**Noname manuscript No.**
(will be inserted by the editor)

# On the Weber facility location problem with limited distances and side constraints

**Isaac F. Fernandes · Daniel Aloise · Dario J. Aloise · Pierre Hansen · Leo Liberti**

Received: date / Accepted: date

**Abstract** The objective in the continuous facility location problem with limited distances is to minimize the sum of distance functions from the facility to the customers, but with a limit on each of the distances, after which the corresponding function becomes constant. The problem has applications in situations where the service provided by the facility is insensitive after a given threshold distance. In this paper, we propose a global optimization algorithm for the case in which there are in addition lower and upper bounds on the numbers of customers served.

**Keywords** facility location · global optimization · reformulation · decomposition

## 1 Introduction

The continuous minisum single facility problem is one of the most fundamental problems in location theory. The objective is to locate a single facility in the plane so that the sum of distances from the facility to a set of demand points is minimized. The problem is often referred to in the literature as the Weber (or Fermat-Weber) problem [4,16,18]. It traces back to Fermat in the $17^{th}$

I. Fernandes, D. Aloise and D.J. Aloise
Universidade Federal do Rio Grande do Norte, Campus Universitário s/n, Natal-RN, Brazil, 59072-970
E-mail: isaac@syncsistemas.com, daniel.aloise@gerad.ca, aloise@pep.ufrn.br

P. Hansen
GERAD and HEC Montréal, 3000, Chemin de la Côte-Sainte-Catherine, Montréal, Québec, Canada, H3T 2A7
E-mail: pierre.hansen@gerad.ca

L. Liberti
LIX, École Polytechnique, F-91128, Palaiseau, France
E-mail: leo.liberti@lix.polytechnique.fr

century who posed a purely geometrical version of the problem with only three points. Torricelli, in 1647, is credited to prove that the circles circumscribing the equilateral triangles constructed on the sides of the triangle formed by the three given points intersect in the fourth point sought (see [24,26] for a historical survey).

Drezner, Mehrez and Wesolowsky investigated in [13] the Weber problem for the case in which the distance functions are constant after given threshold values, which they call *the facility location problem with limited distances*. This problem has applications in situations where the service provided by the facility is insensitive after a given threshold time/distance. For instance, consider the problem of locating a fire station. In this context, each property has a distance limit after which the service provided by the firemen is useless, and the property is completely destroyed. An example of this operations research application is provided in [13]. The authors suppose a situation where a certain damage occurs in a property located in $p_i$ for $i = 1, \ldots, n$ at zero distance from the fire station (located at $y \in \mathbb{R}^2$), and that this damage linearly increases up to a distance $\lambda_i$ where the damage is 100%. By denoting $d(p_i, y)$ the distance between point $p_i$ and the facility located at $y$, and $\Omega$ the proportion of damage at zero distance, the proportion of damage in $p_i$ is given by $\Omega + (1 - \Omega)d(p_i, y)/\lambda_i$ in the case $d(p_i, y) < \lambda_i$, and 1 otherwise. The corresponding facility location problem is then expressed as

$$\min_{y \in \mathbb{R}^2} \sum_{i=1}^{n} \Omega + (1 - \Omega) \frac{\min\{d(p_i, y), \lambda_i\}}{\lambda_i} \tag{1}$$

The first term of the summation is constant and $(1-\Omega)$ is irrelevant to the second term. By introducing binary variables $v_i$, that select between $d(p_i, y)$ and $\lambda_i$, to the summation of the objective function, we end up with the following minimization problem

$$\min_{y \in \mathbb{R}^2, v \in \{0,1\}^n} \sum_{i=1}^{n} \frac{1}{\lambda_i} \left(\lambda_i(1 - v_i) + d(p_i, y)v_i\right). \tag{2}$$

Real examples for the application of this location model also include other types of emergency services (e.g. ambulances, police calls). For example, a person suffering from a heart attack has more chances to survive if he/she is quickly treated, and will certainly die if help does not come after a given period of time. In the case of a police call, criminals would be likely untraceable after a time limit.

In this work, we study the version of the problem for which there are lower and upper bounds on the number of demand points that can be served within the distance limits. This is indeed a natural extension to the model presented in [13]. In practical applications, a lower bound in the number of served points may be used to justify the installation of a facility (e.g. it is not reasonable to construct a fire station that can save only a few properties nearby), while an upper bound may express the capacity limitations of the service to maintain an acceptable quality level.

Another extension to the facility location problem with limited distances formulated in [13] is presented by Drezner, Wesolowsky and Drezner in [15]. In this paper, the authors formulate a problem in which the distance function from the demand point is equal to zero up to a first threshold value $l$, linear between $l$ and a second threshold distance value $u$, and constant after $u$. The model is equivalent to that of [13] when $l = 0$. In one hand, the mathematical developments presented here cannot be used to help solving the global optimization problem presented in [15], at least not straightforwardly. On the other hand, adding the side constraints to the model of [15] would require a considerable effort on investigating new lower and upper bounds to be used in a specialized branch-and-bound. We decided to work on an extension to the model of [13] by means of convex exact reformulations in the sense of [20]. This yielded an enumerative algorithm based only on the resolution of convex problems.

An adjacent facility location problem to the one approached here is the Maximal Covering Location Problem (MCLP) [8,9,21] which maximizes the number of demand points covered within a specified critical distance or time by a fixed number of facilities. Although these models also incorporate facility distance limitations, they deal with a covering objective function which is mathematically distinct from the minisum objective. Covering problems are a chapter apart in the facility location theory (see [3,22,14] for a survey).

The mathematical formulation of the problem approached in this paper is given in the next section. A global optimization algorithm for it is described in Section 3. Computational results on synthesized instances and on a real-life problem are reported in Section 4 and compared with those of the literature. Finally, conclusion are given in the last section.

## 2 Problem definition

Let us denote $\|p_1 - p_2\|_q$ the $\ell_q$-distance between points $p_1$ and $p_2$ in the plane. Given $n$ service points in the plane $p_1, p_2, \ldots, p_n$ with threshold distances $\lambda_i > 0$ and weights $w_i \geq 0$ for $i = 1, \ldots, n$, the Limited Distance Minisum Problem with Side Constraints (LDMPSC) can be expressed by:

$$\begin{aligned} &\min_{y \in \mathbb{R}^2, v \in \{0,1\}^n} \sum_{i=1}^{n} w_i \left(\lambda_i(1 - v_i) + \|p_i - y\|_q v_i\right) \\ &\text{subject to} \\ &\|p_i - y\|_q v_i \leq \lambda_i \qquad \text{for } i = 1, \ldots, n \qquad (3) \\ &L \leq \sum_{i=1}^{n} v_i \leq U. \end{aligned}$$

The first set of constraints defines bounds $L$ and $U$ in the number of variables $v_i$ which can be equal to 1. The second set of constraints assures that $v_i$ can be equal to 1 only if the distance between $p_i$ and the facility located at $y$ is

inferior (or equal) to the distance limit $\lambda_i$. This avoids the attribution $v_i = 1$ only to satisfy constraint $\sum_{i=1}^{n} v_i \geq L$. The objective function of (3) as well as its feasible set are non convex which demands more sophisticated solution methods.

The objective function of (3) can still be rewritten thereby removing its constant terms. It is then expressed as

$$\sum_{i=1}^{n} w_i \lambda_i + \min \sum_{i=1}^{n} w_i (\|p_i - y\|_q - \lambda_i) v_i. \tag{4}$$

## 3 Optimization algorithm

From the formulation above, we have that for a given location $y$, $v_i$ may be equal to 1 only if $\|p_i - y\|_q \leq \lambda_i$. If $q = 2$, this is geometrically equivalent in the plane to the condition that $v_i$ may be equal to 1 only if $y$ belongs to a disc $\mathcal{D}_i = \{y | \|p_i - y\|_2 \leq \lambda_i\}$ (i.e., a disc with radius $\lambda_i$ centered at $p_i$). Analogously, if $q = 1$, this is equivalent to the condition that $v_i$ may be equal to 1 only if $y$ belongs to a square $\mathcal{S}_i = \{y | \|p_i - y\|_1 \leq \lambda_i\}$ with sides making a 45 (or -45) degree angle with the axes (i.e., a 45 degrees rotated square with diagonal $2\lambda_i$ centered at $p_i$).

A branch-and-bound algorithm based on the vector $v$ would consider implicitly all $2^n$ subproblems generated by branching on binary variables $v_i$ for $i = 1, \ldots, n$, while adding constraints $\|p_i - y\|_q \leq \lambda_i$ and $\|p_i - y\|_q \geq \lambda_i$ to the resulting subproblems. Another possibility is to focus on components $v_i$ of $v$ which might be equal to 1 at the same time. When $q = 2$, these components are directly associated to convex regions generated by intersections of discs (see Figure 1). For instance, for the region indicated by the bullet in Figure 1, only the components $v_1$, $v_2$ and $v_3$ can be equal to 1.

Hence, we can solve (3) by solving subproblems of the following type:

$$\begin{aligned} \min_{y \in \mathbb{R}^2, v \in \{0,1\}^{|S|}} \quad & \sum_{i \in S} w_i (\|p_i - y\|_q - \lambda_i) v_i \\ \text{subject to} \\ & \|p_i - y\|_q \leq \lambda_i \quad \forall i \in S, \\ & L \leq \sum_{i \in S} v_i \leq U \end{aligned} \tag{5}$$

where $S \subseteq \{1, 2, \ldots, n\}$ is a non-empty set. Each one of the subproblems of type (5) is associated to a distinct region in the plane. For instance, we have a subproblem with $S = \{1, 2, 3\}$ for the region indicated by the bullet in Figure 1. The number of these convex regions for $q \geq 1$ was proved to be quadratically bounded in the number of points in [13].[1]

[1] the proof omits a case. We completed the proof for $q = 2$ in [1].

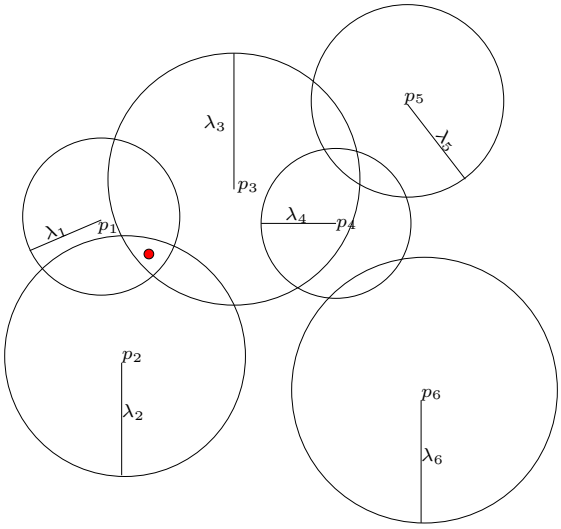


**Fig. 1** Intersection of discs.

Problem (5) contains (nonconvex) bilinear terms in the objective function, so that its continuous relaxation is not necessarily easy to solve. To address this issue we propose the following reformulation of (5): we add variables $z_i \in [-\lambda_i, 0]$ for all $i \in S$, we replace the objective function with $\sum_{i \in S} w_i z_i$, and adjoin the following constraints:

$$\forall i \in S \quad \|p_i - y\|_q - \lambda_i \leq z_i \tag{6}$$

$$\forall i \in S \quad z_i + \lambda_i v_i \geq 0. \tag{7}$$

We remark that the original constraints $\forall i \in S$ ($\|p_i - y\|_q \leq \lambda_i$) are also part of the reformulation, as well as bound and integrality constraints on $y, v$. The resulting reformulation:

$$\left.\begin{array}{rr} \min_{y \in \mathbb{R}^2, v \in \{0,1\}^{|S|}} & \sum\limits_{i \in S} w_i z_i \\ \forall i \in S & \|p_i - y\|_q - \lambda_i \leq z_i \\ \forall i \in S & z_i + \lambda_i v_i \geq 0 \\ \forall i \in S & \|p_i - y\|_q \leq \lambda_i \\ & L \leq \sum_{i \in S} v_i \leq U \\ \forall i \in S & z_i \in [-\lambda_i, 0]. \end{array}\right\} \tag{8}$$

is an *exact reformulation* of (5), i.e. the optima of (8) can be mapped surjectively onto the optima of (5) [20], as shown in Prop. 1. We also remark that (8) is a convex Mixed-Integer Nonlinear Program (MINLP), as it involves continuous and integer variables as well as nonlinear terms, and its continuous relaxation is a convex Nonlinear Program (NLP). This is because: (i) all norms are convex functions [7, p. 73], (ii) adding a linear function to a convex one

results in a convex function, and (iii) bounding convex functions from above defines a convex set.

**Proposition 1** *Prob.* (8) *is an exact reformulation of Prob.* (5)*.*

*Proof* Let $(y^*, v^*, z^*)$ be an optimal solution of (8). For $i \in S$, if $v_i^* = 0$ then, because $\|p_i - y^*\|_q \leq \lambda_i$, the left hand side of (6) is nonpositive; hence, by (7), $z_i^* \geq 0$ is the most stringent constraint. By the upper bound constraint on $z$, we have $z_i^* = 0$. If $v_i^* = 1$, (7) implies $z_i^* \geq -\lambda_i$, which is redundant with respect to the lower bound constraint on $z$. Hence, by (6), we have $z_i^* \geq \|p_i - y^*\|_q - \lambda_i$, and the objective function direction enforces $z_i^* = \|p_i - y^*\|_q - \lambda_i$. Thus, the optimal objective function value of (8) is the same as that of (5) at the optimum $(y^*, v^*)$ of (5). As concerns surjectivity, it is easy to remark that for each feasible $(y, v)$ in (5) there exist a corresponding $z$ such that $(y, v, z)$ is feasible in (8): simply define $z_i = (\|p_i - y\|_q - \lambda_i)v_i$. Hence, the projection operator of the optima of (8) onto the variables of (5) is surjective on the optima on (5) and certifies that the reformulation is exact.

For $q = 1$, Prob. (8) is a Mixed Integer Linear Program (MILP); this follows by a classical exact reformulation of convex constraints involving absolute values $|x| \leq r$ into pairs of constraints $-r \leq x \leq r$. For example, $\|p_i - y\|_1 = |p_{i_1} - y_1| + |p_{i_2} - y_2| \leq \lambda_i$ is initially replaced by:

$$p_{i_1} - y_1 \leq \lambda_i - |p_{i_2} - y_2| \qquad \text{and} \qquad p_{i_1} - y_1 \geq |p_{i_2} - y_2| - \lambda_i.$$

Then, each of the constraints above is replaced by two others, yielding the constraints:

$$\begin{aligned} y_1 + y_2 &\geq p_{i_1} + p_{i_2} - \lambda_i \\ y_1 - y_2 &\geq p_{i_1} - p_{i_2} - \lambda_i \\ y_1 - y_2 &\leq p_{i_1} - p_{i_2} + \lambda_i \\ y_1 + y_2 &\leq p_{i_1} + p_{i_2} + \lambda_i. \end{aligned}$$

For $q = 2$, Prob. (8) is a convex MINLP. Due to the presence of the square root term in the Euclidean norm, the problem is not everywhere differentiable. Specifically, this sometimes causes floating-point errors in local NLP solvers, but there exist practically efficient convex MINLP solvers for this type of problems (e.g. Bonmin [5]).

Proposition 2 shows that the resolution of subproblems can still be further simplified, allowing to solve simpler subproblems whenever their size (i.e., $|S|$) is not greater than $U$.

**Proposition 2** *There exists a solution* $(y^*, v^*)$ *optimal to (5) such that* $V^* = \{i | v_i^* = 1\}$ *has cardinality equal to* $\beta = \min\{|S|, U\}$*.*

*Proof* The proof is done by construction. Let us assume $y^*$ as the optimal facility location and consider an initial solution $(y^*, v^*)$ with $V^* = \{i | v_i^* = 1\} = \emptyset$. This solution can be improved by choosing an element $i' = argmin_{i \in S}\{w_i(\|p_i -$

$y^*\|_q - \lambda_i)\}$ and making $v^*_{i'} = 1$. Since $\|p_{i'} - y^*\|_q - \lambda_{i'} \leq 0$ and $w_i \geq 0$, the new solution is better (or equal) than the initial one. This procedure can be continuously repeated by choosing a new element $i'' = argmin_{i \in S, i \notin V^*}\{w_i(\|p_i - y^*\|_q - \lambda_i)\}$ until $|V^*| = |S|$ or $|V^*| = |U|$. The final solution $(y^*, v^*)$ constructed in this way is optimal since inserting elements to $V^*$ is no longer possible as well as removing elements from $V^*$ is not profitable. □

Proposition 2 ensures that the optimal solution of (5), and its exact reformulation (8), has all decision variables $v$ equal to 1 whenever $|S| \leq U$. In this case, subproblems can be expressed by

$$\begin{aligned} &\min_{y \in \mathbb{R}^2} \sum_{i \in S} w_i z_i \\ &\text{subject to} \\ &z_i \geq \|p_i - y\|_q - \lambda_i && \forall i \in S \\ &\|p_i - y\|_q \leq \lambda_i && \forall i \in S \\ &z_i \in [-\lambda_i, 0] && \forall i \in S \end{aligned} \tag{9}$$

which is an LP for $q = 1$ and a convex NLP for $q = 2$.

Algorithm 1 enumerates the sets $S$ corresponding to regions delimited by convex figures (i.e., rotated squares when $q = 1$, discs when $q = 2$). This algorithm executes in $O(n^2\tau)$ time where $\tau$ is the time required for solving each subproblem in steps 4 and 7. This time is larger when (8) is solved instead of (9).

Algorithm 1

1. Enumerate all intersection points of convex figures in the plane as well as all convex figures whose boundary does not intersect any other one. Let $L_1$ and $L_2$ be the corresponding lists.
2. For each intersection point $p \in L_1$ defined by convex figures centered at points $p_i$ and $p_j$, find the set $S$ of all $k$ such that $k \neq i, j$ and $\|p_k - p\|_q \leq \lambda_k$.
3. Consider the four sets: $S, S \cup \{i\}, S \cup \{j\}$, and $S \cup \{i, j\}$.
4. For each one of these sets, if $|S| \geq L$, solve the associated subproblem of type (8) if $|S| > U$. Otherwise, solve subproblem of type (9).
5. Update the best solution if an improving one is found.
6. For each convex figure in $L_2$ find the set $S'$ composed of its own index and the indices of all convex figures containing it.
7. For each one of these sets $S'$, if $|S'| \geq L$, solve the associated subproblem of type (8) if $|S'| > U$. Otherwise, solve subproblem of type (9).
8. Update the best solution if an improving one is found.

Step 1 in Algorithm 1 relies on the solution of a geometric problem consisting in enumerating all intersection points between pairs of convex figures in the plane. For $q = 1$, a possible approach for enumerating the intersection points of rotated squares is to use the popular sweep line algorithm [11], since a square can be decomposed into four distinct linear segments. For $q = 2$, a

pair of circles may intersect in a single degenerate point or in two distinct points (two identical circles coincide in an uncountable number of points, but this case occurs with probability 0 and is therefore omitted). An algorithm for enumerating all the intersection points for a set of circles can be found at [12]. For $q > 2$, the delimited regions are convex, but finding the intersection points for a pair of regions will involve the numerical solutions of a nonlinear system of two equations.

We remark that for $q = \infty$, the convex regions defined by $\|p_i - y\|_\infty \leq \lambda_i$, for $i = 1, \ldots, n$ are squares whose sides are orthogonal with the coordinate axes, and subproblems (8) can be reformulated as:

$$
\begin{aligned}
&\min_{y \in \mathbb{R}^2, v \in \{0,1\}^{|S|}} \sum_{i \in S} w_i z_i \\
&\text{subject to} \\
&z_i \geq t_i - \lambda_i && \forall i \in S \\
&z_i + v_i \lambda_i \geq 0 && \forall i \in S \\
&t_i \leq \lambda_i && \forall i \in S \\
&t_i \geq p_{i_1} - y_1 && \forall i \in S \\
&t_i \geq p_{i_2} - y_2 && \forall i \in S \\
&L \leq \sum_{i \in S} v_i \leq U \\
&z_i \in [-\lambda_i, 0] && \forall i \in S,
\end{aligned}
\tag{10}
$$

which is a a MILP problem.

## 4 Computational experiments

Our experiments are designed to assess the performance of Algorithm 1 on random instances, as well as compare it with respect to existing state-of-the-art nonconvex MINLP solvers, such as Couenne [2] and Baron [25]. These are two different implementations of the spatial Branch-and-Bound (sBB) algorithm [19]. Much like a Branch-and-Bound (BB) algorithm for MILPs, sBB explores the feasible space exhaustively but implicitly, finding a guaranteed $\varepsilon$-approximate solutions for any given $\varepsilon > 0$ in finite (worst-case exponential) time. Unlike MILPs, whose continuous relaxation is a Linear Program (LP), and unlike convex MINLPs, whose continuous relaxation is a convex NLP, the continuous relaxation of a nonconvex MINLP is generally difficult to solve. To address this issue, sBB algorithms form and solve *convex relaxations* of the given MINLP (the most common approach to build such relaxations is by using symbolic reformulation techniques [23]). The convexity gap between the original MINLP and its convex relaxation therefore stems from two factors: the relaxation of the integrality constraints, as well as the relaxation of the nonconvex terms appearing in the MINLP. Accordingly, sBB algorithms may

branch on both integer and continuous variables, when the latter occur in a nonconvex term.

Algorithm 1 iteratively solves subproblems of form (8). For the case $q = 2$, these are convex MINLPs, which we solve using Bonmin [5]. This is a software framework for convex MINLP, which implements different types of algorithms based on combining Outer Approximation (OA) with Branch-and-Bound techniques [6].

Problem instances were artificially generated from an uniform distribution in a square with sides equal to 1000, in order to evaluate the performance of Algorithm 1. The instances so obtained were created by stochastically controlling the number of intersections among the convex figures associated to each point $p_i$, for $i = 1, \ldots, n$. In $\ell_2$-norm, if there exists a pair of points $p_{i_1}$ and $p_{i_2}$ for which $\|p_{i_1} - p_{i_2}\|_2 < 2\lambda$, then their associated discs $\mathcal{D}_{i_1}$ and $\mathcal{D}_{i_2}$ intersect. Consequently, if there is a disc $\mathcal{D}_i$ associated to a point $p_i$ that does not intersect any other disc $\mathcal{D}_j$ for $j = 1, \ldots, n, j \neq i$, then no other point $p_j$ can be generated inside the disc $\overline{\mathcal{D}}_i = \{y | \|y - p_i\|_2 \leq 2\lambda\}$. The area of $\overline{\mathcal{D}}_i$ is equal to $\pi(2\lambda)^2 = 4\pi\lambda^2$. Hence, considering an uniform distribution, the probability $P$ that a region $\mathcal{D}_i$ centered at $p_i$ does not intersect another region associated to any of $n - 1$ points generated in a plane of dimension $d \times d$ is given by

$$\left(\frac{d^2 - 4\pi\lambda^2}{d^2}\right)^{n-1}. \tag{11}$$

Using the same reasoning in $\ell_1$-norm, the following probability formula is obtained

$$\left(\frac{d^2 - 8\lambda^2}{d^2}\right)^{n-1}. \tag{12}$$

By means of equations (11) and (12), one can derive threshold distance values for instances of (3) as a function of the desired probability of intersection among discs, for $q = 2$, and squares, for $q = 1$.

Table 1 and 2 present the values of threshold distances $\lambda$ obtained from equations (11) and (12) for $q = 1$ and $q = 2$, respectively, considering different values of $P$ and different number of points $n$.

**Table 1** Threshold distance values ($\lambda$) for $q = 1$

| | $n = 10$ | $n = 100$ | $n = 1000$ |
|---|---|---|---|
| $P = 10^{-3}\%$ | 300.36 | 117.15 | 37.85 |
| $P = 10^{-6}\%$ | 329.93 | 145.68 | 47.79 |
| $P = 10^{-9}\%$ | 342.79 | 167.98 | 55.94 |

Eighteen different instance categories were generated based on the scenarios presented in Tables 1 and 2; nine for $q = 1$ and nine for $q = 2$. For these instance categories, the threshold distance values are taken from the values presented in Tables 1 and 2 plus a perturbation obtained from a normal distribution with mean 0 and variance equal to 10, 5, and 2 for the instances

**Table 2** Threshold distance values ($\lambda$) for $q = 2$

| | $n = 10$ | $n = 100$ | $n = 1000$ |
|---|---|---|---|
| $P = 10^{-3}\%$ | 239.66 | 93.47 | 30.19 |
| $P = 10^{-6}\%$ | 263.25 | 116.24 | 38.13 |
| $P = 10^{-9}\%$ | 273.51 | 134.03 | 44.63 |

with $n = 10, 100$, and 1000, respectively. They are named according to the norm used, the number $n$ of points and the probability $P$ used to create each one of them. Thus, $\ell_1$_10_$10^{-3}$ refers to the category composed by instances of 10 points, uniformly distributed in a $1000 \times 1000$ square with threshold $\ell_1$-distances $\lambda_i \in \mathcal{N}(\mu = 300.36, \sigma^2 = 10)$, for $i = 1, \ldots, n$.

Ten distinct instances were generated in each category, totalizing $18 \cdot 10 = 180$ instances. For this set of instances, all the points have unitary weights (i.e., $w_i = 1$, for $i = 1, \ldots, n$). The instances used here can be found at `http://www.gerad.ca/~aloise/publications.html`.

Computational experiments were performed on a Pentium Quad Core Xeon X3353 with a 2.66 GHz clock and 24 Gigabytes of RAM memory. Algorithm 1 was implemented in C++ and compiled by gcc 4.4. Table 3 presents the computational results for the generated instances. Its first column contains the category identifiers. The second and third columns present the lower ($L$) and upper ($U$) bounds values used in the execution of each instance category. The fourth column presents average optimum solution values of the 10 instances in each category. The fifth, sixth, seventh and eighth columns report the average and standard deviations of CPU times (in seconds) spent by Couenne and Baron. The two solvers are executed in the same platform aforementioned, except for Baron executions of instances with 1000 points, for which we used the NEOS server [10] due to size limitation in our academic version. The platform used in the NEOS server was a Xeon X5660 with 2.80 Ghz clock and RAM memory capped to 3 Gigabytes per job. The ninth and tenth columns show, respectively, average CPU times and standard deviations of CPU times spent by Algorithm 1. Finally, the eleventh and twelveth columns report the average and standard deviations of the number of subproblems (8) solved within Algorithm 1. The subproblems were solved by CPLEX 12.1.0 for the case $q = 1$, and by Bonmin [5] for $q = 2$.

Results in Table 3 show that the optimum solution values increase with the threshold distances values $\lambda$ (see Table 1). Let us consider two instances $I_1$ and $I_2$ formed by the same set of points with distinct threshold distance values $\lambda_1$ and $\lambda_2$, respectively, for all of their points such that $\lambda_1 < \lambda_2$. Consequently, any feasible solution for $I_1$ is also feasible $I_2$. Furthermore, that solution has greater cost in $I_2$, since for all terms $i = 1, \ldots, n$ for which $v_i = 0$, the associated contribution in the objective function is $\lambda_2 (> \lambda_1)$. For the terms $i$ for which $v_i = 1$, the value of the associated contribution in the objective function (i.e., $\|p_i - y\|_1$) remains unchanged. Despite that, if two instances share the same set of points, we cannot state that the one with the largest threshold distance

**Table 3** Computational results of Algorithm 1, Couenne and Baron over the 180 generated instances.

| | | | | Couenne | | Baron | | Algorithm 1 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Category | $L$ | $U$ | opt.value | CPU time(s) | std.dev. | CPU time(s) | std.dev. | CPU time(s) | std.dev. | # subs | std.dev. |
| $\ell_1$_10_$10^{-3}$ | 2 | 5 | 2199.25 | 5.95 | 7.17 | 38.84 | 76.11 | 0.43 | 0.27 | 97.6 | 29.56 |
| $\ell_1$_10_$10^{-6}$ | 2 | 5 | 2389.11 | 7.90 | 10.52 | 35.04 | 68.49 | 0.49 | 0.21 | 120.4 | 26.75 |
| $\ell_1$_10_$10^{-9}$ | 2 | 5 | 2471.41 | 7.12 | 9.52 | 34.35 | 67.00 | 0.58 | 0.13 | 136.0 | 23.99 |
| $\ell_1$_100_$10^{-3}$ | 5 | 10 | 11140.50 | 40.88 | 9.82 | 244.18 | 154.89 | 4.67 | 1.71 | 1034.1 | 307.17 |
| $\ell_1$_100_$10^{-6}$ | 5 | 10 | 13766.76 | 56.94 | 12.66 | 274.71 | 172.39 | 14.73 | 5.96 | 2825.8 | 495.58 |
| $\ell_1$_100_$10^{-9}$ | 5 | 10 | 15787.32 | 74.82 | 22.55 | 453.34 | 306.70 | 28.70 | 11.99 | 4919.6 | 418.58 |
| $\ell_1$_1000_$10^{-3}$ | 10 | 20 | 34207.28 | 2022.68 | 183.83 | 17703.40 | 7169.93 | 0.84 | 0.27 | 110.11 | 59.27 |
| $\ell_1$_1000_$10^{-6}$ | 10 | 20 | 44037.39 | 3251.83 | 339.03 | 25896.00 | 1524.11 | 12.94 | 4.44 | 2737.8 | 1017.28 |
| $\ell_1$_1000_$10^{-9}$ | 10 | 20 | 52088.12 | 5148.95 | 846.75 | 27645.66 | 2511.78 | 67.48 | 13.72 | 14621.6 | 2901.73 |
| $\ell_2$_10_$10^{-3}$ | 2 | 5 | 1874.70 | 1.83 | 0.96 | 6.90 | 10.87 | 4.06 | 4.55 | 99.0 | 28.42 |
| $\ell_2$_10_$10^{-6}$ | 2 | 5 | 2026.11 | 1.88 | 0.67 | 1.55 | 1.14 | 5.97 | 7.96 | 116.0 | 26.02 |
| $\ell_2$_10_$10^{-9}$ | 2 | 5 | 2090.42 | 1.85 | 0.78 | 1.59 | 0.92 | 6.20 | 6.53 | 127.8 | 30.45 |
| $\ell_2$_100_$10^{-3}$ | 5 | 10 | 2199.25 | 66.85 | 31.10 | 92.34 | 19.67 | 104.31 | 51.29 | 1037.9 | 306.79 |
| $\ell_2$_100_$10^{-6}$ | 5 | 10 | 2389.11 | 144.10 | 410.03 | 94.21 | 26.26 | 269.85 | 84.04 | 2828.4 | 538.19 |
| $\ell_2$_100_$10^{-9}$ | 5 | 10 | 2471.41 | 379.02 | 334.88 | 90.80 | 24.78 | 542.75 | 169.91 | 4878.6 | 444.51 |
| $\ell_2$_1000_$10^{-3}$ | 10 | 20 | 29971.70 | 3016.79 | 859.24 | 8266.79 | 917.73 | 2.58 | 0.84 | 88.6 | 53.78 |
| $\ell_2$_1000_$10^{-6}$ | 10 | 20 | 37823.32 | 13916.58 | 3403.56 | 15898.89 | 2666.81 | 97.28 | 47.69 | 2654.8 | 991.75 |
| $\ell_2$_1000_$10^{-9}$ | 10 | 20 | 44243.26 | 30583.92 | 16320.27 | 18040.20 | 4351.75 | 515.68 | 162.04 | 14459.3 | 2995.86 |

has the largest optimal solution value (the trivial case is an infeasible location problem with $\lambda_1$, which becomes feasible by using $\lambda_2$).

The results in Table 3 also reveal that:

(i) Standard deviation values presented in the table are sometimes large. This demonstrates that CPU times spent by the tested algorithms in a category depend considerably on the distribution of points in the space. This observation did not invalidate what observed in (ii)-(v).
(ii) Couenne outperforms Baron in the $\ell_1$-norm categories, and is outperformed by the latter in the $\ell_2$-norm categories with larger $\lambda$ values.
(iii) Algorithm 1 increases its execution time as the threshold distance value ($\lambda$) increases. Indeed more CPU time is spent for instances with large values of $\lambda$. This is due to the fact that when $\lambda$ is large, more intersections of convex regions are likely to exist in the instance, and consequently, more subproblems have to be solved in order to optimize (3). Couenne execution times also appear to be influenced by $\lambda$ augmentation, though they increase in a slower rate than Algorithm 1. Baron seems to be the algorithm affected the least by $\lambda$, and for this reason, likely the best option when $\lambda$ is large and there are too many subproblems for Algorithm 1 to solve.
(iv) Algorithm 1 outperforms Couenne and Baron in all $\ell_1$-norm categories. Particularly for category $\ell_1$_1000_$10^{-9}$, Algorithm 1 is approximately 2000 times faster than Couenne and 20000 times faster than Baron.
(v) Algorithm 1 outperfoms Couenne and Baron in the $\ell_2$-norm categories with 1000 points, but is outperfomed by them in the $\ell_2$-norm categories with 100 points. As observed in (ii), CPU times spent by Algorithm 1 depend largely on the number of subproblems to be solved. At first glance, this could not be explained only by the values shown in Table 3. For example, the average number of subproblems solved in category $\ell_2$_1000_$10^{-9}$ is 14459.3 while the average number of subproblems solved in category $\ell_2$_100_$10^{-9}$ is 4878.6. However, Algorithm 1 is, in average, approximately 26 faster in the first category. The reason for this fact lies on the number of subproblems of type (9) solved within Algorithm 1: in average, 344.5 for the instances in category $\ell_2$_100_$10^{-9}$, and 0.4 for those in category $\ell_2$_100_$10^{-9}$.

Our next set of experiments focus on how parametes $L$ and $U$ influence the performance of Algorithm 1. The number of subproblems solved within Algorithm 1 is directly related to the lower bound value $L$ in (3). Subproblems with size smaller than $L$ are not even considered for resolution since their associated region cannot lodge the optimal facility location. Hence, as the value of $L$ increases, less CPU time is spent by Algorithm 1. As subproblem resolution becomes more complex, the value of parameter $L$ turns out to be even more weighty for Algorithm 1 performance.

Tables 4 and 5 present computing times (in seconds) spent by Algorithm 1, Couenne and Baron for solving 20 instances with 50 random points uniformly distributed in a $1000 \times 1000$ square. The threshold distance values are made the same for all points; $\lambda_i = \lambda = 300, \forall i = 1, \ldots, n$ for the ten $\ell_1$-norm instances and $\lambda_i = \lambda = 250, \forall i = 1, \ldots, n$ for the other ten $\ell_2$-norm instances.

The tables report, varying only the lower bound value $L$ (in this set of experiments $U = +\infty$), the average and the standard deviation of the CPU time spent by Algorithm 1, Couenne and Baron on solving the generated instances. Regarding Algorithm 1, the tables also report the average and the standard deviation of the number of subproblems solved.

**Table 4** CPU times (in seconds) of Algorithm 1, Couenne and Baron as a function of lower bound in ten $\ell_1$-norm distinct instances with 50 random points

| | Algorithm 1 | | | | Couenne | | Baron | |
|---|---|---|---|---|---|---|---|---|
| $L$ | CPU time(s) | std.dev. | # subs | std.dev. | CPU time(s) | std.dev. | CPU time(s) | std.dev. |
| 5 | 8.57 | 1.67 | 3651.9 | 399.47 | 45.14 | 15.82 | 116.481 | 119.90 |
| 10 | 3.29 | 1.32 | 1374.0 | 544.96 | 41.87 | 7.64 | 125.83 | 120.57 |
| 15 | 0.35 | 0.73 | 133.2 | 263.26 | 26.08 | 9.91 | 35.86 | 16.83 |

**Table 5** CPU times (in seconds) of Algorithm 1, Couenne and Baron as a function of lower bound in ten $\ell_2$-norm distinct instances with 50 random points

| | Algorithm 1 | | | | Couenne | | Baron | |
|---|---|---|---|---|---|---|---|---|
| $L$ | CPU time(s) | std.dev. | # subs | std.dev. | CPU time(s) | std.dev. | CPU time(s) | std.dev. |
| 5 | 392.80 | 116.01 | 3998.0 | 381.78 | 319.61 | 295.17 | 33.49 | 12.99 |
| 10 | 84.67 | 55.96 | 1799.0 | 586.56 | 264.92 | 299.82 | 33.85 | 12.89 |
| 15 | 5.38 | 10.60 | 224.6 | 313.11 | 105.25 | 224.00 | 30.10 | 8.39 |

We notice from Tables 4 and 5 that Algorithm 1 improves considerably its performance as $L$ augments and the number of subproblems decreases. The same is also observed for Couenne, though in a smaller rate. For this experiment, Algorithm 1 is clearly the best approach for the $\ell_1$ instances, but became the best option for the $\ell_2$ instances only after $L$ was increased to 15 and the number of subproblems decreased enough. The reason for this performance difference while solving $\ell_1$ and $\ell_2$-norm instances relies on the complexity of the subproblems of type (8): linear for $\ell_1$-norm instances and convex non-linear for $\ell_2$-norm ones. Furthermore, Baron does not appear to be influenced by $L$ as much as the other algorithms are. For the $\ell_2$-norm instances with $L = 5, 10$, Baron outperforms Algorithm 1 and Couenne.

The last set of experiments addresses the influence of parameter $U$ on Algorithm 1 performance. For that purpose, the algorithm was used to solve the same 20 instances of the last experiment, but this time for different combinations of $L$ and $U$ values. Tables 6 and 7 report, the average computing times, and its standard deviations, spent by Algorithm 1. Moreover, the average and standard deviation of the number of subproblems of type (8) and (9) solved within the algorithm are presented. The tables also present the average and

the standard deviation of the CPU times spent by Couenne and Baron on solving (3) for the 10 instances of each norm.

From these last results, we notice that the performance of Algorithm 1 is improved as $U$ augments. Whenever the size of a subproblem is smaller or equal to $U$, a subproblem of type (9) is solved instead of a more difficult subproblem of type (8). For example, when $U$ increases from 15 to 20 in Table 7, the average number of subproblems of type (8) that becomes solvable by model (9) within Algorithm 1 is $135.1 - 4.1 = 131.1$. Consequently, the average CPU time spent by the algorithm drops from 145.49 to 86.55 seconds.

Furthermore, we observe from Table 7 that Algorithm 1 always outperforms Couenne, but is outperformed by Baron with parameters $L = 10, U = 15$ and $L = 10, U = 20$. Indeed if the total number of subproblems is large, the decomposition approach of Algorithm 1 may not be the best strategy to choose. Although the subproblems solved by Algorithm 1 are smaller and require less computing time, they may be too numerous so that the total time spent on solving all them is greater than solving the original problem (3) directly. This fact was not observed in the results of Table 6 because the $\ell_1$-subproblems are easier to solve than those in the $\ell_2$ norm.

Finally, it is worthy mentioning that standard deviations of computing times and the number of subproblems for Algorithm 1 and Couenne are due to the presence of instances with different hardness degrees. In particular, one instance was noticed to be much harder than the others for Algorithm 1 and Couenne. If the same was removed from the experiment with $\ell_2$-norm instances, the average CPU time of Algorithm 1 would be 85.61 seconds with parameters $L = 10$ and $L = 15$ (std.dev. 57.34), 68.49 seconds with $L = 10$ and $U = 20$ (std.dev. 54.56), and 1.87 seconds with $L = 15$ and $U = 20$ (std.dev. 2.08). For Couenne, the corresponding average CPU times would be: 221.39 (std.dev. 161.95), 231.36 (std.dev 192.00), and 28.43 seconds (std.dev. 24.94). This demonstrates that the points distribution in the space plays a key role in the performance of our algorithm. Indeed it directly influences the number and the type of the subproblems to be solved.

### 4.1 Application to a real-life problem

We report computational results obtained on a real-life problem provided by the Natal Police Department in Brazil. The data consists of 586 sites in Natal, Brazil where criminal activities were recorded in the period from 01/01/09 to 09/30/2009. It contains, for each site, its coordinates in UTM scale, which are approximated to the Euclidean space, and the number of recorded crimes in the analyzed period, which are used to weight the demand of that site for a police station close-by. The objective for this problem is to locate a new police station close to where the police demand is high, but also respecting lower capacity constraints which are useful to better distribute the police coverage over the city.

**Table 6** CPU times (in seconds) of Algorithm 1, Couenne and Baron as a function of lower and upper bounds in ten $\ell_1$-norm distinct instances with 50 random points

| | | Algorithm 1 | | | | | | Couenne | | Baron | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $L$ | $U$ | CPU time(s) | std.dev. | # subs(6) | std.dev. | # subs(7) | std.dev. | CPU time(s) | std.dev. | CPU time(s) | std.dev. |
| 10 | 15 | 2.32 | 2.47 | 1374.0 | 544.96 | 76.2 | 193.29 | 63.19 | 24.69 | 63.19 | 24.69 |
| 10 | 20 | 1.58 | 0.68 | 1374.0 | 544.96 | 0.8 | 2.52 | 47.22 | 17.27 | 130.50 | 114.97 |
| 15 | 20 | 0.18 | 0.37 | 133.2 | 263.26 | 0.8 | 2.52 | 27.92 | 10.45 | 40.26 | 19.89 |

**Table 7** CPU times (in seconds) of Algorithm 1, Couenne and Baron as a function of lower and upper bounds in ten $\ell_2$-norm distinct instances with 50 random points

| | | Algorithm 1 | | | | | | Couenne | | Baron | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $L$ | $U$ | CPU time(s) | std.dev. | # subs(6) | std.dev. | # subs(7) | std.dev. | CPU time(s) | std.dev. | CPU time(s) | std.dev. |
| 10 | 15 | 145.49 | 166.60 | 1799.0 | 586.56 | 135.1 | 254.10 | 658.99 | 1313.73 | 35.18 | 13.02 |
| 10 | 20 | 86.55 | 59.07 | 1799.0 | 586.56 | 4.1 | 12.96 | 431.68 | 579.25 | 35.18 | 9.27 |
| 15 | 20 | 7.13 | 16.06 | 224.6 | 313.11 | 4.1 | 12.96 | 136.64 | 332.98 | 26.83 | 9.74 |

Our tests used $\lambda = 1$ kilometer (estimated in [17]) in $\ell_1$-norm for all sites. Besides, no upper bound capacity $U$ is used due to the nature of the application. Two different scenarios were tested by varying the value of $L$: scenario $A$ uses $L = 10$, and scenario $B$ uses $L = 25$. We solved the corresponding LDMPSC problem using Algorithm 1, Couenne and Baron. The optimal solutions in scenarios $A$ and $B$ are shown in Figure 2, where the problem is geographically represented.

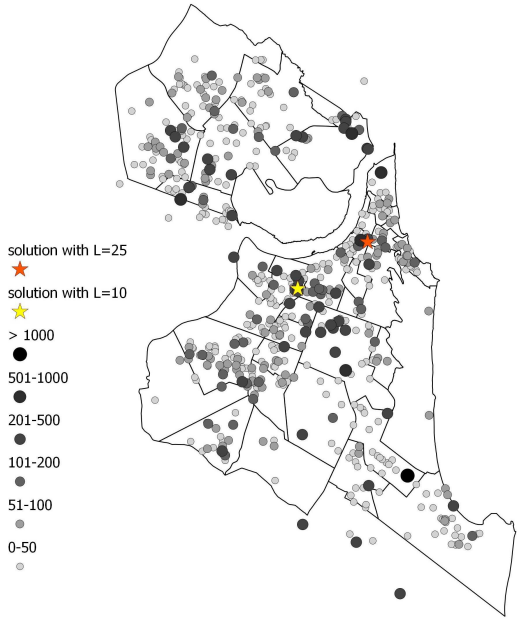


**Fig. 2** Geographic distribution of criminal records in the city of Natal, Brazil from 01/01/2009 to 09/30/2009. The size of the plotted points is proportional to the number of criminal records of the corresponding site. Thus, the smallest points represent sites where the number of criminal records was between 0 and 50, and the largest points represent sites where the number of records was greater than 1000. The optimal locations for positioning a police unity in scenarios $A$ and $B$ are represented by a yellow and a red star, respectively.

Algorithm 1 outperforms Couenne and Baron in both scenarios. Algorithm 1 solves scenario $A$ in 102.30 seconds, Couenne is not able to solve it within 1 hour, and Baron solves it in 1614.38. Regarding scenario $B$, Algorithm 1 solves it in 10.72 seconds, Couenne is again not able to solve the problem within 1 hour, and Baron solves it in 1715.37.

## 5 Conclusions

The introduction of side constraints while locating a facility in the plane with limited distances may serve to justify its installation or to describe service

limitations. Our work extends that of Drezner, Mehrez and Wesolowsky [13], adapting it to the presence of side constraints. This approach leads to subproblems having products of the continuous location variable with assignment binary variables. The subproblem model is then reformulated in order to ease its resolution. In summary, the performance of the presented algorithm is influenced by:

(i) the complexity of the subproblems - e.g. subproblem (8) is a MILP if $\ell_1$-distances are used, and a non-differentiable convex MINLP for $\ell_2$-distances;
(ii) the number of subproblems to be solved - this is linked with the threshold distances values;
(iii) the lower bound of service - this allows to disregard the resolution of some subproblems;
(iv) the upper bound of service - easier subproblems to solve due to removing integer decision variables.

Finally, it is important to remark that the proposed algorithm can be straightforwardly parallelized, since no dependence exists among the solved subproblems.

**Acknowledgements** We are thankful to André Morais Gurgel and the Natal Police Department for providing us the data for the real instance. We also thank two anonymous referees for insightful remarks.

## References

1. Aloise, D., Hansen P., Liberti, L.: An improved column generation algorithm for minimum sum-of-squares clustering, Mathematical Programming, 131, 195–220 (2012).
2. Belotti, P., Lee J., Liberti L., Margot F., Wächter A.: Branching and bounds tightening techniques for non-convex MINLP, Optimization Methods and Software, 24, 597–634 (2009).
3. Berman, O., Drezner, Z., Krass, D.: Generalized coverage: New developments in covering location models, Computers and Operations Research, 37, 1675–1687 (2010).
4. Brimberg, J., Chen, R., Chen D.: Accelerating convergence in the Fermat-Weber location problem. Operations Research Letters, 22, 151–157 (1998).
5. Bonami, P., Lee J.: BONMIN user's manual. Technical report, IBM Corporation (2007).
6. P. Bonami, L. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, A. Wächter: An algorithmic framework for convex mixed integer nonlinear programs, Discrete Optimization, 5, 186–204 (2008).
7. Boyd, S., Vandenberghe, L.: Convex Optimization, Cambridge University Press, Cambridge 2004.
8. Church R., ReVelle C.: The maximal covering location problem. Papers of the Regional Science Association, 32, 101–118 (1974).
9. Church, R., Roberts K.L.: Generalized coverage models and public facility location. Papers of the Regional Science Association, 53, 117–135 (1983).
10. Czyzyk, J., Mesnier, M., Moré, J.: The NEOS Server. IEEE Journal on Computational Science and Engineering, 5, 68–75 (1998).
11. de Berg, M., van Krefeld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications, Springer (1997).
12. Drezner Z., Wesolowsky, G.O.: A maximin location problem with maximum distance constraints, AIIE Transactions, 12, 249–252 (1980).

13. Drezner Z., Mehrez A., Wesolowsky, G.O.: The facility location problem with limited distances. Transportation Science, 25, 183–187 (1991).
14. Drezner, Z., Hamacher H.W.: Facility Location: Applications and Theory. Springer (2004).
15. Drezner, Z., Wesolowsky, G.O., Drezner, T.: The gradual covering problem. Naval Research Logistics, 51, 841–855 (2004).
16. Fekete, S.P., Mitchell, J.S.B, Beurer, K.: On the continuous Fermat-Weber problem. Operations Research, 53, 61–76 (2005).
17. A.M. Gurgel: Melhoria da segurança pública: Uma proposta para a alocação de unidades policiais utilizando o modelo das p-medianas e do caixeiro viajante. M.Sc. dissertation. Universidade Federal do Rio Grande do Norte (2010).
18. Hansen, P., Mladenović, N., Taillard, É.: Heuristic solution of the multisource Weber problem as a image-median problem". Operations Research Letters, 22, 55–62 (1998).
19. Liberti, L.: Writing global optimization software, in Liberti, L., Maculan, N. (eds.), Global Optimization: from Theory to Implementation, 211-262, Springer, Berlin (2006).
20. Liberti, L.: Reformulations in Mathematical Programming: Definitions and systematics. RAIRO-RO, 43, 55–86 (2009).
21. Pirkul, H., Schilling, D.A.: The maximal covering location problem with capacities on total workload. Management Science, 37, 233–248 (1991).
22. Schilling, D.A., Jayaraman V., Barkhi, R.: A review of covering problems in facility location. Location Science, 1, 25–55 (1993).
23. Smith, E., Pantelides, C.: A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs, Computers & Chemical Engineering, 23, 457-478 (1999).
24. Smith, H.K., Laporte, G., Harper, P.R.: Locational analysis: highlights of growth to maturity. Journal of the Operational Research Society, 60, 140–148 (2009).
25. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Mathematical Programming, 103, 225–249 (2005).
26. Wesolowsky, G.O.: The Weber problem: history and perspectives. Location Science, 1, 5–23 (1993).