$ into account. And finally, we have evaluated $\Pi_1^{(myop)} = \Pi_1(nf_1^{(myop)}, \text{ UEGO\_med}(Follower, nf_1^{(myop)}))$.

In Table 10 we can see the results obtained. In the first column we can see the setting of the problems solved (for three settings more than one problem was gener-

Table 10 Comparison between the myopic and the long term view

(n, n, k)	$nf_1^{(myop)}$			$\Pi_1^{(myop)}$	nf_1^*			Π_1^*	% lost
	x_1	x_2	α_1		x_1	x_2	α_1		
(21, 5, 2)	2.234	3.352	1.524	226.645	2.981	4.482	2.218	228.394	0.76
(21, 5, 3)	3.024	6.576	0.536	363.451	2.234	3.352	1.162	379.943	4.34
(50, 5, 0)a	6.082	2.378	2.230	9.156	6.082	2.378	2.230	9.156	0.00
(50, 5, 0)b	5.419	6.411	5.000	67.569	5.417	6.906	4.851	94.044	28.15
(50, 5, 1)	4.452	5.920	3.839	116.424	4.917	5.150	3.418	143.498	18.87
(50, 5, 2)a	2.264	2.096	2.421	189.113	2.228	2.138	2.122	189.653	0.28
(50, 5, 2)b	3.573	4.044	2.554	109.514	3.572	4.044	2.549	111.246	1.56
(50, 6, 3)a	1.122	3.362	3.224	291.052	1.161	4.222	3.663	292.554	0.51
(50, 6, 3)b	1.733	5.848	3.991	194.486	7.151	3.487	3.123	212.358	8.42
(50, 6, 3)c	6.851	3.459	4.486	218.890	4.103	3.055	4.255	230.329	4.97
(50, 8, 4)	5.677	2.830	2.973	198.546	5.893	2.629	2.864	223.983	11.36
(100, 2, 0)	4.471	4.704	5.000	168.430	4.724	4.591	5.000	169.717	0.76
(100, 2, 1)	3.379	6.298	5.000	271.951	3.255	6.366	5.000	272.027	0.03
(100, 10, 0)	2.758	5.119	5.000	40.944	2.758	5.119	5.000	40.944	0.00

ated, and we have added the letters a, b, and c at the end of the setting to highlight it), in the second and third columns we give $nf_1^{(myop)}$ and $\Pi_1^{(myop)}$, in the following two columns we give the facility (nf_1^*) and profit (Π_1^*) obtained with UEGO_cent.SASS, and in the last column the loss in profit caused by the myopic decision as compared to the long term decision, in percentage.

As we can see, although in half of the problems the loss is almost negligible (less than 1%), in 6 out of 14 problems it is over 4%, and in three of them over 11%. This clearly indicates the usefulness of anticipating a competitive entry, since the gains that can be achieved are substantial. Furthermore, this is true regardless the configuration (n, m, k) of the problem: the number of existing facilities belonging to the leader as compared to the total number of existing facilities has no influence on this (in fact, the two extreme cases, with 0% loss and 28.15% loss, have the same configuration (50, 5, 0)). What matters is the actual distribution of the demand points over the region and the actual locations and qualities of the existing facilities: as we can see, $nf_1^{(myop)}$ is usually close to nf_1^*, whereas in some problems the value of $\Pi_1^{(myop)}$ is also close to Π_1^*, in other problems the difference is very high.

6 Conclusions and future research

In this study we have dealt with a type of competitive facility location problem in continuous space which, due to its difficulty, has only been addressed in very few papers before. The problem is known as (1|1)-centroid (Stackelberg or Simpson) problem and can be described as follows. A chain (the leader) wants to set up a single new facility in a planar market where similar facilities of a competitor (the follower), and possibly of its own chain, are already present. After the location of the leader's facility, the competitor will react by locating another new facility at the place that maximizes its own profit. The objective of the leader is to find the location and the quality of its facility that maximizes its profit following the reaction of the follower. The patronizing behavior of customers is assumed to be probabilistic. Notice that contrary to what is commonly done in existing literature, we not only consider the decision in location but also in quality (design of the new facility). This is the first paper in which that general centroid problem is considered.

We have introduced four heuristics for handling the problem, namely, a grid search procedure, an alternating method and two evolutionary algorithms. The computational studies have shown that the problem is very difficult to solve, with many local maxima and in some instances with very different objective values at quite close feasible points. These facts, together with the presence of constraints, provoke that in some problems the heuristics become trapped in local solutions. Only the evolutionary algorithm called UEGO_cent.SASS seems to be able to overcome those difficulties. This is in part due to the use of SASS as local optimizer within UEGO. The configuration of the problem changes every time that the leader changes the location and/or the quality of the new facility, and whereas other local optimizers do not capture those changes well enough, SASS carries out a more exhaustive search over the region which allows it to adapt itself to the changes.

The computational studies have also shown that it is very important to take the future reaction of the follower into account, since the loss in profit when this is not done can be very high (in some cases the loss can be greater than 25%).

In the future we plan to parallelize UEGO_cent.SASS, so that it can solve larger problems and quicker. We also plan to carry out a sensitivity analysis of the optimal solution of the problem to change in the different parameters that define the problem.

Appendix A: Weiszfeld-like algorithm WLM

In this section we describe the Weiszfeld-like algorithm to solve the medianoid problem, regardless the chain acting as follower. With this aim, let us denote $nf = (z, \alpha) = (x, y, \alpha)$ the location and quality of the new facility to be located, $\hat{m}$ the number of existing facilities (the first $\hat{k}$ of those facilities belong to the chain that wants to locate the new facility), $\widehat{SF}$ the set of the $\hat{m}$ facilities, S the location space where the new facility has to be located, and $q^{\min}$ (resp. $q^{\max}$) the minimum (resp. maximum) allowed quality for the new facility. The objective function is assumed to be given by $\Pi(nf) = F(M(nf)) - \sum_{i=1}^{n} \Phi^i(z) - G^b(\alpha)$. Let us also denote

$$s_i = \sum_{j=1}^{\hat{m}} \frac{\alpha_{ij}}{g_i(d_{ij})}, \qquad t_i = w_i \sum_{j=\hat{k}+1}^{\hat{m}} \frac{\alpha_{ij}}{g_i(d_{ij})},$$

$$H_i(nf) = \frac{\partial \Pi}{\partial d_{iz}} = -\frac{\mathrm{d}F}{\mathrm{d}M} \cdot \frac{\alpha \gamma_i t_i g_i'(d_{iz})}{(\gamma_i \alpha + s_i g_i(d_{iz}))^2} - \frac{\mathrm{d}\Phi^i}{\mathrm{d}d_{iz}},$$

Algorithm 5: WLM($\widehat{SF}$). Weiszfeld-like algorithm

```
1 Set iteration counter $ic = 0$
2 Initialize $nf^{(0)} = (x^{(0)}, y^{(0)}, \alpha^{(0)})$
3 WHILE Stopping criteria are not met DO
4     Update $nf^{(ic+1)} = (x^{(ic+1)}, y^{(ic+1)}, \alpha^{(ic+1)})$
5     IF $nf^{(ic+1)}$ infeasible THEN $nf^{(ic+1)}$ is computed as a point in the segment
          $[nf^{(ic)}, nf^{(ic+1)}]$ at the border of the feasible region
6     $ic = ic + 1$
```

where d_{iz} is a distance function such that

$$\frac{\partial d_{iz}}{\partial x} = xA_{i1}(z) - B_{i1}(z), \qquad \frac{\partial d_{iz}}{\partial y} = yA_{i2}(z) - B_{i2}(z),$$

where $A_{it}(z)$ and $B_{it}(z)$ are functions of z. Then the Weiszfeld-like algorithm for solving the corresponding medianoid problem is described by Algorithm 5 (for more details see [13]).

Values of $x^{(ic+1)}$ and $y^{(ic+1)}$ in Algorithm 5 are obtained as:

$$x^{(ic+1)} = \frac{\sum_{i=1}^{n} H_i(nf^{(ic)})B_{i1}(z^{(ic)})}{\sum_{i=1}^{n} H_i(nf^{(ic)})A_{i1}(z^{(ic)})}, \tag{A.1}$$

$$y^{(ic+1)} = \frac{\sum_{i=1}^{n} H_i(nf^{(ic)})B_{i2}(z^{(ic)})}{\sum_{i=1}^{n} H_i(nf^{(ic)})A_{i2}(z^{(ic)})}, \tag{A.2}$$

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

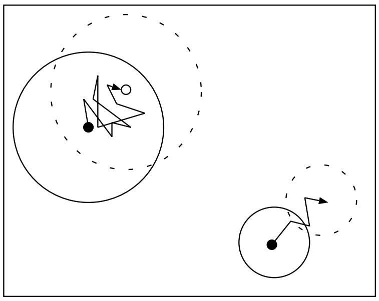
$$\frac{\mathrm{d}F}{\mathrm{d}M} \cdot \sum_{i=1}^{n} \frac{\gamma_i t_i g_i(d_{iz^{(ic+1)}})}{(\gamma_i \alpha + r_i g_i(d_{iz^{(ic+1)}}))^2} - \frac{\mathrm{d}G_2}{\mathrm{d}\alpha} = 0. \tag{A.3}$$

Two stopping rules are applied in WLM. The first stopping criterion stops the algorithm if $\|(x^{(ic-1)}, y^{(ic-1)}) - (x^{(ic)}, y^{(ic)})\|_2 < \epsilon_1$ and $|\alpha^{(ic-1)} - \alpha^{(ic)}| < \epsilon_2$, for given tolerances $\epsilon_1, \epsilon_2 > 0$. The second stopping criterion, which sets a maximum number of iterations $ic_{\max}$, is needed because the convergence of the algorithm cannot be guaranteed. In our computational studies we have set $\epsilon_1 = \epsilon_2 = 10^{-3}$ and $ic_{\max} = 400$.

Appendix B: UEGO algorithm

This section shows a description of the evolutionary algorithm UEGO, one of whose core parts is 'cluster-management', as mentioned in Sect. 2. It consists of procedures for creating, merging 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 Fig. 7). This window

Fig. 7 Concept of species



is defined by its center and a radius. The center is a solution, and the radius is a positive number. In particular, for the (reverse) medianoid problems a species is an array of the form (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 an 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 therefore multiple solutions, so a species would be equivalent to a sub-population or cluster in an evolutionary algorithm based on sub-populations (clustering algorithm).

This definition assumes a *distance* defined over the search space. The role of this window is to 'focus' the optimizer that is always called by a species so that it can 'see' only within the window, so every new sample is taken from there. This means that any single step made by the optimizer in a given species is no larger than the radius of the given species. If the value of a new solution 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 Fig. 7).

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

$$r_i = r_1 \left(\frac{r_L}{r_1}\right)^{\frac{i-1}{L-1}} \qquad (i = 2, \ldots, L).$$

The parameter L indicates the maximal number of levels in the algorithm, i.e. the number of different 'cooling' stages. Every level i (with $i \in [1, L]$) has a radius value (r_i) and two maxima on the number of function evaluations, namely new_i (maximum number of function evaluations allowed when creating new species) and n_i (maximum number of function evaluations allowed when optimizing individual species).

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 maximal length of the species list is given by *max_spec_num* (*maximum population size*).

B.1 Input parameters

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

The reader is referred to [23] for an in-depth description of these input parameters and their relationship to the parameters at each level.

B.2 The algorithm

A global description of UEGO is given in Algorithm 6.

In the following, the different key stages of the algorithm are described:

- *Init_species_list*: A species list, consisting of a single species with a random center at level 1, is created.
- *Create_species*(*creat_evals*): For every species in the list, random trial points in the 'window' of the species are created. Those trial points are combined to form all the possible pairs of potential species. For every pair of points the objective function is evaluated at the midpoint of the *segment* connecting the pair (see Fig. 8). If the objective value at the midpoint is better than at the center of the species, the center is replaced by the midpoint (hence, the species moves towards an optimum). Furthermore, if the value of the objective function at the midpoint is worse than the values at the members of the pair, then the members of the pair are inserted into the species list. As a consequence of this procedure, the

Algorithm 6: Algorithm UEGO

```
1 Init_species_list
2 Optimize_species($n_1$)
3 FOR $i = 2$ to levels
4     Determine $r_i$, $new_i$, $n_i$
5     Create_species($new_i$)          # budget_per_species = $new_i$/length(species_list)
6     Fuse_species($r_i$)
7     Shorten_species_list(max_spec_num)
8     Optimize_species($n_i$)          # budget_per_species = $n_i$/max_spec_num
9     Fuse_species($r_i$)
```

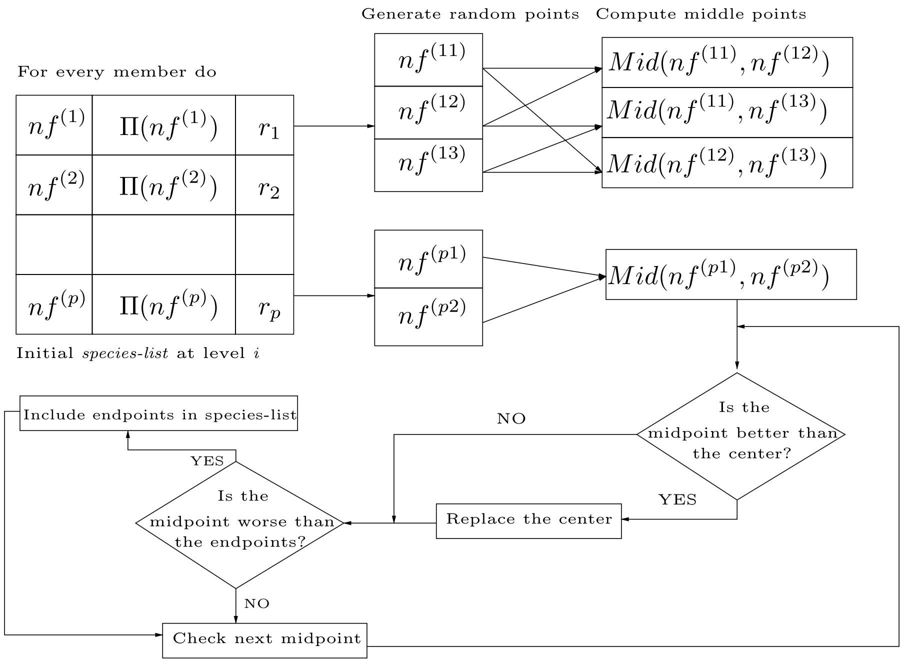

Generate random points
Compute middle points
For every member do
$nf^{(11)}$
$nf^{(12)}$
$nf^{(13)}$
$Mid(nf^{(11)}, nf^{(12)})$
$Mid(nf^{(11)}, nf^{(13)})$
$Mid(nf^{(12)}, nf^{(13)})$
$nf^{(1)}$ $\Pi(nf^{(1)})$ r_1
$nf^{(2)}$ $\Pi(nf^{(2)})$ r_2
$nf^{(p)}$ $\Pi(nf^{(p)})$ r_p
$nf^{(p1)}$
$nf^{(p2)}$
$Mid(nf^{(p1)}, nf^{(p2)})$
Initial *species-list* at level i
Include endpoints in species-list
NO
Is the midpoint better than the center?
YES
Is the midpoint worse than the endpoints?
YES
Replace the center
NO
Check next midpoint

Fig. 8 Creation procedure in UEGO_med

same members of a pair may appear in the *species_list* several times. Nevertheless, the *Fuse_species* procedure described below will eliminate the possible duplicates. Every newly inserted species is assigned the current level value (i). Although each one of the pairs (and its corresponding midpoint) is evaluated independently of the remaining ones, the total number of functions evaluations per species for this process is bounded in advance (every species in the list has a fixed maximum number of function evaluations for the creation of new points, equal to $budget\_per\_species = new_i/\text{length}(species\_list)$). Hence, the process is always under control.

As a result of this procedure the species list will possibly contain several species with different levels (hence different radii).

The motivation behind this method is to create species that are on different 'hills' thereby ensuring that there is a valley between the new species. The parameter of this procedure (*creat_evals*) is an upper bound of the number of function evaluations. In terms of genetic algorithms, it could be thought that, in this procedure, a single parent (species) is used to generate offspring (new species), and all parents are involved in the procedure of generating offspring.

- *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 merged (see Fig. 9). 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).

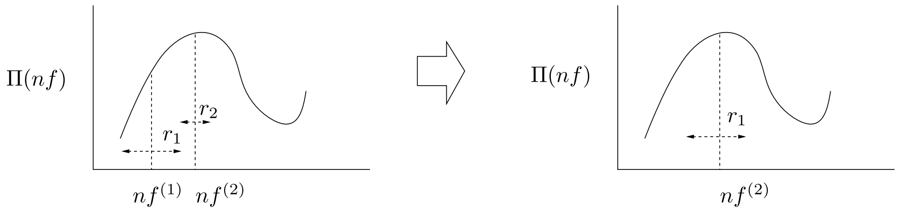


Fig. 9 Fusion procedure

- *Shorten_species_list*(*max_spec_num*): It deletes species to reduce the length of the list to the given value (*max_spec_num*). Higher level species are deleted first, therefore species with larger radii are always kept. For this reason 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.
- *Optimize_species*(*opt_evals*): It executes a local optimizer for every species using a given number of evaluations (*budget_per_species*) (see Fig. 7). At level i, *budget_per_species* is computed as n_i/*max_spec_num*, i.e., and it depends on the maximum population size.

Note that UEGO may terminate simply because it has executed all of its levels. The final number of function evaluations thus depends on the complexity of the objective function. This is qualitatively different from genetic algorithms, which typically run up to a limit on the number of function evaluations.

References

1. Benati, S., Laporte, G.: Tabu search algorithms for the $(r|X_p)$-medianoid and $(r|p)$-centroid problems. Location Sci. **2**(4), 193–204 (1994)
2. Bhadury, J., Eiselt, H.A., Jaramillo, J.H.: An alternating heuristic for medianoid and centroid problems in the plane. Comput. Oper. Res. **30**(4), 553–565 (2003)
3. Drezner, T.: Optimal continuous location of a retail facility, facility attractiveness, and market share: an interactive model. J. Retail. **70**(1), 49–64 (1994)
4. Drezner, T.: Location of multiple retail facilities with limited budget constraints in continuous space. J. Retail. Consum. Serv. **5**(3), 173–184 (1998)
5. Drezner, T., Drezner, Z.: Facility location in anticipation of future competition. Location Sci. **6**(1), 155–173 (1998)
6. Drezner, T., Drezner, Z.: Retail facility location under changing market conditions. IMA J. Manag. Math. **13**(4), 283–302 (2002)
7. Drezner, T., Drezner, Z.: Finding the optimal solution to the Huff based competitive location model. Comput. Manag. Sci. **1**(2), 193–208 (2004)
8. Drezner, Z.: Competitive location strategies for two facilities. Reg. Sci. Urban Econ. **12**(4), 485–493 (1982)
9. Drezner, Z.: Facility Location: A Survey of Applications and Methods. Springer, Berlin (1995)
10. Drezner, Z., Hamacher, H.W.: Facility Location. Applications and Theory. Springer, Berlin (2002)
11. Eiselt, H.A., Laporte, G.: Sequential location problems. Eur. J. Oper. Res. **96**(2), 217–231 (1997)
12. Eiselt, H.A., Laporte, G., Thisse, J.F.: Competitive location models: a framework and bibliography. Transp. Sci. **27**(1), 44–54 (1993)
13. Fernández, J., Pelegrín, B., Plastria, F., Tóth, B.: Solving a Huff-like competitive location and design model for profit maximization in the plane. Eur. J. Oper. Res. **179**(3), 1274–1287 (2007)
14. Francis, R.L., McGinnis, L.F., White, J.A.: Facility Layout and Location: An Analytical Approach. Prentice Hall, Englewood Cliffs (1992)

15. González-Linares, J.M., Guil, N., Zapata, E.L., Ortigosa, P.M., García, I.: Deformable shapes detection by stochastic optimization. In: 2000 IEEE International Conference on Image Processing (ICIP'2000). Vancouver, Canada, 2000
16. Hakimi, S.L.: On locating new facilities in a competitive environment. Eur. J. Oper. Res. **12**(1), 29–35 (1983)
17. Hodgson, M.J.: A location–allocation model maximizing consumers' welfare. Reg. Stud. **15**(6), 493–506 (1981)
18. Huff, D.L.: Defining and estimating a trading area. J. Mark. **28**(3), 34–38 (1964)
19. Kilkenny, M., Thisse, J.F.: Economics of location: a selective survey. Comput. Oper. Res. **26**(14), 1369–1394 (1999)
20. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
21. Love, R.F., Morris, J.G., Wesolowsky, G.O.: Facilities Location. Models and Methods. North-Holland, Amsterdam (1988)
22. Okabe, A., Suzuki, A.: Stability of spatial competition for a large number of firms on a bounded two-dimensional space. Environ. Plan. A **19**(8), 1067–1082 (1987)
23. Ortigosa, P.M., García, I., Jelasity, M.: Reliability and performance of UEGO, a clustering-based global optimizer. J. Glob. Optim. **19**(3), 265–289 (2001)
24. Plastria, F.: Static competitive facility location: an overview of optimisation approaches. Eur. J. Oper. Res. **129**(3), 461–470 (2001)
25. Plastria, F., Carrizosa, E.: Optimal location and design of a competitive facility. Math. Program. **100**(2), 247–265 (2004)
26. Redondo, J.L., Fernández, J., García, I., Ortigosa, P.M.: A robust and efficient global optimization algorithm for planar competitive location problems. Ann. Oper. Res. (2008, to appear). DOI: 10.1007/s10479-007-0233-x
27. Redondo, J.L., Ortigosa, P.M., García, I., Fernández, J.J.: Image registration in electron microscopy. A stochastic optimization approach. In: Proceedings of the International Conference on Image Analysis and Recognition, ICIAR 2004. Lecture Notes in Computer Science, vol. 3212(II), pp. 141–149 (2004)
28. Solis, F.J., Wets, R.J.B.: Minimization by random search techniques. Math. Oper. Res. **6**(1), 19–30 (1981)
29. Weber, A.: Uber den Standort der Industrien 1. Teil: Reine Theorie des Standortes. Tübingen, Niemeyer (1909)
30. Weiszfeld, E.: Sur le point pour lequel la somme des distances de *n* points donnés est minimum. Tohoku Math. J. **43**, 355–386 (1937)