Huff-like Stackelberg location problems on the plane

José Fernández, Juana L. Redondo, Pilar M. Ortigosa and Boglárka G.-Tóth

Abstract The so-called leader-follower (or Stackelberg) problem is researched. A chain, the leader, wants to locate a single new facility in a region of the plane. After that, as a reaction, the competitor chain, the follower, will locate a single new facility too, knowing the decision taken by the leader. Several variants of the problem are analyzed. In the simplest one, the objective of both the leader and the follower is to maximize the market share, the qualities of the facilities to be located are given beforehand, and the demand is fixed (no costs are considered). In the second one, the qualities of the facilities to be located are considered variables of the problem, and costs related both to location and quality are taken into account; the demand is fixed as in the first model. Finally, the last model extends the previous one considering that the demand varies depending on the location and the quality of the facilities. Exact (for the first problem) and heuristic (for the second and third problems) approaches proposed for the aforementioned location models are described and analyzed. High performance computing approaches for the heuristic methods are also reviewed. A new exact branch-and-bound method for the last two problems is also suggested.

José Fernández
Dpt. Statistics and Operations Research, University of Murcia, Campus de Espinardo, 30100 Espinardo, Murcia, Spain, e-mail: josefdez@um.es

Juana L. Redondo
Dpt. Informatics, University of Almería, ceiA3, Ctra. Sacramento s/n, La Cañada de San Urbano, 04120 Almería, Spain, e-mail: jlredondo@ual.es

Pilar M. Ortigosa
Dpt. Informatics, University of Almería, ceiA3, Ctra. Sacramento s/n, La Cañada de San Urbano, 04120 Almería, Spain, e-mail: ortigosa@ual.es

Boglárka G.-Tóth
Dpt. Differential Equations, Budapest University of Technology and Economics, Egry József u. 1., 1111 Budapest, Hungary, e-mail: bog@math.bme.hu

1 Introduction

Locating a new facility usually requires a massive investment. In order to guarantee the survival of the facility, especially in a competitive environment (where other facilities offering the same product or service exist), the locating firm tries to take all the factors which may affect the market share captured by the facility (or its profit) into account. A well-known aphorism states that 'the most important attributes of stores are location, location and location'. The literature about facility location corroborates that point as the number of papers devoted to that topic is huge. Mathematical location models try to combine all the factors of interest for the facility into neat equations which try to faithfully represent (a simplified version of) reality. The location decisions provided by the location models can be of invaluable help to the decision-maker, as the location of a facility cannot be easily altered.

Depending on the location space, competitive facility location models can be subdivided, as any other type of location problems, into three main categories: (i) continuous problems, where the set of feasible locations for the new facility (or facilities) is (a subset of) the plane; (ii) network problems, where any point in a network (on an edge or a vertex) is a possible location, and (iii) discrete problems, when the set of potential locations is reduced to a finite set of points. In this chapter we restrict ourselves to continuous models, as this is the main research field of the authors, but the interested reader can find many references on network and discrete competitive location models in literature, see for instance [3, 4, 16, 29, 30, 45] and references therein.

In competitive models there is a demand which has to be, or may be, served by the facilities. This demand is commonly assumed to be concentrated at a finite set of points, called *demand points* (also referred to as *customers*). In most of the research works it is assumed that the demand is *fixed*, regardless the conditions of the market (price, distance to the facilities,...). This implicitly assumes that goods are 'essential' to the customers. It is only recent that the case of 'inessential' goods has been addressed [28, 35]. In those models it is assumed that the demand *varies* depending on the location of the facilities.

The *attraction* of a customer towards a facility depends on both the location and the characteristics of the facility. Usually the characteristics are combined into a single figure which represents the *quality* of the facility. The closer the facility to the customer and the higher its quality, the higher the attraction of the customer towards the facility. Although there are many ways to model the attraction (see [34]), the formula quality divided by a function of the distance (already proposed in [22]) is the most popular in literature, and the one followed in this chapter.

The *patronizing behavior* of customers, which establishes how customers split their demand among the available facilities, is another key factor of the model. Two rules dominate literature. In the *deterministic rule* it is assumed that customers only buy at a single facility, the one to which they are attracted most [7, 33]. However, this hypothesis has not found much empirical support, except in areas where shopping opportunities are limited and transportation is difficult. On the contrary, in the *probabilistic rule* customers patronize all the facilities. However, the demand served at

each facility is not the same: it is proportional to the attraction. Hence, more attractive facilities capture more demand than less attractive facilities. The probabilistic rule was already suggested in [22] to estimate the market share captured by competing facilities, and first used in a location model in [8]. In that paper, as in most of the ones using the probabilistic rule, the quality of the facility to be located was fixed, given beforehand. It was in [18] when quality was first considered an additional variable to the problem to be determined. In fact, it was empirically proved that both the location and the quality of the facility to be located have to be found simultaneously, as the location influences the quality, and vice-versa. In general, the probabilistic rule has proved to approximate the market share captured by the facilities more accurately than other alternatives, and it will be the one used in the models in this chapter.

Another point to be taken into account is the possible *reaction* of the competitors. In most competitive location models it is assumed that the competition is *static*. This means that competitors are already present in the market, the locating chain knows their characteristics and no reaction to the location of the new facility (or facilities) is expected from them. However, there are situations where the competitors do react to the location of the new facilities. In those cases, it is very important to *foresee* those reactions, as the market share and profit obtained by the locating chain may vary substantially. Although there are *dynamic* location models, where competitors can change their decisions indefinitely, and then the existence of *equilibrium* situations is of major concern (see for instance [6, 27, 19]), in this chapter the focus is on the so-called 'leader-follower' (or Stackelberg) problems. The scenario considered in that type of problems is that of a duopoly. A chain, the *leader*, makes the first movement, and locates p new facilities in the market, where similar facilities of a competitor (the *follower*), and possibly of its own chain, already exist. Then, the follower, as a reaction, decides to locate r new facilities. Hakimi [20] seems to be the first considering this type of two-level optimization problems. He introduced the term $(r|X_p)$ *medianoid* to refer to the follower's problem of locating r facilities in the presence of the p new leader's facilities located at the set of points X_p. And the term $(r|p)$ *centroid* problem to refer to the leader's problem of locating p new facilities, knowing that the follower, as a reaction, will locate r new facilities by solving the corresponding $(r|X_p)$ medianoid problem. In this chapter only the $(1|1)$ centroid problem will be considered, i.e., it is assumed that the leader will locate only one new facility, and the follower's reaction consists of the location of a new single facility too.

Even in this simple case the leader-follower problem is very hard to solve. In fact, the follower's problem is already a highly nonlinear global optimization problem (see [8, 18]). The literature on leader-follower location problems is scarce (see [15] for a review on the topic until 1996). And this shortage is even more pronounced in the case of continuous problems, largely due to the complexity of this type of bilevel programming problems. Drezner [14] solved the $(1|1)$ centroid problem for the Hotelling model and Euclidean distances exactly, through a geometric-based approach. Bhadury *et al.* [2] considered the $(r|p)$ centroid problem also for the Hotelling model with Euclidean distances, and gave an alternating heuristic to

cope with it. In [10] Drezner and Drezner considered the Huff model, and proposed three heuristic approaches for handling the $(1|1)$ centroid problem (see also [11]).

More recently, the authors of this chapter have worked and extended the Huff-like Stackelberg problems. In [44] an exact branch-and-bound method is proposed for a model closely related to that in [10]. This model was later extended in [37] to consider the quality of the new facilities as additional variables of the model, and also changing the objective from market share maximization to profit maximization; both sequential and parallel heuristics were proposed to cope with it (see [37, 41]). Finally, in [36], the model was extended to take into account the possibility of the variability of the demand (see also [1]); again, sequential and parallel heuristic procedures were proposed. The goal of this chapter is to make a critical review of those papers and to point lines for future research. First, in the next section, the basic notation is introduced, and then, in the three following sections, the three aforementioned models are reviewed. Finally, in the last section we point out an idea which may be used to develop exact methods for the last two models.

Although here we only consider that, as a reaction, the follower will locate an additional facility too, other alternatives have been recently proposed in literature. They all consider that the follower can change the quality of its existing facilities. In particular, in [43], the leader locates one single facility in a region of the plane, and then the follower may *increase* the quality of some of its facilities. The follower does not locate any new facility. In [26] the leader enters the market by locating several facilities at some of the points of a finite set of feasible locations (discrete problem), and then, the reaction of the competitor is to adjust (i.e., increase or decrease) the attractiveness of its existing facilities so as to maximize its own profit. However, it cannot open new facilities and/or close existing ones, either. The model is extended in [25], where the follower can also open new facilities or close some existing ones. The probabilistic rule is used in the three aforementioned papers. A different approach is followed in [13] (see also [12]) where a discrete location model based on the concept of coverage is presented. Each facility attracts consumers within a *sphere of influence* defined by a radius. The leader and the follower, each has a budget to be spent on the expansion of their chains either by *improving* their existing facilities or constructing new ones.

2 Notation

A chain, the *leader*, wants to locate a new single facility in a given area of the plane, where m facilities offering the same goods or product already exist. The first k (≥ 0) of those m facilities belong to the chain, and the other $m-k$ (>0) to a competitor chain, the *follower*. The leader knows that the follower, as a reaction, will subsequently position a new facility too.

The following notation will be used throughout this chapter:

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.

l index for the new facilities, $l = 1$ for the leader, $l = 2$ for the follower.

Variables

$z_l = (x_l, y_l)$ location of the new leader's ($l = 1$) or follower's ($l = 2$) facility.

α_l quality of the new leader's ($l = 1$) or follower's ($l = 2$) facility (in case the quality is to be determined by the model).

$nf_l = (z_l, \alpha_l)$ variables of the new leader's ($l = 1$) or follower's ($l = 2$) facility.

Input data

p_i location of the i-th demand point.

$\widehat{w}_i$ fixed demand (or purchasing power) at p_i, $\widehat{w}_i > 0$ (when the demand is assumed to be fixed).

$w_i^{\min}$ minimum possible demand at p_i, $w_i^{\min} > 0$ (when the demand is assumed to be variable).

$w_i^{\max}$ maximum possible demand at p_i, $w_i^{\max} \geq w_i^{\min}$ (when the demand is assumed to be variable).

f_j location of the j-th existing facility.

d_{ij} distance between p_i and f_j, $d_{ij} > 0$.

β_j quality of f_j, $\beta_j > 0$.

γ_i weight for the quality of (both existing and new) facilities as perceived by demand point p_i, $\gamma_i > 0$.

$d_i^{\min}$ minimum distance from p_i at which the new facilities can be located, $d_i^{\min} > 0$.

S_l location space where the leader ($l = 1$) or the follower ($l = 2$) will locate its new facility.

$\alpha_l^{\min}$ minimum level of quality for the new leader's ($l = 1$) or follower's ($l = 2$) facility, $\alpha_l^{\min} > 0$ (when the quality is a variable of the model).

$\alpha_l^{\max}$ maximum level of quality for the new leader's ($l = 1$) or follower's ($l = 2$) facility, $\alpha_l^{\max} \geq \alpha_l^{\min}$, (when the quality is a variable of the model).

Miscellaneous

$g_i(\cdot)$ a non-negative, non-decreasing function, which modulates the decrease in attractiveness as a function of distance.

$d_i(z_l)$ distance between p_i and $z_l, l = 1, 2$.

u_{i,nf_l} attraction that p_i feels for $nf_l, l = 1, 2$, $u_{i,nf_l} = \gamma_i \alpha_l / g_i(d_i(z_l))$.

$U_i(nf_1, nf_2)$ total utility perceived by a customer at p_i provided by all the facilities.

$w_i(U_i(nf_1, nf_2))$ actual demand at p_i (when the demand is assumed to be variable).

Computed parameters

u_{ij} attraction that p_i feels for f_j (or utility of f_j perceived by the people at p_i), $u_{ij} = \gamma_i \beta_j / g_i(d_{ij})$.

Market share and profit functions

$M_l(nf_1, nf_2)$ market share obtained by the leader ($l = 1$) or the follower ($l = 2$) after the location of the new facilities.

$\Pi_l(nf_1, nf_2)$ profit obtained by the leader ($l = 1$) or the follower ($l = 2$) after the location of the new facilities.

The profit functions Π_1 and Π_2 vary in each of the problems analyzed, and are detailed in the corresponding sections.

In all the models in this chapter it is assumed 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. 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

$$M_1(nf_1, nf_2) = \sum_{i=1}^{n} w_i \frac{u_{i,nf_1} + \sum_{j=1}^{k} u_{i,j}}{u_{i,nf_1} + u_{i,nf_2} + \sum_{j=1}^{m} u_{i,j}}, \tag{1}$$

where w_i stands for $\widehat{w}_i$ when the demand is fixed, and for $w_i(U_i(nf_1, nf_2))$ when the demand is variable. Analogously, the market share attracted by the follower's chain is

$$M_2(nf_1, nf_2) = \sum_{i=1}^{n} w_i \frac{u_{i,nf_2} + \sum_{j=k+1}^{n} u_{i,j}}{u_{i,nf_1} + u_{i,nf_2} + \sum_{j=1}^{m} u_{i,j}}. \tag{2}$$

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

$$(FP(nf_1)) \quad \begin{cases} \max\ \Pi_2(nf_1, nf_2) \\ \text{s.t.} \quad z_2 \in S_2 \\ \qquad d_i(z_2) \geq d_i^{\min}, i = 1, \ldots, n \\ \qquad \alpha_2 \in [\alpha_2^{\min}, \alpha_2^{\max}] \end{cases} \tag{3}$$

whose objective is the maximization of the profit obtained by the follower (once the leader has set up its new facility at nf_1). In case the problem $(FP(nf_1))$ has multiple optimal solutions, then it is assumed that the follower selects an optimal solution which provides the worst possible objective function value for the leader (the so-called *pessimistic approach* in bilevel programming [5]).

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

$$(LP) \begin{cases} \max\ \Pi_1(nf_1, nf_2^*(nf_1)) \\ \text{s.t.} \quad z_1 \in S_1 \\ \qquad d_i(z_1) \geq d_i^{\min}, i = 1, \ldots, n \\ \qquad \alpha_1 \in [\alpha_1^{\min}, \alpha_1^{\max}] \end{cases} \tag{4}$$

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 Π_1 at a given point nf_1, we have to first solve the corresponding medianoid problem $(FP(nf_1))$ to obtain $nf_2^*(nf_1)$.

3 A model without costs

3.1 The model

The first model we will describe is that in [44]. Essential goods are considered. Therefore, the demand has to be served by the facilities. The demand quantities are assumed to be known and fixed. Also the quality values of the new facilities to be located, α_1 and α_2, are assumed to be given, i.e., *they are not variables of the model*. As the qualities are fixed, no cost related to the achievement of a given level of quality is considered. No cost related to the setting-up of the facilities at a given location is considered either. Then, taking into account that the profit obtained by a player is an increasing function of the market share it captures, the objective functions considered in [44] were

$$\Pi_l(nf_1, nf_2) = M_l(nf_1, nf_2),\ l = 1, 2.$$

In addition to this, no weights for the quality of facilities as perceived by demand points are used (i.e., it is assumed that $\gamma_i = 1, i = 1, \ldots, n$), and the location space is the same for the leader and the follower, i.e., $S_1 = S_2$. No other constraints are considered in the model. The corrected Euclidean distance [9] was used as distance function.

Since the demand is fixed and has to be served, then

$$M_1(nf_1, nf_2) + M_2(nf_1, nf_2) = \sum_{i=1}^{n} \widehat{w}_i. \tag{5}$$

In particular, what is a gain for one chain is a loss for the other. This zero-sum concept is the key used in [44] to develop a Branch-and-Bound (B&B) procedure to solve the leader problem rigorously, to have a guarantee on the reached accuracy.

3.2 A B&B algorithm for the follower problem

Branch-and-bound (B&B) algorithms recursively decompose the original problem into smaller disjoint subproblems until the solution is found. The method avoids visiting those subproblems which are known not to contain a solution. The initial set $C_1 = S_1 (= S_2)$ is subsequently partitioned in more and more refined subsets (branching). At every iteration, the method has a list Λ of subsets C_k of C_1. The method stops when the list is empty. For every subset C_k in Λ, upper bounds UB^k of the objective function on C_k are determined. Moreover, a global lower bound *GLB* is updated. If $UB^k < GLB$ for a given subset C_k, it can be removed from the list, since it cannot contain a maximum.

The steps of the method can be seen in Algorithm 1. In the solution procedure for the leader problem, a similar problem to that of the follower, in which the leader

wants to locate a new facility at nf_1, given the location and the quality of all the facilities of the competitor (the follower), has to be solved. In this case, the leader has to solve a medianoid problem in which the roles of leader and follower are interchanged. We will call this problem a *reverse medianoid problem*. To take both the medianoid and the reverse medianoid problems into account, in Algorithm 1 the new facility of the competitor is denoted by nf, the objective function by $M(nfa)$ (where $M(nfa) = M_2(nf, nfa)$ when solving a medianoid problem and $M(nfa) = M_1(nfa, nf)$ when solving a reverse medianoid problem), and the feasible set by C.

Algorithm 1: B&B algorithm for the *(reverse) follower problem*: Function $FunctB\&B(M, nf, C, \varepsilon_f)$

1: $\Lambda := \emptyset$.
2: $C_1 := C$.
3: Determine an upper bound UB^1 on C_1.
4: Compute nfa^1 :=midpoint(C_1), $BestPoint := nfa^1$.
5: Determine lower bound: $LB^1 := M(nfa^1)$, $GLB := LB^1$.
6: Put C_1 on list Λ, $r := 1$.
7: **while** $\Lambda \neq \emptyset$ **do**
8: Take subset C from list Λ and bisect into C_{r+1} and C_{r+2}.
9: **for** $t := r+1$ to $r+2$ **do**
10: Determine upper bound UB^t.
11: **if** $UB^t > GLB + \varepsilon_f$ **then**
12: Compute nfa^t :=midpoint(C_t) and $LB^t := M(nfa^t)$.
13: **if** $LB^t > GLB$ **then**
14: $GLB := LB^t$, $BestPoint := nfa^t$ and remove all C_i from Λ with $UB^i < GLB$.
15: **if** $UB^t > GLB + \varepsilon_f$ **then**
16: save C_t in Λ.
17: $r := r+2$.
18: OUTPUT: $\{BestPoint, GLB\}$.

The B&B method introduced in [44] uses boxes (2-dimensional intervals) as subsets of the initial region and the subdivision rule bisects a box C over its longest edge. Several selection rules of the next box to be selected (Step 8 of Algorithm 1) were tested in [44], see Section 3.4.

Concerning the computation of bounds, the global lower bound is updated by evaluating the objective function at some points (the centers of the boxes). As for the upper bounds, four variants were proposed in [44]. The simplest one (which turned out to be competitive with the other three more elaborated bounds based on D.C. decompositions of the objective function) is based on the underestimation of the distance from demand point p_i to facilities in a box C. Since the new facility is only located at one point within the box, we obtain an overestimation (upper bound) of the market captured by the new facility. The idea developed in [44] is similar to that in [32].

The demand points p_i within box C have a distance $\Delta_i(C) = 0$ from C. For demand points out of box C, $p_i \notin C$, the shortest distance $\Delta_i(C)$ of p_i to the box is calculated, $\Delta_i(C) = \min_{x \in C} d(x, p_i)$. The distance $\Delta_i(C)$ can be determined as fol-

lows. Box C is defined by two points: lower-left point $LL = (ll_1, ll_2)$ and upper-right point $UR = (ur_1, ur_2)$. The shortest distance from demand point p_i to the box C can be computed by

$$\Delta_i(C) = \begin{cases} 0 & \text{if } p_i \in C \\ \sqrt{\Delta_{i1}^2 + \Delta_{i2}^2} & \text{if } p_i \notin C \end{cases}$$

where

$$\begin{aligned} \Delta_{i1} &= \max\{ll_1 - p_{i1}, p_{i1} - ur_1, 0\} \\ \Delta_{i2} &= \max\{ll_2 - p_{i2}, p_{i2} - ur_2, 0\} \end{aligned}$$

Notice that this distance calculation can be extended to higher dimensions.

The output of Algorithm 1 is the best point found during the process and its corresponding function value. The best point is guaranteed to differ less than ε_f in function value from the optimal solution of the problem.

Another B&B algorithm which can be used to solve the follower problem is described in [18]. It uses interval analysis tools (see [47]) and can also handle the follower problems in the next two sections.

3.3 A B&B algorithm for the leader problem

The corresponding B&B method for the leader problem is given in pseudocode form in Algorithm 2. The branching and selection rules used were the same as in Algorithm 1, as well as the computation of the global lower bound.

The key point in the algorithm is computation of the upper bounds. Let $C \subseteq \mathbb{R}^2$ denote a subset of the search region of the leader problem (LP). An upper bound of the objective function $M_1(nf_1, nf_2^*(nf_1))$ over C can be obtained by having the leader solve the reverse medianoid problem, as the following lemma proves.

Lemma 1. *Let nf_2 be a given solution for the new follower's facility. Then*

$$UB(C, nf_2) = \max_{nf_1 \in C} M_1(nf_1, nf_2)$$

is an upper bound of $M_1(nf_1, nf_2^*(nf_1))$ over C.

Proof. According to (5), maximizing the market share captured by the follower given nf_1 is equivalent to finding the facility nf_2 that minimizes the market share captured by the leader. Hence, $M_1(nf_1, nf_2^*(nf_1)) \leq M_1(nf_1, nf_2)$ such that

$$\max_{nf_1 \in C} M_1(nf_1, nf_2^*(nf_1)) \leq \max_{nf_1 \in C} M_1(nf_1, nf_2) = UB(C, nf_2). \ \square$$

For a given box C_t, the choice of nf_2^t for the upper bound calculation is done as follows. First, the midpoint of C_t is computed, and considering it as the new leader's facility, nf_1^t, the corresponding follower's problem is solved, $(FP(nf_1^t))$, obtaining nf_2^t. Then, the upper bound is obtained by solving the reverse medianoid problem up to an accuracy ε_l

Algorithm 2: B&B algorithm for the *leader problem*

1: $\Lambda := \emptyset$.
2: $C_1 := S$.
3: Compute nf_1^1 :=midpoint(C_1), $BestPoint := nf_1^1$.
4: Solve the problem for the follower: $\{nf_2^1, lbobj\} := FunctB\&B(M_2, nf_1^1, C_1, \varepsilon_f)$.
5: Determine an upper bound UB^1 on C_1 solving a reverse medianoid problem:
$\{nfa, UB^1\} := FunctB\&B(M_1, nf_2^1, C_1, \varepsilon_l)$.
6: Determine lower bound: $LB^1 := M_1(nf_1^1, nf_2^1)$, $GLB := LB^1$.
7: Put C_1 on list Λ, $r := 1$.
8: **while** $\Lambda \neq \emptyset$ **do**
9: Take subset C from list Λ and bisect into C_{r+1} and C_{r+2}.
10: **for** $t := r+1$ to $r+2$ **do**
11: Compute nf_1^t =midpoint(C_t).
12: Solve the problem for the follower: $\{nf_2^t, lbobj\} := FunctB\&B(M_2, nf_1^t, C_1, \varepsilon_f)$.
13: Determine upper bound UB^t solving a reverse medianoid problem:
$\{nfa, UB^t\} := FunctB\&B(M_1, nf_2^t, C_t, \varepsilon_l)$
14: **if** $UB^t > GLB + \varepsilon_l$ **then**
15: Determine $LB^t := M_1(nf_1^t, nf_2^t)$.
16: **if** $LB^t > GLB$ **then**
17: $GLB := LB^t$, $BestPoint := nf_1^t$, and remove all C_i from Λ with $UB^i < GLB$.
18: **if** $UB^t > GLB + \varepsilon_l$ **then**
19: save C_t in Λ.
20: $r := r+2$.
21: OUTPUT: $\{BestPoint, GLB\}$.

$$UB^t = UB(C_t, nf_2^t) = \max_{nf_1 \in C_t} \{M_1(nf_1, nf_2^t)\} = FunctB\&B(M_1, nf_2^t, C_t, \varepsilon_l).$$

Again, the output of the B&B method (see Algorithm 2) is the best point found during the process and its corresponding function value, which differs less than ε_l from the optimum value of the problem.

3.4 Computational studies

A random problem with $n = 10$ demand points and $m = 4$ existing facilities was first solved to illustrate the algorithm. The number k of facilities belonging to the leader's chain was varied from $k = 0$ to 4. The other parameters of the problem were chosen from uniform distributions (see [44]). Table 1 shows the resulting optimal locations and market capture of both chains. In the last line, the gain or loss for the leader, to be understood as the difference between the market captured by the leader after and before the location of the facilities, is given. The accuracy for algorithms 1 and 2 were set both to $\varepsilon_l = \varepsilon_f = 10^{-2}$.

One can observe a characteristic of the problem, where leader and follower tend to *co-locate* when the number of existing facilities of the leader is low. Notice also that when the leader is dominant in the market then the leader suffers a decrease

Table 1 Optimal locations and market capture for different number of leader facilities, $k = 0, \ldots, 4$; locations and market captures are rounded to two decimals.

		$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
Optima location	Leader	$\begin{pmatrix}2.44\\3.97\end{pmatrix}$	$\begin{pmatrix}5.03\\0.69\end{pmatrix}$	$\begin{pmatrix}5.33\\4.34\end{pmatrix}$	$\begin{pmatrix}5.33\\4.34\end{pmatrix}$	$\begin{pmatrix}5.03\\0.69\end{pmatrix}$
	Follower	$\begin{pmatrix}2.44\\3.97\end{pmatrix}$	$\begin{pmatrix}5.03\\0.69\end{pmatrix}$	$\begin{pmatrix}1.41\\4.65\end{pmatrix}$	$\begin{pmatrix}1.75\\3.79\end{pmatrix}$	$\begin{pmatrix}1.75\\3.79\end{pmatrix}$
Market Capture	Leader	186.29	367.87	497.70	611.07	773.44
	Follower	813.71	632.13	502.30	388.93	226.56
Gain or loss for the leader		186.29	100.67	14.17	-72.46	-226.56

in market share after the location of the two new facilities (see the negative values in the last line of Table 1). This is because in those cases the follower increases its market share more than the leader.

Concerning the efficiency of the selection rule of the next box to be processed, breadth-first and best-bound strategies were researched. The results in [44] concluded that best-bound strategy is the one providing the best results, as in average, the number of iterations employed by Algorithm 1 was reduced significantly. The influence in the number of iterations of Algorithm 2 was not so clear when using the upper bound described in Section 3.2, but when additional bounds are employed the best-bound selection rule was also clearly the best for Algorithm 2.

As for the memory requirement, it is known that branch-and-bound algorithms are usually hindered by huge search trees that need to be stored in memory. This complexity usually increases rapidly with dimension and with accuracy. Interestingly, this does not seem to be the case for this problem. There are never more than 30 boxes in the storage tree. And the same remains valid when the accuracy is increased up to 0.0001 for both algorithms 1 and 2.

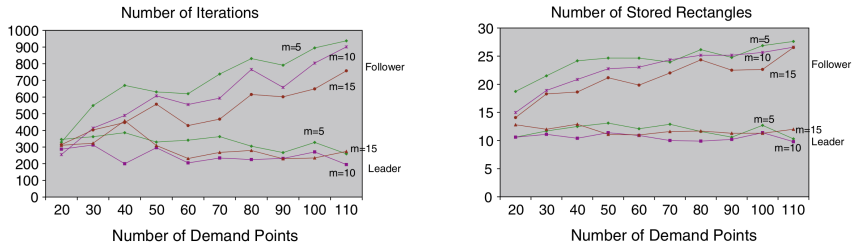

Fig. 1 Average number of iterations and memory requirement (rectangles) over ten random cases varying number of demand points $n = 20, \ldots, 110$, existing facilities $m = 5, 10, 15$ and $k = [m/2]$. $\varepsilon_l = \varepsilon_f = 0.01$

The last set of experiments done in [44] studied whether larger problems could be solved in reasonable time. To this aim, random problems were generated varying the number of demand points ($n = 20, 30, \ldots, 110$), number of existing facilities

($m = 5, 10, 15$) and number of those facilities belonging to the leader's chain ($k = [m/2]$). For each (n,m) setting, ten problems were generated by randomly selecting the parameters of the problem from uniform distributions. The results can be seen in Figure 1. It can be seen that increasing the number of demand points does not make the problem more complex in terms of the memory requirement. The leader problem neither needs more iterations, although the follower problem needs more iterations on average. Hence, the results suggest that no exponential effort is required to solve the problems with increasing number of demand points, confirming the viability of the approach.

4 A model with costs assuming fixed demand

4.1 The model

The scenario considered in this section (see [37]) is similar to the one previously described. The demand is again supposed to be fixed and known. But now, both the location and the quality (design) of the new facilities have to be found and several types of costs are considered.

The objective function Π_2 for the follower problem (see Eq. 3), is now formulated as the difference between the revenues obtained from the captured market share minus the operating costs of the new facility:

$$\Pi_2(nf_1, nf_2) = F_2(M_2(nf_1, nf_2)) - G_2(nf_2). \tag{6}$$

Similarly, the profit obtained by the leader (see Eq. 4) is given by:

$$\Pi_1(nf_1, nf_2^*(nf_1)) = F_1(M_1(nf_1, nf_2^*(nf_1))) - G_1(nf_1). \tag{7}$$

Functions F_l, $l = 1, 2$, are strictly increasing differentiable functions that transform the market share into expected sales. In the computational studies in [37], they are linear, $F_l(M_l) = c_l \cdot M_l$, where c_l is the income per unit of goods sold.

Functions $G_l, l = 1, 2$, are the operating costs functions. G_l should increase as z_l gets closer to any demand point, since it is rather likely the operating costs of the facility will be higher as the facility approaches the demand points. Furthermore, G_l should be a nondecreasing and convex function in the variable α_l, since the more quality the facility requires, the higher the costs will be, at an increasing rate. In [37] it is assumed that functions G_l consist of the sum of the location costs and the costs needed to achieve a given level of quality, i.e. $G_l(nf_l) = G_l^a(z_l) + G_l^b(\alpha_l)$. In the computational experiments the following choices were made: $G_l^a(z_l) = \sum_{i=1}^n \Phi_l^i(d_i(z_l))$, with $\Phi_l^i(d_i(z_l)) = \widehat{w}_i/((d_i(z_l))^{\phi_l^{i0}} + \phi_l^{i1})$, $\phi_l^{i0}, \phi_l^{i1} > 0$ and $G_l^b(\alpha_l) = \exp(\alpha_l/\xi_l^0 + \xi_l^1) - \exp(\xi_l^1)$, with $\xi_l^0 > 0$ and $\xi_l^1 \in \mathbb{R}$ given values. See [18] for a detailed explanation of these functions, as well as other possible expressions for F_l and $G_l(nf_l)$.

Notice that the key to solving the problem of the previous section with precision was that *what is a gain for one chain is a loss for the other*, see (5). This is no longer true for this model: notice that now $\Pi_1(nf_1, nf_2) + \Pi_2(nf_1, nf_2)$ is not necessarily constant due to the cost functions. This fact impedes using the methodology employed in the previous section to develop a B&B method for the new leader's problem (Lemma 1 does not hold any more). That is why heuristic procedures are proposed in [37] to cope with the new problem. However, other strategies are possible, as described in Section 6.

4.2 Solving the medianoid problem

The algorithm UEGO is used here to deal with the medianoid problem. UEGO, which stands for Universal Evolutionary Global Optimizer, is a memetic multimodal global optimization method especially suitable to be parallelized and highly adaptable to different problems [24, 31, 38, 39, 40, 42].

The key concept of UEGO is that of species, which is defined by a center and a radius. The center is a solution, and the radius is a positive number that defines an attraction area and hence, multiple solutions. In particular, for the medianoid problem, a species is an array of the form $(nf_2, \Pi_2(nf_1, nf_2), R)$ (we also store information about the objective value at the center of the species). During the optimization procedure, UEGO works with a set of species stored in the *species_list*.

The adaptability of UEGO mainly relies on being defined in two levels, global an local. In the global level, UEGO defines an iterative and progressively cooled management process over a set of available species, and this process is the same for all the problems to which UEGO is applied. In the local one, a particular local optimizer is selected for the studied problem at the context defined by every species. For the current problem, a Weiszfeld-like method (WLM) has been considered as a local optimizer. The UEGO algorithm executed with WLM to solve the medianoid problem will be called UEGO_med throughout.

A global description of UEGO_med is given in Algorithm 3. The input given parameter nf_1 indicates the additional leader facility, which has to be taken into account apart from the m pre-existing facilities. Additionally, UEGO_med has four more user given parameters: (i) N, the maximum number of function evaluations (f.e.) allowed for the entire optimization process; (ii) L, the maximum number of levels (iterations) of the algorithm; (iii) M, which refers to the maximum length of the *species_list*, and (iv) R_L, which indicates the minimum radius that a species can have. Furthermore, from these four input parameters, three important values are computed at each level i: the maximum number of f.e. for the creation of new species (new_i), the maximum number of f.e. for the optimization of species (n_i), and the radius assigned to the new species (R_i). The equations linking all these parameters are detailed in [23, 31].

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

Algorithm 3: Algorithm UEGO_med(nf_1, N, L, M, R_L)

```
1: Init_species_list
2: Optimize_species(n_1)
3: for i = 2 to L do
4:    Determine R_i, new_i, n_i
5:    Create_species(new_i)          {# budget_per_species = new_i/length(species_list_i)}
6:    Fuse_species(R_i)
7:    Shorten_species_list(M)
8:    Optimize_species(n_i)          { # budget_per_species = n_i/M}
9:    Fuse_species(R_i)
```

- *Init_species_list:* The initial *species_list* is composed of a single species. The value of nf_2 is randomly computed and the corresponding radius is set to R_1.
- *Create_species(create_evals)*: In terms of evolutionary computation, this procedure can be interpreted as an algorithm to create offspring. The input parameter *create_evals* indicates the number of function evaluations allowed for the creation procedure at the current level. The most remarkable aspect of this mechanism is that every species in the *species_list* is able to generate a new progeny without participation from the remaining ones. The parameter *create_evals* is internally divided by the current number of existing species ($length(species\_list_i)$), which means that the budget available per species for the creation of new points is equal to:

$$budget\_per\_species = new_i/length(species\_list_i).$$

  For each single species, the creation method proceeds as follows: New random exploratory points are created within the area defined by its radius, and for every pair of those points, a new candidate solution is created at the middle of the *segment* connecting the pair. Then, all the candidate points are evaluated, and the one with the best objective function value replaces the center of the original species in the case that it improves the objective function of the center. Later, the merit of the extreme points to become a new species, is analyzed. Both extreme points are inserted into the *species_list* if their objective function values are better than the one at the corresponding midpoint. Every new inserted species is assigned the current radius value (R_i).
- *Fuse_species(radius)*: This procedure unites species from the *species_list* that are closer than the distance defined by the parameter *radius*. Then, for every pair of species in the list, the Euclidean distance is computed. If such a distance is smaller than the given radius, the species with the lowest fitness are removed. The radius of the species that remains is set equal to the maximum of the radii of the original two species.
- *Shorten_species_list (max_list_length)*: It deletes species to reduce the list length to *max_list_length* value. The species with the smaller radius are deleted first.
- *Optimize_species(opt_evals)*: In this procedure, every species calls a local optimizer once, using the nf_2 value of the caller species as initial point. If after

the execution of the local method a new point with a better objective function is found, then the original nf_2 is updated. The budget per species for the optimization process, in terms of number of function evaluations, is n_i/M. For the problem at hand, a Weiszfeld-like algorithm has been considered as local optimizer.

4.2.1 Weiszfeld-like algorithm WLM

This algorithm is a steepest descent method. The derivatives of the objective function are equated to zero and the next iterate is obtained by implicitly solving these equations. Notice that, here, the derivatives are computed taking the F_l and G_l functions described in subsection 4.1 into account. Of course, they should be recomputed if any other expression is considered.

If we denote

$$r_i = \sum_{j=1}^{m} u_{ij}, t_i = \widehat{w}_i \sum_{j=k+1}^{m} u_{ij},$$

$$H_i(nf_2) = \frac{\partial \Pi_2}{\partial d_i(z_2)} = -\frac{\mathrm{d}F_2}{\mathrm{d}M_2} \cdot \frac{\alpha_2 \gamma_i t_i g_i'(d_i(z_2))}{(\gamma_i \alpha_2 + r_i g_i(d_i(z_2)))^2} - \frac{\mathrm{d}\Phi^i}{\mathrm{d}d_i(z_2)},$$

and $d_i(z_2)$ is a distance function such that

$$\frac{\partial d_i(z_2)}{\partial x_2} = x_2 A_{i1}(z_2) - B_{i1}(z_2), \quad \frac{\partial d_i(z_2)}{\partial y_2} = y_2 A_{i2}(z_2) - B_{i2}(z_2), \tag{8}$$

then the Weiszfeld-like algorithm for solving the corresponding problem is described by Algorithm 4 (for more details see [18]).

Algorithm 4: WLM (Weiszfeld-like algorithm)

1: Set iteration counter $ic = 0$
2: Initialize $nf_2^{(0)} = (x_2^{(0)}, y_2^{(0)}, \alpha_2^{(0)})$
3: **while** stopping criteria are not met **do**
4: Update $nf_2^{(ic+1)} = (x_2^{(ic+1)}, y_2^{(ic+1)}, \alpha_2^{(ic+1)})$
5: **if** $nf_2^{(ic+1)}$ is unfeasible **then**
6: $nf_2^{(ic+1)} \in [nf_2^{(ic)}, nf_2^{(ic+1)}] \cap \partial S_2$
7: $ic = ic + 1$

Values of $x_2^{(ic+1)}$ and $y_2^{(ic+1)}$ in Algorithm 4 are obtained as:

$$x_2^{(ic+1)} = \frac{\sum_{i=1}^{n} H_i(nf_2^{(ic)}) B_{i1}(z_2^{(ic)})}{\sum_{i=1}^{n} H_i(nf_2^{(ic)}) A_{i1}(z_2^{(ic)})}, \qquad y_2^{(ic+1)} = \frac{\sum_{i=1}^{n} H_i(nf_2^{(ic)}) B_{i2}(z_2^{(ic)})}{\sum_{i=1}^{n} H_i(nf_2^{(ic)}) A_{i2}(z_2^{(ic)})}$$

and $\alpha_2^{(ic+1)}$ as a solution of the equation:

$$\frac{\mathrm{d}F_2}{\mathrm{d}M_2} \cdot \sum_{i=1}^{n} \frac{\gamma_i t_i g_i(d_i(z_2^{(ic+1)})))}{(\gamma_i \alpha_2 + r_i g_i(d_i(z_2^{(ic+1)}))^2} - \frac{\mathrm{d}G_2}{\mathrm{d}\alpha_2} = 0.$$

Two stopping rules are applied in WLM: (i) the algorithm stops if

$$\|(x_2^{(ic-1)}, y_2^{(ic-1)}) - (x_2^{(ic)}, y_2^{(ic)})\|_2 < \varepsilon_1 \text{ and } |\alpha_2^{(ic-1)} - \alpha_2^{(ic)}| < \varepsilon_2,$$

for given tolerances $\varepsilon_1, \varepsilon_2 > 0$; and (ii) the procedure finishes if a maximum number of iterations $ic_{\max}$ is achieved or the number of function evaluations exceeds the budget assigned.

In Step 6 of Algorithm 4, $nf_2^{(ic+1)}$ is set to a point in the segment $[nf_2^{(ic)}, nf_2^{(ic+1)}]$ which is also on the border ∂S_2 of the feasible region S_2.

The l_{2b} distance, given by

$$d_i(z_l) = \sqrt{b_1(x_l - p_{i1})^2 + b_2(y_l - p_{i2})^2},$$

satisfies the conditions in (8). Furthermore, it has proved to be a good distance predicting function (see [17]), and it is therefore a good distance function to be used in competitive location models, as it measures distances (or travel time) as they are perceived by customers on their ways to and from facilities.

4.3 Solving the centroid problem

Four heuristics are introduced in [37] for handling the centroid problem, namely, a grid search procedure (GS), an alternating method called AlternatMed and two evolutionary algorithms based on the UEGO_med structure. These two variants, which differ basically in the considered local optimizer, are named UEGO_cent.WLM and UEGO_cent.SASS.

A comprehensive computational study in [37] shows that UEGO_cent.SASS is the algorithm which provides the best results. In fact, in all the considered problems, it is the algorithm giving the best solutions. In view of those results, only the algorithm UEGO_cent.SASS is explained below. For the sake of brevity, only the fundamental differences concerning UEGO_med are mentioned. The interested reader can always consult [37] for a detailed account of the remaining methods.

Species definition: A species is now defined by the vector (nf_1, nf_2, R), where nf_1 refers to the leader point, nf_2 is the solution obtained by UEGO_med when taking the original m existing facilities and nf_1 into account, and R is the radius of the species.

Create_species procedure: This procedure is, in essence, the same as the creation process described in subsection 4.2. However, some amendments have been made to comply with certain computational requirements.

In this procedure, random trial points for nf_1 are also created within the area defined by the radius of the species. Additionally, similar to what is done in UEGO_med, the midpoint of each pair of solutions is also computed. However, not all candidate solutions are evaluated, but only the most promising ones, i.e., we do not solve the corresponding medianoid problem associated to each new point to obtain the follower's facility. This is done in this way because this procedure is too costly and the number of points to be evaluated is very high. On the contrary, we first analyze the merit of the candidate solutions by computing an approximate objective value. More precisely, the follower's facility associated to the species from which they were generated is used to obtain an approximate fitness for the leader's candidate solutions.

After this process, for every species in the *species_list* we have a sublist of 'candidate' points to generate new species. Notice that in this creation process, the candidate solutions never replace the original species, as happens in UEGO_med. This is because the comparison in terms of fitness may be misleading, since the objective value at the midpoints or at the endpoints of the segments is only an *approximation*.

Furthermore, in order to reduce the large number of candidate points, those 'candidate' points are merged as described in subsection 4.2 (using the procedure *Fuse_species*). Finally, for each candidate point in this reduced list, its corresponding follower's facility is computed applying UEGO_med, and the objective value for the leader's facility is evaluated. The new species (with the corresponding radius according to the iteration) are inserted in the *species_list*.

Optimize_species procedure: For every species in the list, the local optimization process described in Algorithm 5 is applied. In Step 2, the SASS+WLM local search is applied (see [37]). This method tries to obtain a better solution for the leader (nf_1) based on the current choice of the follower (nf_2). To do so, this algorithm uses the stochastic hill climber SASS (see [46]) for updating the leader's facility and WLM for updating the follower's. Notice that the algorithm WLM is used because obtaining the exact new follower's facility every time the leader's facility changes, using UEGO_med, makes the process very time-consuming. Nevertheless, to prevent that the objective value for the leader becomes misleading (overestimated), UEGO_med is used in Step 3 of Algorithm 5. Finally, the species is replaced only in case a better objective function value is obtained (see steps 5 to 9 of Algorithm 5).

4.4 The cost of a myopic decision

A study is carried out to know how important it is to consider the follower's reaction. To this aim, for fourteen problems, we have calculated the leader's profit by solving

Algorithm 5: Algorithm LeaderOpt

Let (nf_1, nf_2, R) be the species to be optimized.
opt_nf_1 = SASS+WLM(nf_1, nf_2, R)
opt_nf_2 = UEGO_med(opt_nf_1)
if $opt\_nf_1 = nf_1$ **then**
  if $\Pi_2(nf_1, nf_2) > \Pi_2(nf_1, opt\_nf_2)$ **then**
    $opt\_nf_2 = nf_2$
  Update the original species to (nf_1, opt_nf_2, R).
else if $\Pi_1(opt\_nf_1, opt\_nf_2) > \Pi_1(nf_1, nf_2)$ **then**
  Update the original species to (opt_nf_1, opt_nf_2, R)

the medianoid problem but interchanging the roles of the leader and the follower and only taking the original m facilities into account, i.e., the *reverse medianoid problem*. The corresponding optimal solution will be denoted by $nf_1^{(myop)}$. Then, we have solved the corresponding medianoid problem, taking the existing m facilities and $nf_1^{(myop)}$ into account, using UEGO_med. And finally, we have evaluated $\Pi_1^{(myop)} = \Pi_1(nf_1^{(myop)},\ \text{UEGO\_med}(nf_1^{(myop)}))$.

Table 2 shows the obtained results. The first column refers to the setting of the problems solved (for three settings, more than one problem was generated, and the letters a, b, and c at the end of the setting has been added to highlight it). Columns two and three show the values of $nf_1^{(myop)}$ and $\Pi_1^{(myop)}$. The following two columns provide the values of the facility (nf_1^*) and the profit (Π_1^*) obtained with UEGO_cent.SASS. Finally, the loss in profit caused by the myopic decision as compared to the long term decision, in percentage, is shown.

Table 2 Comparison between the myopic and the long term view.

(n,n,k)	$nf_1^{(myop)}$			$\Pi_1^{(myop)}$	nf_1^*			Π_1^*	% loss
	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

As can be seen, the loss is less than 1% for half of the problems, it is over 4% for 6 out of 14 problems, and it exceeds 11% in three of them. This clearly indicates how important anticipating the competitor's reaction is, since the loss that can be produced may be substantial. Furthermore, note that the obtained results are independent of the setting (n,m,k) of the problem. Notice, for example, that the two extreme cases, with 0% loss and 28.15% loss, have the same configuration $(50,5,0)$. What is important is the actual distribution of the demand points and the actual locations and qualities of the existing facilities. Notice also that even though $nf_1^{(myop)}$ may be close to nf_1^*, the value of $\Pi_1^{(myop)}$ may be very different from Π_1^*, see problem (50,5,0)b.

4.5 High performance computing for the leader-follower problem

UEGO_cent.SASS is a costly algorithm, since the evaluation of the objective function value implies the resolution of a global optimization problem. Its parallelization may allow to reduce the execution time and to increase the size of the problems that can be solved. In [41], a *master-slave* algorithm and four *coarse-grain* methods are presented to parallelize UEGO_cent.SASS. The efficiency of the parallel algorithms is tested through an extensive computational testbed. Results showed that the *master-slave* method outperforms all the *coarse-grain* proposals, i.e. it is able to solve more instances using fewer processing elements and to obtain efficiencies close to or even greater than the ideal one.

In the following, the main features of the *master-slave* strategy are detailed. Readers interested in delving into the *coarse-grain* methods as well as into the performance comparison among parallel algorithms are referred to [41].

4.5.1 A master-slave strategy (MS)

Broadly speaking, in this parallel strategy, two types of processing elements are considered: the *master* processor, which makes global decisions and delivers data among the slaves, and the *slaves*, which execute different tasks simultaneously.

In our particular master-slave (MS) model (see Algorithm 6), the master processor executes UEGO_ cent.SASS sequentially. The parallelism has been included in new creation and optimization procedures (see Steps 5 and 8 in Algorithm 6). Next, they are briefly described.

- *Create_species_paral*: In this procedure, the master obtains a new offspring of candidate solutions for the leader sequentially. The parallelism comes from the simultaneous resolution of the medianoid problems to evaluate the new leader's trial points. To do so, the master divides the list of candidate solutions by the number of processors P and delivers the resulting sublists among all the process-

Algorithm 6: Algorithm MS

```
1: Init_species_list
2: Optimize_species(n_1)
3: for i = 2 to L do
4:    Determine R_i, new_i, n_i
5:    Create_species_paral(new_i)
6:    Fuse_species(R_i)
7:    Shorten_species_list(M)
8:    Optimize_species_paral(n_i)
9:    Fuse_species(R_i)
```

ing elements (including itself). Each processing element applies UEGO_med to every received leader's facility to obtain the associated follower's location.
The master processor does not receive information from the slaves until it has finished its work (first synchronization point). When it does so, it picks up all the follower sublists sent by the slaves, updates the candidate solutions list with such information and includes it in the $species\_list_i$, with the radius value associated to the current level i.

- *Optimize_species_paral*: In this procedure, the master divides the $species\_list_i$ among all the processing elements (again including itself). Once the sublist has been received, each slave applies the local optimization process SASS+WLM to every leader's facility and executes UEGO_med to obtain the corresponding follower (see [41]). Finally, once the master finishes its work, it starts to receive the new species sublists from the slaves (second synchronization point).

Note that the synchronization points are imposed because the master is working with the whole $species\_list_i$, or because it is needed to know the fitness value at the points of the leader before executing the next stage of the optimization procedure.

4.5.2 Improving the quality of the solution: a new creation procedure

Parallel algorithms can use more computational resources. Then, they can incorporate computationally intensive techniques that help at intensifying the search for more effective solutions. In [41], new alternative procedures to be included in UEGO_cent.SASS are studied. In particular, new creation methods that explore the search space deeper are analysed. After an exhaustive computational study, where several options are examined, it is found that the procedure named $Create\_species_{21}$ is the best choice, since it maintains a good balance between the quality of the final solution and the execution time and memory resources required by UEGO_cent.SASS.

The idea behind this method is to take advantage of the non-consumed evaluations of the previous level. The budget per species in the $Optimize\_species$ procedure is $bo_i = n_i/M$. This means that there is a remainder of $n_i - bo_i \cdot length(species\_list_i)$ function evaluations in the optimization process, when the length of the $species\_list_i$

is not equal to the maximum allowed. Then, these function evaluations can be used to force the creation of more candidate solutions at the next level. Therefore, the budget per species in the level $i+1$ is:

$$bc_{i+1} = \frac{new_{i+1} + n_i - bo_i \cdot length(species\_list_i)}{length(species\_list_{i+1})}.$$

As a consequence of the previous generation procedure, a huge list of candidate solutions is obtained. To reduce the list length while keeping the most promising solutions, a fusion procedure with the radius set to $2R_i$ is applied.

This new creation procedure makes the sequential UEGO_cent.SASS run out of memory most of the times. Then, to be able to use it, high performance computers are required. In [41], this new proposal is checked with the *master-slave* parallel model, since this algorithm does not modify the behavior of the sequential version, i.e., it considers the same number of function evaluations and acts over the species in the same way as the sequential algorithm. For the studies, the use of 2 processing elements has been enough to solve all the problems. An exhaustive analysis has proved that the $Creation\_species_{21}$ method can improve the objective value more than 1% in some instances, which is not a negligible value.

4.5.3 Efficiency results of MS

In this subsection the behavior of MS is analyzed by solving a representative set of location problems. The settings (n,m,k) employed in this experiment can be seen in Table 3. For every setting, five problems are generated. Furthermore, all the instances are solved 5 times and average values are considered.

Table 3 Settings of the larger test problems.

n	100			150			200		
m	1	2	5	1	3	7	2	5	10
k	0	0, 1	0, 2	0	0, 1	0, 3	0, 1	0, 2	0, 5

Table 4 shows average results (for all the values of m and k) for each value of n and P. In the column labelled $Av(Obj)$, the average objective function value is given, in $Av(T)$ the average computational time and in the last column $Eff(P,Q)$, efficiency values are given.

Results reveal how costly solving the centroid problem is. As can be seen in Table 4, the higher the number of demand points of the problem at hand, the larger the minimum number of processing elements required to solve it. Nevertheless, the performance of the parallel algorithm is good, i.e. its efficiency is larger than the ideal one for problems with 100 and 200 demand points, and very close to ideal for problems with $n = 150$.

Table 4 Efficiency results.

n	P	$Av(Obj)$	$Av(T)$	$Eff(P,Q)$
100	2	472.66	2512.24	-
	4	472.66	1218.48	1.03
	8	472.67	580.96	1.08
	16	472.66	271.28	1.06
	32	472.66	152.44	1.03
150	4	646.90	2271.08	-
	8	646.90	1161.28	0.99
	16	646.90	582.28	0.98
	32	646.90	295.71	0.96
200	8	850.70	964.53	-
	16	850.70	474.53	1.02
	32	850.70	238.74	1.01

5 A model with costs and variable demand

5.1 The model

The model considered in this section, introduced in [36], extends the previous model by relaxing the assumption that the demand is fixed. On the contrary, an endogenous (variable) demand is contemplated so that it varies depending on several factors. In real problems, for example, consumer expenditures on services or products that are offered by the facilities may increase depending on different reasons related to the location of the new facility. So, opening new outlets may increase the overall utility of the product. Also, the 'marketing presence' of a product may be increased with the marketing expenditures resulting from the new facilities. Another thing that can happen is that some consumers who did not patronize any of the facilities may now be induced to do so. The quality of the facilities may also modify consumer expenditures because a better service usually leads to more sales. The fact that the demand is endogenous is commonly disregarded in literature, usually due to the difficulty of the problems to be solved (see [35]).

The demand at a demand point p_i is now assumed to be a function of $U_i(nf_1,nf_2) = u_{i,nf_1} + u_{i,nf_2} + \sum_{j=1}^{m} u_{i,j}$, in the form

$$w_i(U_i(nf_1,nf_2)) = w_i^{\min} + incr_i \cdot e_i(U_i(nf_1,nf_2)),$$

where $incr_i = w_i^{\max} - w_i^{\min}$, and $w_i^{\max}$ (resp. $w_i^{\min}$) denotes the maximum (resp. minimum) possible demand at p_i. Function $e_i(U_i(nf_1,nf_2))$ can be interpreted as the share of the maximum possible increment that a customer decides to spend given a location scenario.

The objective functions Π_2 for the follower problem and Π_1 for the leader one, are formulated as in Section 4.1 (see (6) and (7), respectively), although

the market share function expressions (M_l) contain the variable demand function $w_i(U_i(nf_1,nf_2))$ instead of the constant $\widehat{w}_i$:

$$M_2(nf_1,nf_2) = \sum_{i=1}^{n} w_i(U_i(nf_1,nf_2)) \frac{u_{i,nf_2} + \sum_{j=k+1}^{m} u_{i,j}}{u_{i,nf_1} + u_{i,nf_2} + \sum_{j=1}^{m} u_{i,j}},$$

$$M_1(nf_1,nf_2) = \sum_{i=1}^{n} w_i(U_i(nf_1,nf_2)) \frac{u_{i,nf_1} + \sum_{j=1}^{k} u_{i,j}}{u_{i,nf_1} + u_{i,nf_2} + \sum_{j=1}^{m} u_{i,j}}.$$

The operating costs also are modified to include the variable demand in the $\Phi_l^i(d_i(z_l))$ functions, so that now

$$\Phi_l^i(d_i(z_l)) = Aver_{A_i}(w_i(U_i(nf_1,nf_2)))/((d_i(z_l))^{\phi_l^{i0}} + \phi_l^{i1}).$$

$Aver_{A_i}(w_i(U_i(nf_1,nf_2)))$ stands for the average value of $w_i(U_i(nf_1,nf_2))$ over the feasible set and can be thought of as an estimation of the demand at p_i by a fixed number (see [35] for more details about how to compute this average). In [36] *linear expenditures* is considered, i.e., $w_i^{\min} = 0$, $w_i(U_i(nf_1,nf_2)) = w_i^{\max} \cdot e_{i_1}(U_i(nf_1,nf_2))$, where $e_{i_1}(U_i(nf_1,nf_2)) = q_i U_i(nf_1,nf_2)$, with q_i a given constant such that $q_i \leq 1/U_i^{\max}$, where $U_i^{\max}$ is the maximum utility that could be observed by a customer at i.

Certainly, other functions could be defined depending on the real problem considered, and for each real application the most appropriate F_l and G_l functions should be discovered. In [48] a pseudo-real application to the case of the location of supermarkets in the Autonomous Region of Murcia, in Southern Spain, can be found. Although in that paper the demand was assumed to be exogenous (fixed) and no reaction from the competitor was expected, the parameters and functions have the same meaning as those in this section.

It must be emphasized that although the objective function of the follower's problem with exogenous demand is multimodal, it tends to be smoother than the one of the follower's problem with endogenous demand, which has much more local optima and whose landscape is much steeper. Consequently, the complexity of the centroid problem is greatly increased due to the endogenous demand assumption.

5.1.1 A real example

In order to show the difficulty of the problem at hand, and its differences with the exogenous demand case, in [36] the quasi-real example introduced in [48] dealing with the location of supermarkets in an area around the city of Murcia was solved. There are five supermarkets in the area: three from a first chain, 'E', and two from another chain, 'C'. Two problems have been considered: the first one assumes that the leader belongs to chain 'E' and the second one assumes that it belongs to chain 'C'. Each problem was solved both considering fixed and variable demand. The numerical results are shown in Table 5. The interested reader can find a detailed description of the example with some illustrative figures in [36].

Table 5 Examples.

Demand	nf_1	M_1	m_1	Π_1	nf_2	M_2	m_2	Π_2
Leader: chain E								
Exogenous	(3.303, 6.433, 0.500)	18.915	2.112	593.352	(3.259, 4.285, 3.696)	16.625	7.123	461.776
Endogenous	(5.407, 5.798, 0.961)	2.807	0.419	73.454	(5.190, 6.276, 0.571)	3.618	0.249	101.563
Leader: chain C								
Exogenous	(8.487, 3.026, 3.277)	15.961	6.247	442.122	(3.274, 6.441, 0.500)	19.579	2.187	614.652
Endogenous	(5.368, 6.166, 1.042)	3.822	0.453	106.320	(5.298, 6.228, 0.571)	2.6378	0.2489	70.227

As can be seen, when the leader belongs to chain 'E', in the exogenous demand case, the optimal location for the leader is near the city of Alcantarilla ($x_1 = 3.303, y_1 = 6,433$), with a quality of 0.5. At that location, the market share captured by the new leader's facility is $m_1 = 2.112$, which coincides with the 5.94% of the total market share. Taking into consideration all its facilities, chain 'E' obtains 53.22% of the market, and a profit $\Pi_1 = 593.352$. The location for the follower's facility is near the city of Molina ($x_1 = 3.259, y_1 = 4.285$), with a quality of 3.696, where it captures 20.04% of the total market share. However, the results are rather different for the endogenous case, where the leader's optimal location is in the suburb of Puente Tocinos ($x_1 = 5.407, y_1 = 5.798$), in Murcia city, with a quality of 0.961. The market share captured by the facility is 0.419, which is only 5.94% of the total one. The whole chain obtains 43.68% of the market and a smaller profit $\Pi_1 = 73.454$. The location for the follower's facility is near the suburb of San Benito ($x_1 = 5.190, y_1 = 6.276$), in Murcia city, with a quality of 0.571, where it only captures 3.875% of the total market share.

For the second problem, where it is assumed that chain 'C' is the leader, then, in the exogenous demand case, the optimal location for the leader is near the city of Orihuela, with a quality of 3.277, where the facility gets 17.57% of the total market share. The location for the follower's facility is near the city of Alcantarilla, with a quality of 0.5, where it captures 6.15% of the total market share. However, the leader's optimal location in the endogenous demand case is near the suburb of San Benito, in Murcia city, with a quality of 1.042 and only captures 6.52% of the total market share. The location for the follower's facility is near the suburb of San Benito too, with a quality of 0.571, where it captures 3.88% of the total market share.

These two examples indicate how important it is to consider endogenous demand. As can be seen, depending on whether endogenous or exogenous demand is considered, the maximum profit for a chain is obtained at different locations and with different qualities. Additionally, it is interesting to remark that even the percentage of market share captured by the chains may change to the point that the chain obtaining more profit may be the competitor's one.

5.2 Solving the centroid problem

Considering the algorithms proposed for solving the centroid problem with exogenous demand (see section 4.3), the following three algorithms are implemented to solve the centroid problem with endogenous demand [36]: a grid search procedure, a multistart method named MSH, and an evolutionary algorithm named TLUEGO. MSH and TLUEGO require the use of a local optimizer. In particular, a local optimizer based on SASS and WLM has been designed. In fact, two variants of the local optimizer have been implemented, leading to two versions of MSH and TLUEGO. Next we describe the corresponding algorithms.

5.2.1 The local optimizer SASS+WLMv

In [37], after studying several strategies, a local procedure SASS+WLMv, similar to SASS+WLM in Section 4.3 is proposed. The main differences between this local algorithm and SASS+WLM are:

- The Weiszfeld-like algorithm used now for updating the follower's facility is WLMv, a variant of WLM to take the variability of the demand into account (see [35]). Similar to what was considered for WLM (see Subsection 4.2.1), WLMv stops when either two consecutive iterations are closer than the tolerance $\varepsilon_1 = \varepsilon_2 = 0.0001$, or when a maximum number of $ic_{max} = 400$ iterations is reached.
- Due to the high increment in the complexity of the problem when using endogenous demand, the WLMv algorithm is not as reliable as the corresponding method WLM for the fixed demand case. Consequently, due to the cumulative error, a large number of consecutive iterations in SASS could give rise to the leader achieving overestimated solutions. To deal with this drawback, the number of consecutive iterations in SASS+WLMv has been reduced to only 15. In addition, in order to compensate the possible error obtained using WLMv, after every 15 iterations, the medianoid problem is solved accurately using a reliable global optimizer. Two global optimizers have been considered: iB&B [18] or UEGO_med (see Section 4.2), resulting in two versions of the local optimizer.

5.2.2 TLUEGO: A two-level evolutionary global optimization algorithm

The evolutionary algorithm TLUEGO is rather similar to the UEGO_cent.SASS algorithm introduced in section 4.3 for the fixed demand case. The main differences are the following:

- *Create_species procedure:* In the same way that for UEGO_cent.SASS, after the *creation* procedure it is very important to precisely evaluate the fitness of the new species. In this problem, two alternative algorithms to compute a reliable follower solution have been implemented: iB&B or UEGO_med.

Algorithm 7: Algorithm SASS+WLMv$(nf_1, nf_2, iter_{\max}(=15), \sigma_{ub})$

1: Initialize SASS parameters. Set $iter = 1, nf_1^{opt} = nf_1, \Pi_1^{opt} = \Pi_1(nf_1, nf_2)$.
2: **while** $iter \leq iter_{\max}$ **do**
3: Update SASS parameters considering the previous successes at improving the objective function value of the leader.
4: Generate a location for the leader $nf_1^{(iter)}$ within the updated radius.
5: Solve the corresponding medianoid problem using WLMv and let $nf_2^{(iter)}$ denote the solution obtained.
6: **if** $\Pi_1(nf_1^{(iter)}, nf_2^{(iter)}) > \Pi_1^{opt}$ **then**
7: set $nf_1^{opt} = nf_1^{(iter)}$ and $\Pi_1^{opt} = \Pi_1(nf_1^{(iter)}, nf_2^{(iter)})$.
8: $iter = iter + 1$.
9: Compute the corresponding follower nf_2^{opt} for nf_1^{opt} using either iB&B or UEGO.
10: **if** $\Pi_1(nf_1^{opt}, nf_2^{opt}) > \Pi_1(nf_1, nf_2)$ **then**
11: return (nf_1^{opt}, nf_2^{opt})
12: **else**
13: Return (nf_1, nf_2).

- *Optimize_species procedure:* The *local optimizer* algorithm used in TLUEGO is SASS+WLMv. There is another difference: this local optimizer is executed twice in order to have more chances of obtaining a better point. The input parameter value of σ_{ub} passed to SASS+WLMv is always (the two times it is called) the radius associated to the calling species. Therefore, the scope of the local optimizer coincides with the region covered by the species. As it has been mentioned in 5.2.1, the execution of SASS+WLMv implies that a reliable optimization algorithm, iB&B or UEGO_med, is run at the end of the algorithm (Step 9 in Algorithm 7). As a result, the inclusion of iB&B or UEGO_med in TLUEGO derives two algorithms for solving the centroid problem, TLUEGO_BB and TLUEGO_UE, respectively. The reader is referred to [36] for a more detailed description of these procedures.

5.2.3 MSH: A multistart heuristic algorithm

The MSH algorithm consists of randomly generating *MaxStartPoints* feasible candidate solutions for the leader and then applying a local optimizer to each one in order to improve it to an optimized leader solution. The final solution provided by the algorithm will be obtained by selecting the solution with best objective function value.

For this problem with exogenous demand, the considered local optimizer has been SASS+WLMv (see Algorithm 7). In order to provide a better balance between exploitation and exploration of the search space, this method has also been executed twice as in TLUEGO, but with different values for σ_{ub} because the multistart heuristic does not have a cooling process for the radius. In the first call, a value of $\sigma_{ub} = 2.083895$ (the one corresponding to level 10 in TLUEGO) was considered.

This value was chosen because then the initial random candidate solutions in the multistart strategy can cover the whole searching space, and at the same time, they can search on an area small enough so that the local procedure can find a good local optimum. In the second call, a value of $\sigma_{ub} = 0.162375$ (level 23 in TLUEGO) was used to improve the quality of the local optima obtained with the first call. These σ_{ub} values were selected after doing some preliminary studies, in which eight problems of different sizes were solved trying different strategies for the heuristic algorithm.

As in TLUEGO, two versions of the MSH method have been implemented: MSH_BB and MSH_UE. They differ in whether iB&B or UEGO_med is used as a method of computing the follower nf_2^{opt} in Step 9 of Algorithm 7.

5.2.4 Computational studies

To study the performance of the algorithms, a set of 24 problems has been generated 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 Table 6. For each setting, the problem has been generated by randomly choosing its parameters within given intervals. In all the problems, $S_1 = S_2 = ([0,10],[0,10])$ and $\alpha_1, \alpha_2 \in [0.5,5]$.

For every heuristic algorithm, each problem has been solved ten times and average values have been computed. However, the heuristic GS has only been run once and the results obtained in that run (no average results) are given. All results for all the problems are shown in [36]. In this section only some average results for $n = 15$ and $n = 50$ are shown in Table 7. In the column labeled '*Time*', the average time in the ten runs (in seconds) of each problem is shown; the '*MaxDist*' column indicates the maximum Euclidean distance (for the three variables (x_1, y_1, α_1)) between every pair of solutions provided by the algorithm in different runs, which gives an idea of how far these solutions can be; in the following three columns, the minimum, the average and the maximum objective value are computed. Finally, in the '*Dev*' column, the standard deviation is shown. As can be seen in these tables, two versions of TLUEGO and MSH algorithms have been executed. It is worth mentioning that the number of times that MSH_BB (resp. MSH_UE) was allowed to repeat its basic local optimizer was chosen so that the CPU time employed by MSH_BB (resp. MSH_UE) was, on average (when considering all the problems with the same value of n), similar to the CPU time employed by TLUEGO_BB (resp. TLUEGO_UE) or a bit higher. In particular, for the problems with 15 and 50 demand points, the number of starting points were 150 and 250, respectively.

Table 6 Settings of the test problems.

n	15			25			50		
m	2	5	10	2	5	10	2	5	10
k	0,1	0,1,2	0,2,4	0,1	0,1,2	0,2,4	0,1	0,1,2	0,2,4

Table 7 Results for the problems with $n = 15$ and $n = 50$. TLUEGO_BB ($\varepsilon_1 = \varepsilon_2 = 0.0001$), TLUEGO_UE, MSH_BB and MSH_UE and GS.

			Max	Objective Function			
(n)	Algorithm	*Time*	*Dist*	*Min*	*Av*	*Max*	*Dev*
15	TLUEGO_BB	226	0.015	15.478	15.478	15.479	0.000
	TLUEGO_UE	891	0.009	15.478	15.478	15.479	0.001
	MSH_BB	258	1.164	15.350	15.413	15.453	0.038
	MSH_UE	1091	0.516	15.290	15.409	15.469	0.067
	GS	490338	-	-	15.445	-	-
50	TLUEGO_BB	9470	0.186	39.866	39.960	40.065	0.081
	TLUEGO_UE	8259	0.185	39.912	40.072	40.174	0.102
	MSH_BB	11090	2.855	25.597	31.329	37.722	4.508
	MSH_UE	9911	2.769	23.769	33.088	38.084	5.346
	GS	3003794	-	-	37.280	-	-

Analyzing the results, it can be seen that the method used to reliably solve the medianoid problem does not seem to have an influence on the quality of the final solution, i.e., TLUEGO and MSH behave similarly, regardless whether iB&B or UEGO_med is employed. This is due to the reliability of UEGO (in spite of its metaheuristic nature). The iB&B technique is faster than UEGO_med for small size problems ($n = 15$), which directly reduces the execution time of both TLUEGO and MSH. Specifically, the use of iB&B reduces the computing time of TLUEGO_BB by 74.6% as compared to TLUEGO_UE. A similar behavior in computing time can be seen in MSH when iB&B is used instead of UEGO_med. Nevertheless, for medium size problems (with $n = 50$ demand points), TLUEGO_UE and MSH_UE reduce the computing time as compared to TLUEGO_BB and MSH_BB, by 12.79% and 10.63%, respectively. These results are also consistent with the ones showed in [39], where it was observed that the increase of requirements for iB&B with the size of the problem was greater than for UEGO_med.

Focusing now on the strategies proposed to solve the current centroid problem, it can be stated that TLUEGO (in both versions) is the algorithm achieving the best results. Their average objective function values are always higher than the ones provided by both MSH and GS. It is also remarkable that the minimum objective function value found by TLUEGO in the ten runs is always better than the average values obtained by both MSH and GS (see columns '*Min*' and '*Av*'). Additionally, TLUEGO is the most robust algorithm in the sense that it usually attains the same solution in all the runs, whereas MSH is more erratic, and can provide different solutions in each run (see the values of '*MaxDist*' and '*Dev*').

5.3 Influence of the fuse process in the creation procedure

Taking into account the main structure of TLUEGO, based on UEGO_cent.SASS algorithm, it can be seen that in the creation procedure, for every species in the list, a set of possible new solutions is computed, fused and evaluated with the objective

of finding new promising species, and therefore increasing the species-list. This creation process is applied independently to each species as no relation among species exists.

Taking into consideration that the evaluation of a single species in TLUEGO requires intensive computational effort, since it implies the execution of another expensive optimization algorithm (UEGO_med or iB&B) to obtain the optimal location of the follower (by solving the corresponding medianoid problem), TLUEGO had to be designed to maintain a small-size species-list. This was done by including a *'fuse'* process just after the creation of candidate solutions and before the evaluation of the resulting ones.

However, it is known that working with larger species-list sizes helps to explore the search space deeply and consequently to obtain better solutions. With this aim, in this section, new creation procedures are proposed, where the fuse process is relaxed in part by modifying the threshold distance to apply the fusion of two species. Now two species will be fused if the distance between their centers is smaller than the new thresholds R_t, $R_t/2$ or 0 instead of $2R_t$. In what follows, only TLUEGO_UE will be used, since it can solve larger instances. It will simply be denoted by TLUEGO. For the analysis at hand, only medium size problems have been considered, i.e. $n = 50, 100$ (the actual settings can be seen in Table 8).

Table 8 Settings of the test problems.

n	50			100		
m	2	5	10	2	5	10
k	0,1	0,1,2	0,2,4	0,1	0,1,2	0,2,4

Table 9 Effectiveness evaluation of the fuse process in TLUEGO (sequential algorithm) for problems with $n = 100$ and $n = 50$ demand points.

n	threshold	*Time*	*MaxDist*	Π_1	*Dev*	$Dif\Pi_1$	*DifSol*
50	$2R_t$	10993	0.520	148.316	0.578	-	-
	R_t	17689	0.307	149.616	0.177	0.782	1.812
	$R_t/2$	18686	0.129	150.296	0.113	1.235	2.364
	0	22898	0.135	151.002	0.064	1.794	2.940
100	$2R_t$	32029	0.755	177.364	1.992	-	-
	R_t	52125	0.146	183.341	0.490	3.260	4.221
	$R_t/2$	56932	0.133	185.710	0.272	4.562	5.998
	0	65470	0.056	186.551	0.058	5.033	7.027

Considering that each run of TLUEGO may provide a different solution, each problem has been solved ten times and average values have been computed. Table 9 shows the average results obtained by the algorithms considering all the configurations for the problems with $n = 50$ and $n = 100$, respectively. In [36] a complete set of tables with detailed results for each configuration can be found. The first column gives the size of the problem. The second one indicates the threshold value used in the fuse process. In the third column, the average time in the ten runs (in seconds) is computed. The *MaxDist* column provides the maximum Euclidean distance (for the

three variables (x_1, y_1, α_1)) between any pair of solutions provided by the algorithm in the ten runs, which gives an idea of how far the solutions computed by the algorithm in different runs can be. The average objective function value (column Π_1) in the ten runs and the corresponding standard deviation (column *Dev*) are given next. Column $Dif\Pi_1$ shows the relative improvement in the objective function value between the solution obtained by the algorithms when a threshold different from $2R_t$ is used as compared to the result obtained when using $2R_t$. The final column shows the relative difference between the solutions.

As can be seen, the CPU time increases as the threshold decreases, and when this is set to 0, the time is more than double as compared to the $2R_t$ case. The algorithm also becomes more robust (see the decrease in columns *Dev*), in the sense that the objective function value at different runs are more similar. In addition, analysing column Π_1 it can be deduced that the quality of the solution also becomes better. Regarding the relative improvement in the objective function value, it can be seen that for the problems with $n = 50$ demand points is moderate, with an average of 1.794%. However when the threshold is set to 0, for the problems with $n = 100$ it attains a significant 5.033%. This clearly shows that the smaller the threshold, the better the solutions are. Unfortunately this is at the cost of increasing the CPU time and the memory requirements.

5.4 High performance computing

Due to the high computational cost of TLUEGO, which is even higher than that of UEGO_cent.SASS, a parallelization of the algorithm is required, especially if real problems, with more demand points than the studied in the previous section must be solved. In [1], three programming paradigms for the parallelization of TLUEGO are designed. More specifically, a pure message passing paradigm, a pure shared memory programming model and a hybrid one which combines message passing with shared memory are implemented and their efficiency and effectiveness are analyzed and compared. Results showed that both pure message passing and pure shared memory paradigms have almost the same performance, while the hybrid one shows less efficiency though it can exploit all computational resources of the parallel architecture.

Considering that TLUEGO structure is similar to UEGO_cent.SASS, the message passing algorithm is based on a *master-slave strategy* like the one described in section 4.5. For this reason only the main features of pure shared memory strategy are detailed here. Readers interested in a deep description of the three strategies as well as in the performance comparison among them are referred to [1].

5.4.1 Shared memory programming for TLUEGO: SMP_TLUEGO

For the implementation of this parallel strategy, OpenMP has been selected, since it is a portable and scalable model, and gives programmers a simple and flexible interface for developing parallel applications.

Concerning the parallel model, it can be considered a *pseudo* master-slave technique, similar to the MS described in section 4.5. OpenMP includes mechanisms to distribute the species list among the different processors without the existence of a master processor. Therefore there does not exist a master processor which globally controls the algorithm and manages the species list. This task can be done in parallel by all the processors. However, the existence of a kind of pseudomaster processor to be in charge of applying the *Selection* procedure and updating the species list that will be accessible to all processors, is still necessary. Accordingly, the parallelism is applied to the evaluation of the new candidate solutions in the *Creation* and *Optimization* procedures. Consequently, new creation and optimization procedures have also been designed. They are briefly described next.

The parallel algorithm developed considers that the species-list is stored in shared memory. When the *Create_species_paral* is executed, each processor picks up a new single species and evaluates it. Once a processor has finished this task, it collects another species. This cyclical process finished when all the new offspring are evaluated. Notice that mutual exclusion is not needed because each processor accesses different memory areas.

The *Optimize_species_paral* procedure maintains a similar structure to the previous method *Create_species_paral*. But instead of only evaluating the species, it applies the local search procedure. Considering that the number of function evaluations required to optimize a single species, and therefore, the computational load assumed by each processor, may be quite different, this strategy of selecting the species one by one helps to balance the computational burden and to reduce the waiting time of the processors.

5.4.2 Efficiency results of SMP_TLUEGO

In this subsection the behavior of SMP_TLUEGO is analyzed by solving a set of 24 problems whose settings can be found in Table 10. For every setting one problem was generated. Additionally, all the instances are solved 10 times and average values are considered.

Table 10 Settings of the test problems.

n	50			100		
m	2	15	25	2	15	25
k	0,1	0,5,10	0,7,15	0,1	0,5,10	0,7,15

Table 11 Efficiency results for SMP_TLUEGO.

P	n	$Time$	$Eff(P)$	n	$Time$	$Eff(P)$
1	100	65470	-	500	565358	-
2		32878	1.00		283707	1.00
4		16928	0.97		143416	0.99
8		8703	0.94		73065	0.97

Table 11 shows, for the problems with $n = 100$ and $n = 500$ demand points, the average computing time (in secs.) and the mean efficiency $Eff(P)$ obtained. As can be seen, SMP_TLUEGO has either optimal or near-optimal efficiency for up to $P = 8$ processors. For a given n the efficiency values slightly decrease as the number of processors P increases. Notice, however, that the algorithm is *scalable*, as it shows a better performance (see $Eff(P)$ columns) when the problem size increases, i.e. the efficiency improves with higher n values.

6 Solving the models with costs exactly

In this section we propose an exact solution method for the problems described in sections 4 and 5, i.e. when operational costs are taken into account. As already mentioned, the B&B method described in Section 3 works only when no costs are present, that is, the zero-sum property holds for the objective functions of the leader and follower. The method we propose to solve these harder problems exactly is a generalization of the algorithm presented in [49]. In that paper almost the same problem is solved exactly on networks, although with fixed qualities. Here, we propose a modification of this method to be able to solve the problem on the plane having the quality as additional variables for the new facilities.

In [49] a B&B method is used to solve the leader problem, while in an embedded way another B&B was used to refine the follower. The main difference between this method and algorithms 2 and 1 is that the follower problem has to be solved for a set of leader placements instead of for a leader point. This is much more challenging, and it may even be impossible if the aim is to solve the problem with a small accuracy. Therefore, instead of solving the follower problem in the inner B&B to optimality, its searching set is only refined, and the solution (set of sets) is stored together with the leader set. The method proposed next differs from that in [49] mainly in the searching space and the solution sets, that instead of being segments of edges of the network, they are now 3-dimensional boxes (vector of intervals) in $\mathbb{R}^3$.

6.1 Overcoming the difficulty of the lack of the zero-sum property

In Section 3 we have already seen that when the objective function is the market share (no costs are present), and the qualities of the facilities are given parameters, the problem can be solved efficiently by a B&B method. The key point there is the zero-sum property of the objective functions: minimizing the objective of the leader, one directly maximizes the objective of the follower and vice-versa. What makes the method very efficient is that although (reverse) medianoid problems have to be solved to obtain bounds, the other new facility is always fixed to a point. This is no longer the case when costs are taken into account. It may even happen that changing the location of the follower increase both the leader and the follower objective. Therefore the result of Lemma 1 cannot be used directly, and so a new trick is needed to overcome this difficulty.

When operational costs are present, for the bound calculations of the leader, all possible locations (and qualities) of the follower have to be considered. On the one hand, until the follower is not enclosed tightly in a set of boxes, it might mean that the obtained bounds are very loose. On the other hand, until the leader box is not small enough, it is not possible to enclose the follower tightly. Thus, what is needed is a good and possibly cheap bound calculation procedure in order to overcome the above problem. One promising approach is to use *interval bounds*, as done in [49].

6.2 Interval arithmetic bounds

We propose to use *Interval Arithmetic* to obtain lower and upper bounds of the objective functions automatically when one or both facilities are in boxes. The main idea of Interval Analysis is to change all real arithmetic operators and elementary real functions to their interval versions. As a result, an interval containing all possible results from points from the input intervals is obtained, maybe with some overestimation. See [21] for details of interval analysis in global optimization.

Let us denote intervals with capital letters, e.g. $X = [\underline{x}, \overline{x}]$, where $\underline{x} \leq \overline{x}$ are the lower and upper bounds of X, respectively.

For a given box NF_l containing a new facility nf_l, an interval U_{i,nf_l} containing the utility of any point within NF_l can be computed as

$$U_{i,nf_l} = [\underline{u_{i,nf_l}}, \overline{u_{i,nf_l}}] = [\gamma_i \underline{\alpha_l} / g_i(\overline{d_i(Z_l)}), \gamma_i \overline{\alpha_l} / g_i(\underline{d_i(Z_l)})]$$

where

$$\underline{d_i(Z_l)} = \sqrt{(\max\{\underline{x_l} - p_{i1}, p_{i1} - \overline{x_l}, 0\})^2 + (\max\{\underline{y_l} - p_{i2}, p_{i2} - \overline{y_l}, 0\})^2},$$

$$\overline{d_i(Z_l)} = \sqrt{\max\{(\underline{x_l} - p_{i1})^2, (p_{i1} - \overline{x_l})^2\} + \max\{(\underline{y_l} - p_{i2})^2, (p_{i2} - \overline{y_l})^2\}}.$$

Given a fixed box (or a point) $\widetilde{NF}_2$ for the follower, an upper bound of Π_1 at the box NF_1 can be calculated with interval arithmetic as

$$UB(\Pi_1(NF_1,\widetilde{NF}_2)) = c \cdot UB(M_1(NF_1,\widetilde{NF}_2)) - LB(G_1(NF_1)),$$

where the upper bound of the market share is given by the formula

$$UB(M_1(NF_1,\widetilde{NF}_2)) = \sum_{i=1}^{n} \widehat{w}_i \frac{\overline{u_{i,nf_l}} + \sum_{j=1}^{k} u_{ij}}{\underline{u_{i,nf_1}} + \underline{u_{i,nf_2}} + \sum_{j=1}^{m} u_{ij}},$$

when the demand is fixed, and

$$UB(M_1(NF_1,\widetilde{NF}_2)) = \sum_{i=1}^{n} w_i^{\max} q_i(\overline{u_{i,nf_l}} + \sum_{j=1}^{k} u_{ij}),$$

when the demand is endogenous but linear as introduced in Section 5.

The lower bound $LB(G_1(NF_1))$ of the operational cost function G_1, when it has the form

$$G_1(nf_1) = \sum_{i=1}^{n} w_i/((d_i(z_1))^{\phi_1^{i0}} + \phi_1^{i1}) + \exp(\alpha_1/\xi_1^0 + \xi_1^1) - \exp(\xi_1^1)$$

(where w_i stands for $\widehat{w}_i$ when the demand is fixed, and for $w_i(U_i(nf_1,nf_2))$ when the demand varies) can be computed as

$$LB(G_1(NF_1)) = \sum_{i=1}^{n} \frac{\widehat{w}_i}{\overline{d_i(Z_1)}^{\phi_1^{i0}} + \phi_1^{i1}} + \exp(\underline{\alpha_1}/\xi_1^0 + \xi_1^1) - \exp(\xi_1^1)$$

when the demand is fixed, and as

$$LB(G_1(NF_1)) = \sum_{i=1}^{n} \frac{w_i^{\min}}{\overline{d_i(Z_1)}^{\phi_1^{i0}} + \phi_1^{i1}} + \exp(\underline{\alpha_1}/\xi_1^0 + \xi_1^1) - \exp(\xi_1^1)$$

when it varies.

Of course, if an upper bound for the leader's profit is required when the follower is in a set of boxes $\mathbb{NF}_2$, it can be obtained as

$$UB(\Pi_1(NF_1,\mathbb{NF}_2)) = c \cdot \max_{NF_2 \in \mathbb{NF}_2} UB(M_1(NF_1,NF_2)) - LB(G_1(NF_1)).$$

The interval arithmetic lower bound of the profit can be obtained by interchanging upper bounds and lower bounds in the above formulae. The bounds for the follower are straightforward by the rules above.

One can see that even those computations might be time-consuming for obtaining an upper or a lower bound. However, notice that in the fixed demand case, we can still use the zero-sum property of the market share for its bound calculations, so that

if bounds for the follower's market share are known, they can be used directly for the leader's bounds on the market share and vice-versa.

6.3 Solution method

A B&B method is designed to solve the leader's problem, and consequently the follower's problem as well. The main goal of the method is for every subproblem to simultaneously tighten the set containing the global optimizer of the leader and the set that contains all the global optimizers for the follower problem.

Without loss of generality, it is assumed that the feasible set of both the leader and the follower is a box. We define subproblems of the leader as boxes. For a given box of the leader, the follower's possible position can be in many places, and until the leader is not enclosed tightly, the follower can only be bounded to a set of boxes. Therefore, for every box of the leader we need to store the subboxes that may contain the global optimal solutions of the follower. Hence, a partial solution or subproblem of the leader refers to a box containing the leader and the set of boxes that contain the corresponding solution of the follower problem.

An inner B&B method tightens the boxes of the follower, and a main (outer) B&B method tightens the boxes of the leader. Thus, lower and upper bounds for the leader's (follower's) profit are needed when the follower (leader) is enclosed in a box. For the calculation of the lower and upper bounds of the follower in a given box NF_2, its corresponding single leader's box NF_1 is taken into account. These lower and upper bounds are $LB(\Pi_2(NF_1,\widehat{nf}_2))$ and $UB(\Pi_2(NF_1,NF_2))$, respectively, where $\widehat{nf}_2 \in NF_2$ is a feasible solution within the follower's box. For the calculation of the bounds for a leader's box NF_1, every box of the follower corresponding to it has to be considered, i.e. $LB(\Pi_1(\widehat{nf}_1,\mathbb{NF}_2))$ and $UB(\Pi_1(NF_1,\mathbb{NF}_2))$, where $\widehat{nf}_1$ is a feasible solution in the leader's box and $\mathbb{NF}_2 \ni NF_2$ the set of the corresponding boxes of the follower.

6.3.1 Inner B&B

Both the leader's and their corresponding follower's boxes need to be refined for the algorithm to converge. The inner B&B takes care of the refinement of the follower's boxes.

The termination criterion of the inner B&B is to have the size of each follower's box at least as small as the corresponding leader's box. The algorithm returns the modified list of the boxes of the follower. The selection rule chooses the largest box, while the branching rule bisects the box perpendicularly to the coordinate direction of maximum width.

Given a leader box, this method is applied to the set of follower boxes associated to it, until the corresponding follower's sub-boxes have a size smaller than or equal

to that of the leader's box. Each time a new leader box is created, the inner B&B is run until its follower's boxes are refined.

6.3.2 Outer B&B

The outer B&B refines the leader's boxes and calls the inner B&B method for each new box of the leader. Recall that a subproblem of the leader is a box with the corresponding set of boxes for the follower. Thus, the initial subproblem is the starting box of the leader, and the starting box of the follower. However it might be more efficient to make a pre-division at the very beginning, as the first lower and upper bounds obtained by the algorithm are usually useless, but computing them needs time.

The output is a set of boxes containing any global optimizer, and the interval containing their objective values contains the global optimum of the problem. The selection rule selects the leader box with the highest upper bound of the leader's profit, while the branching rule bisects the leader's box perpendicularly to the coordinate direction of maximum width and leaves the follower's boxes unchanged but duplicated for the new boxes of the leader. The algorithm stops when the interval containing the objective values of all leader's boxes gets smaller than a prescribed tolerance or the size of all the boxes becomes smaller than another tolerance parameter.

6.4 Algorithm

The pseudocode of the inner and outer B&B algorithms are given in Algorithm 8. For the sake of simplicity let us denote the objective function as Π (Π_1 for the outer and Π_2 for the inner B&B).

In line 3 we remove each box known not to contain any global optimizer from list Λ. The main cycle of the general B&B method is listed from line 4 to line 19. The main difference of the outer B&B from the inner B&B is the call of the inner method added in lines 15 and 16. In fact, the additional differences between the inner and outer procedures are hidden in the bound calculations, as well as in the selection and termination rules.

The output of Algorithm 8 is the set of boxes which could not be eliminated and thus contain any global optimizer, and the point at which the best lower bound was achieved.

The proposed method should be tested on a set of test problems to know the size of the problems that it can solve, for both exogenous and endogenous demand. However, this is not the aim of this section, but to show that an exact algorithm can be designed even if operational costs are considered, the qualities are variables of the model and the demand is endogenous.

Algorithm 8: The inner and outer B&B methods

Input: Λ, GLB for the inner B&B
$\Lambda = \{S\}, GLB = -\infty$ for the outer B&B
Remove all NF^i from Λ with $UB^i < GLB$
while $\Lambda \neq \emptyset$ **do**
  Select NF from Λ
  Bisect NF into NF^1 and NF^2
  for $i := 1$ **to** 2 **do**
    Determine an upper bound UB^i on NF^i
    if not $UB^i < GLB$ **then**
      Compute a lower bound LB^i of Π at midpoint(NF^i)
      if $LB^i > GLB$ **then**
        $GLB := LB^i$, $BestPoint :=$ midpoint(NF^i)
        Remove all NF^j from Λ with $UB^j < GLB$
      if not TerminationCriterion(NF^i) **then**
        if outer **then**
          Call the inner B&B on the set of follower boxes of NF^i
        $\Lambda := \Lambda \cup \{NF^i\}$
      else
        $\Gamma := \Gamma \cup \{NF^i\}$
Output: $\Gamma, BestPoint$

7 Conclusions and future research

Despite its inherent difficulty, facility location leader-follower (or Stackelberg) problems can be addressed when the location space considered is the plane, at least in its simple case, when only one new facility is going to be located by the leader and the follower. Exact (interval) branch-and-bound methods can be put to work for solving small instances, whereas evolutionary algorithms can handle large instances. If so required, parallel implementations of the algorithms can help to solve larger instances and with more accuracy.

Dealing with problems where more than one facility is to be located by the leader and/or the follower seems to still be a challenge when the location space is the plane. An extension which deserves to be explored is to allow the existing facilities to modify their quality, or even close some of them. Studying the problems with other patronizing behavior of customers is another line of future research. From the computational point of view, the design of high performance computing approaches for the exact branch-and-bound algorithms is also worth exploring.

Acknowledgements This research has been supported by grants from the Spanish Ministry of Economy and Competitiveness (MTM2015-70260-P, and TIN2015-66680-C2-1-R), the Hungarian National Research, Development and Innovation Office - NKFIH (OTKA grant PD115554), Fundación Séneca (The Agency of Science and Technology of the Region of Murcia, 19241/PI/14), Junta de Andalucía (P11-TIC7176 and P12-TIC301), in part financed by the European Regional Development Fund (ERDF). Juana López Redondo is a fellow of the Spanish 'Ramón y Cajal' contract program.

References

1. A.G. Arrondo, J.L. Redondo, J. Fernández, and P.M. Ortigosa. Solving a leader-follower facility problem via parallel evolutionary aproaches. *The Journal of Supercomputing*, 70(2):600–611, 2014.
2. J. Bhadury, H.A. Eiselt, and J.H. Jaramillo. An alternating heuristic for medianoid and centroid problems in the plane. *Computers and Operations Research*, 30(4):553–565, 2003.
3. B. Biesinger, B. Hu, and G. Raidl. Models and algorithms for competitive facility location problems with different customer behavior. *Annals of Mathematics and Artificial Intelligence*, 76(1):93–119, 2016.
4. M.S. Daskin. *Network and discrete location: models, algorithms and applications*. Wiley, 1995.
5. S. Dempe. *Foundations of Bilevel Programming*. Springer, 2002.
6. P. Dorta-González, D.R. Santos-Peñate, and R. Suárez-Vega. Spatial competition in networks under delivered pricing. *Papers in Regional Science*, 84:271–280, 2005.
7. T. Drezner. Locating a single new facility among existing unequally attractive facilities. *Journal of Regional Science*, 34(2):237–252, 1994.
8. T. Drezner. Optimal continuous location of a retail facility, facility attractiveness, and market share: an interactive model. *Journal of Retailing*, 70(1):49–64, 1994.
9. T. Drezner and Z. Drezner. Replacing continuous demand with discrete demand in a competitive location model. *Naval Research Logistics*, 44:81–95, 1997.
10. T. Drezner and Z. Drezner. Facility location in anticipation of future competition. *Location Science*, 6(1):155–173, 1998.
11. T. Drezner and Z. Drezner. Retail facility location under changing market conditions. *IMA Journal of Management Mathematics*, 13(4):283–302, 2002.
12. T. Drezner, Z. Drezner, and P. Kalczynski. Strategic competitive location: improving existing and establishing new facilities. *Journal of the Operational Research Society*, 63(12):1720–1730, 2012.
13. T. Drezner, Z. Drezner, and P. Kalczynski. A leader–follower model for discrete competitive facility location. *Computers & Operations Research*, 64:51–59, 2015.
14. Z. Drezner. Competitive location strategies for two facilities. *Regional Science and Urban Economics*, 12(4):485–493, 1982.
15. H.A. Eiselt and G. Laporte. Sequential location problems. *European Journal of Operational Research*, 96(2):217–231, 1996.
16. H.A. Eiselt, G. Laporte, and J.F. Thisse. Competitive location models: a framework and bibliography. *Transportation Science*, 27(1):44–54, 1993.
17. J. Fernández, P. Fernández, and B. Pelegrín. 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(6):609–623, 2002.
18. J. Fernández, B. Pelegrín, F. Plastria, and B. Tóth. Solving a Huff-like competitive location and design model for profit maximization in the plane. *European Journal of Operational Research*, 179(3):1274–1287, 2007.
19. J. Fernández, S. Salhi, and B. G. Tóth. Location equilibria for a continuous competitive facility location problem under delivered pricing. *Computers and Operations Research*, 41(1):185–195, 2014.
20. S.L. Hakimi. On locating new facilities in a competitive environment. *European Journal of Operational Research*, 12(1):29–35, 1983.
21. E. Hansen and G. W. Walster. *Global optimization using interval analysis*. Marcel Dekker, second revised and expanded edition, 2004.
22. D. L. Huff. A Programmed Solution for Approximating an Optimum Retail Location. *Land Economics*, 42(3):293–303, 1966.
23. M. Jelásity. *The shape of evolutionary search: Discovering and representing search space structure*. Ph.D. Thesis, Leiden University, 2001.

24. M. Jelásity, P.M. Ortigosa, and I. García. UEGO, An Abstract Clustering Technique for Multimodal Global Optimization. *Journal of Heuristics*, 7(3):215–233, 2001.
25. Hande Küçükaydin, Necati Aras, and I. Kuban Altinel. A leader-follower game in competitive facility location. *Computers and Operations Research*, 39(2):437–448, 2012.
26. H. Kücükaydin, N. Aras, and I.K. Altinel. Competitive facility location problem with attractiveness adjustment of the follower: A bilevel programming model and its solution. *European Journal of Operational Research*, 208(3):206–220, 2011.
27. P.J. Lederer and A.P. Hurter. Competition of firms: discriminatory pricing and location. *Econometrica*, 54(3):623–40, 1986.
28. R.G. McGarvey and T.M. Cavalier. Constrained location of competitive facilities in the plane. *Computers and Operations Research*, 32:359–378, 2005.
29. T. C. Miller, T. L. Friez, and R. L. Tobin. *Equilibrium facility location on networks*. Springer, 1996.
30. P.B. Mirchandani and R.L. Francis, editors. *Discrete location theory*. Wiley-Interscience, 1990.
31. P.M. Ortigosa, I. García, and M. Jelásity. Reliability and performance of UEGO, a clustering-based global optimizer. *Journal of Global Optimization*, 19(3):265–289, 2001.
32. F. Plastria. GBSSS, the generalized big square small square method for planar single facility location. *European Journal of Operational Research*, 62:163–174, 1992.
33. F. Plastria. Avoiding cannibalization and/or competitor reaction in planar single facility location. *Journal of the Operations Research Society of Japan*, 48:148–157, 2005.
34. F. Plastria and E. Carrizosa. Optimal location and design of a competitive facility. *Mathematical Programming*, 100(2):247–265, 2004.
35. J.L. Redondo, J. Fernández, A.G. Arrondo, I. García, and P.M. Ortigosa. Fixed or variable demand? Does it matter when locating a facility? *Omega*, 40(1):9–20, 2012.
36. J.L. Redondo, J. Fernández, A.G. Arrondo, I. García, and P.M. Ortigosa. A two-level evolutionary algorithm for solving the facility location and design $(1|1)$-centroid problem on the plane with variable demand. *Journal of Global Optimization*, 56(3):983–1005, 2013.
37. J.L. Redondo, J. Fernández, I. García, and P. M. Ortigosa. Heuristics for the facility location and design $(1|1)$-centroid problem on the plane. *Computational Optimization and Applications*, 45(1):111–141, 2010.
38. J.L. Redondo, J. Fernández, I. García, and P.M. Ortigosa. Parallel algorithms for continuous competitive location problems. *Optimization Methods & Software*, 23(5):779–791, 2008.
39. J.L. Redondo, J. Fernández, I. García, and P.M. Ortigosa. A robust and efficient global optimization algorithm for planar competitive location problems. *Annals of Operations Research*, 167(1):87–106, 2009.
40. J.L. Redondo, J. Fernández, I. García, and P.M. Ortigosa. Solving the multiple competitive facilities location and design problem on the plane. *Evolutionary Computation*, 17(1):21–53, 2009.
41. J.L. Redondo, J. Fernández, I. García, and P.M. Ortigosa. Solving the facility location and design $(1|1)$-centroid problem via parallel algorithms. *The Journal of Supercomputing*, 58(3):420–428, 2011.
42. J.L. Redondo, P.M. Ortigosa, I. García, and J.J. Fernández. 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(II):141–149, 2004.
43. N. Saidani, F. Chu, and H. Chen. Competitive facility location and design with reactions of competitors already in the market. *European Journal of Operational Research*, 219(1):9–17, 2012.
44. M.E. Sáiz, E.M.T. Hendrix, J. Fernández, and B. Pelegrín. On a branch-and-bound approach for a Huff-like Stackelberg location problem. *OR Spectrum*, 31:679–705, 2009.
45. D. Serra and C. ReVelle. *Facility location: a survey of applications and methods*, chapter Competitive location in discrete space, pages 367–386. Springer, 1995.
46. F.J. Solis and R.J.B. Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6(1):19–30, 1981.

47. B. Tóth and J. Fernández. *Interval methods for single and bi-objective optimization problems - applied to competitive facility location problems*. Lambert Academic Publishing, Saarbrücken, 2010.
48. B. Tóth, F. Plastria, J. Fernández, and B. Pelegrín. On the impact of spatial pattern, aggregation, and model parameters in planar Huff-like competitive location and design problems. *OR Spectrum*, 31(1):601–627, 2009.
49. B. G. Tóth and K. Kovács. Solving a Huff-like Stackelberg location problem on networks. *Journal of Global Optimization*, 64(2):233–257, 2016.