RAIRO-Oper. Res. 49 (2015) 589–600
DOI: 10.1051/ro/2014058

RAIRO Operations Research
www.rairo-ro.org

A CONVEX HULL ALGORITHM FOR SOLVING A LOCATION PROBLEM *

Nguyen Kieu Linh[1] and Le Dung Muu[2]

Abstract. An important problem in distance geometry is of determining the position of an unknown point in a given convex set such that its longest distance to a set of finite number of points is shortest. In this paper we present an algorithm based on subgradient method and convex hull computation for solving this problem. A recent improvement of Quickhull algorithm for computing the convex hull of a finite set of planar points is applied to fasten up the computations in our numerical experiments.

Keywords. Location problem, distance geometry, convex hull, Quickhull algorithm, subgradient method.

Mathematics Subject Classification. 52A20, 90C27.

1. Introduction

The problem being considered in this paper is to find a point x in a given closed convex set $D \subset \mathbb{R}^d$ (most often $d \in \{2, 3\}$) such that the farthest distance from x to the points of a finite set $C \subset \mathbb{R}^d$ is shortest.

This single facility location problem appears in some issues encountered in economics, logistics, infrastructure construction, computer science, and other fields. Often the point x represents the location of a facility (manufacturing plant, hospital, school, wireless station, etc.) to be constructed to serve the users (warehouses, patients, pupils, internet users, etc.) located at the points of the set C, and D represents the restricted region for constructing the facility. We wish to locate the facility for serving the farthest user as well as possible. The objective function in

Received February 23, 2014. Accepted November 12, 2014.

* *This paper is supported in part by the NAFOSTED, Vietnam.*

[1] Thai Nguyen University, Tan Thinh, Thai Nguyen, Vietnam. nguyenkieulinhk4@gmail.com

[2] Institute of Mathematics VAST, 18 Hoang Quoc Viet, 10307 Hanoi, Vietnam. ldmuu@math.ac.vn

Article published by EDP Sciences

the problem is then the farthest distance, which needs to be minimized, from the facility to the users.

Some variants of the location problem have been studied throughout the existing literature. For the same input sets C and D, there may be different objective functions corresponding to different goals of the problems. Hansen *et al.* [7] and Plastria [11] first proposed Branch and Bound methods, such as Big Square Small Square for certain constrained location problems with minisum and minimax objective functions. Some kinds of single and multi-facility location problems with nonconvex objective functions can be found in Kon *et al.* [8], Tuy [14] and in the references cited therein.

The single facility location problem considered in this paper can be modeled as a nonsmooth convex optimization problem with a strongly convex objective function. We propose a subgradient algorithm for solving the resulting nonsmooth optimization problem. Often in practice the cardinality of the set C is a large number. Fortunately, by convexity property of the distance function, in the location problem in question we can replace C by the set V_C consisting of vertices of convex hull of C. In many usual cases, the cardinality of V_C is much less than that of C. For example, when the set C consists of n i.i.d. (independent and identically distributed) random points in $\mathbb{R}^d$ for some general distributions (*e.g.* Gaussian distribution, uniform distribution), Bentley *et al.* [3] showed that the cardinality of V_C is $O(\ln^{d-1} n)$. Thus, finding the vertices of convex hull of C is an important pre-processing step when solving our location problem. To do this step, in our numerical experiments for this problem, we apply a recent improvement of Quickhull algorithm described in Dung and Linh [6].

The paper is organized as follows. After the introduction, Section 2 reviews some concepts and results will be used in the paper. In the next section we model the location problem as a convex optimization problem and investigate its properties. The fourth section is devoted to present the algorithm for solving the problem and to prove its convergence. Numerical experiments are reported in the last section.

2. Preliminaries

For convenience of the readers, in this section we review some concepts and essential results that will be used in the next sections.

Definition 2.1 ([9], p. 60)**.** A function $f : X \to \mathbb{R}$ is said to be *strongly convex* with modulus $\rho > 0$ on the convex set $X \subset \mathbb{R}^d$, shortly ρ-strongly convex, if for every $x, y \in X$ and $\lambda \in [0, 1]$ one has

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) - \frac{\rho}{2}\lambda(1-\lambda)\|x-y\|^2.$$

It is easy to verify the following lemma.

Lemma 2.2.

(*i*) *For any fixed vector $a \in \mathbb{R}^d$, the function $f(x) := \|x-a\|^2$ is strongly convex with modulus $\rho = 2$ on the whole space $\mathbb{R}^d$.*

(*ii*) *Let J be a nonempty finite index set and g_j be strongly convex function on the convex set X with modulus ρ_j for every $j \in J$. Then the function $g = \max_{j\in J} g_j$ is strongly convex on X with modulus $\rho = \min_{j\in J} \rho_j$.*

Definition 2.3 ([12], p. 214)**.** Let $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ be a convex function. A vector v is called a *subgradient* of f at x if for all $y \in \mathbb{R}^d$ one has

$$\langle v, y - x\rangle \leq f(y) - f(x).$$

The set of subgradients of f at x is called *the subdifferential* of f at x, and is denoted by $\partial f(x)$. The function f is called *subdifferentiable at x* if $\partial f(x) \neq \emptyset$. f is called *subdifferentiable* if it is subdifferentiable at every $x \in \mathrm{dom} f$, where

$$\mathrm{dom} f = \{x \in \mathbb{R}^d : f(x) < +\infty\}.$$

Definition 2.4 ([12], p. 10)**.** Let P be a set of k points in $\mathbb{R}^d$. The *convex hull* of P, denoted $\mathrm{conv}(P)$, is defined as follows

$$\mathrm{conv} P := \left\{ \sum_{i=1}^{k} \theta_i x_i \mid x_i \in P, \theta_i \geq 0, i = 1, \ldots, k, \sum_{i=1}^{k} \theta_i = 1 \right\}.$$

Definition 2.5 ([12], p. 162))**.** Let X be a convex set in $\mathbb{R}^d$. A point $x \in X$ is called an *extreme point* of X if there do not exist $y, z \in X, y \neq z$ such that $x = (1 - \lambda)y + \lambda z$ with $0 < \lambda < 1$.

When X is a polyhedral convex set, its extreme points are also called vertices. For a finite set of points P, we will refer to the convex hull problem as the problem of finding all vertices of the convex hull of P.

Lemma 2.6 ([12], p. 215)**.** *Let $f_i : \mathbb{R}^d \to \mathbb{R}$ $(i \in I := \{1, \ldots, m\})$ be convex functions on $\mathbb{R}^d$ and $f(x) := \max\{f_i(x) | i \in I\}$. Then f is a subdifferentiable convex function on $\mathbb{R}^n$ and*

$$\partial f(x) = \mathrm{conv}\left(\cup_{i\in I(x)} \partial f_i(\mathrm{x})\right),$$

where $I(x) := \{i \in I | f(x) = f_i(x)\}$.

Definition 2.7 ([4], p. 397)**.** Let A be a nonempty closed convex subset of $\mathbb{R}^d$ and x be any point in $\mathbb{R}^d$. A point $\bar{x} \in A$ is called the *Euclidean (or metric) projection of x onto A* if

$$\|x - \bar{x}\| \leq \|x - y\| \quad \forall y \in A.$$

We will denote the projection of x onto A by $P_A(x)$. It is well-known (see *e.g.* [4]) that for every x the Euclidean projection $P_A(x)$ uniquely exists. It holds that

$$\bar{x} = P_A(x) \quad \Leftrightarrow \quad \bar{x} \in A \text{ and } \langle x - \bar{x}, y - \bar{x}\rangle \leq 0 \ \forall y \in A. \tag{2.1}$$

We need the following lemma for proving the convergence of the algorithm described in the next section.

Lemma 2.8 ([15]). *Suppose that $\{\xi_k\}$ is a sequence of positive numbers satisfying the condition*

$$\xi_{k+1} \leq \xi_k + \beta_k \ \forall k \in \mathbb{N},$$

where $\beta_k \geq 0$ and $\sum_{k=0}^{\infty} \beta_k < +\infty$. Then the sequence $\{\xi_k\}$ is convergent.

3. Mathematical form and its properties

As we have mentioned in the introduction, to solve the problem we need to find a point x in a given closed convex set D such that the longest Euclidean distance from x to the points in a given finite set C is shortest.

The distance from a point x to a point y is defined by $\|x - y\|$. Let $d(x, C) := \max_{y \in C} \|x - y\|^2$, then the problem to be solved takes the following mathematical form

$$\min_{x \in D} \ d(x, C). \tag{P}$$

Lemma 3.1. *Let V_C denote the set of vertices of* conv*(C). Then one has*

(*i*) $V_C \subseteq C$;
(*ii*) $d(x, C) = \max\{\|x - y\|^2 \mid y \in V_C\}$.

Proof. The assertion (i) follows directly from the definition of the set V_C. To see (ii) we observe that

$$d(x, C) = \max_{y \in C} \|x - y\|^2 = \max_{y \in \text{conv(C)}} \|x - y\|^2 = \max_{y \in V_C} \|x - y\|^2, \tag{3.1}$$

where the last equality comes from the fact that the maximum of a convex function over a convex set attains at some of its extreme points ([12], Thm. 32.2). □

Lemma 3.2. *Let $v^1, \ldots, v^m$ be the elements of V_C and $d_j(x, C) := \|x - v^j\|^2$ for each $j \in J := \{1, \ldots, m\}$. Then one has*

(*i*) *$d(x, C)$ is strongly convex with modulus* 2;
(*ii*) *$\partial d(\cdot, C)(x) = \text{conv}\left(\cup_{j \in J(x)} \partial d_j(\cdot, C)(x)\right)$, where $\partial d_j(\cdot, C)(x)$ is the subdifferential of the convex function $d_j(\cdot, C)$ at x and $J(x) = \{j \in J \mid d(x, C) = d_j(x, C)\}$.*

Proof. It follows from Lemma 2.2(i) that

$$d(x, C) = \max_{j \in J} \|x - v^j\|^2 = \max_{j \in J} d_j(x, C). \tag{3.2}$$

By Lemma 2.2(i), for each $j \in J$ the function $d_j(x, C) = \|x - v^j\|^2$ is strongly convex with modulus 2. Hence the assertion (i) follows from Lemma 2.2(ii), while the assertion (ii) follows from (3.2) and Lemma 2.6. □

From computational point of view, evaluating the objective function $d(x, C)$ at every point of C will be very costly if the number of the points in C is too large. Fortunately, thanks to Lemma 3.2(i), for minimizing the function $d(x, C)$ one needs to consider only the vertices of the convex hull of C.

4. The Algorithm and Its Convergence Analysis

By Lemma 3.1(ii) the problem being solved can take the form

$$\min_{x\in D} d(x,C) = \min_{x\in D}\max_{v\in V_C} \|x-v\|^2. \tag{P}$$

Suppose that D is a closed convex (not necessarily bounded) set. Since $d(x,C)$ is strongly convex on D, problem (P) always admits a unique optimal solution ([9], Thm. 2.2.6). The following algorithm can be considered as a modification of the subgradient algorithm presented in [13] for unconstrained nonsmooth convex optimization problem.

Algorithm 4.1. *Initialization. Select $x^0 \in D$, fix a parameter $\rho > 0$ and choose a sequence $\{\beta_k\}$ of positive numbers satisfying the condition*

$$\sum_{k=0}^{\infty} \beta_k = +\infty, \ \sum_{k=0}^{\infty} \beta_k^2 < \infty. \tag{4.1}$$

Let $k := 0$.
Step 1. Find $v^k \in V_C$ such that

$$v^k \in \arg\max\{\|x^k - v\|^2 : v \in V_C\}.$$

Step 2. Take $g^k := 2(v^k - x^k)$, i.e. the gradient of $\|x^k - v\|^2$ at v^k.
Case (2a): If $g^k = 0$, then terminate: x^k is the optimal solution to (P).
Case (2b): If $g^k \neq 0$, compute

$$\alpha_k := \frac{\beta_k}{\max\{\rho, \|g^k\|\}},$$

and

$$x^{k+1} := P_D(x^k - \alpha_k g^k),$$

where P_D stands for the Euclidean projection operator onto D.
Step 3. If $x^{k+1} = x^k$, then terminate: x^k is the optimal solution to (P). Otherwise, set $k := k+1$ and go back to Step 1.

Theorem 4.2.

(i) If the algorithm 4.1 terminates at some iteration k, then x^k is the optimal solution to problem (P).
(ii) If the algorithm 4.1 does not terminate, then the sequence $\{x^k\}$ converges to the solution x^* to problem (P).

Proof.

(i) If the algorithm 4.1 terminates at iteration k, then either $g^k = 0$ or $x^k = P_D(x^k - \alpha_k g^k)$.
In the first case, $g^k = 0 \in \partial d(x^k, C)$, by the definition of subgradient, it implies that

$$\langle 0, x - x^k\rangle + d\left(x^k, C\right) \leq d(x,C) \quad \forall x \in D.$$

Hence

$$d\left(x^k, C\right) \leq d(x, C) \quad \forall x \in D, \tag{4.2}$$

which means that x^k minimizes the function $d(x, C)$ over D.
In the second case, $x^k = x^{k+1} = P_D(x^k - \alpha_k g^k)$. Then using property (2.1) of the metric projection we obtain

$$\begin{aligned}\left\langle \left(x^k - \alpha_k g^k\right) - x^k, x - x^k \right\rangle \leq 0 &\Leftrightarrow -\alpha_k \left\langle g^k, x - x^k \right\rangle \leq 0 \\ &\Leftrightarrow \left\langle g^k, x - x^k \right\rangle \geq 0. \end{aligned} \tag{4.3}$$

Since $g^k \in \partial d(x^k, C)$, one has

$$\left\langle g^k, x - x^k \right\rangle + d(x^k, C) \leq d(x, C).$$

Combining this inequality with (4.3) yields $d(x^k, C) \leq d(x, C)$ for every $x \in D$. Hence x^k is the optimal solution to (P).

(ii) Now suppose that the algorithm does not terminate. Let x^* be the solution of problem (P). We prove the assertion (ii) throughout several claims. □

Claim 4.3. *One has*

$$\left\|x^{k+1} - x^k\right\| \leq \beta_k \ \forall k \in \mathbb{N}.$$

Proof. According to the definition of α_k we have

$$\alpha_k \left\|g^k\right\| = \frac{\beta_k \left\|g^k\right\|}{\max\{\rho, \|g^k\|\}} \leq \beta_k.$$

Since $x^{k+1} = P_D(x^k - \alpha_k g^k)$, using again the property (2.1) of the metric projection, we have

$$\left\langle x^k - \alpha_k g^k - x^{k+1}, x - x^{k+1} \right\rangle \leq 0 \ \forall x \in D. \tag{4.4}$$

Replacing x by x^k yields

$$\begin{aligned}\left\|x^k - x^{k+1}\right\|^2 &\leq \left\langle \alpha_k g^k, x^k - x^{k+1} \right\rangle \\ &\leq \alpha_k \left\|g^k\right\| \left\|x^k - x^{k+1}\right\| \\ &\leq \beta_k \left\|x^k - x^{k+1}\right\|, \end{aligned} \tag{4.5}$$

which implies $\|x^{k+1} - x^k\| \leq \beta_k$.

Claim 4.4. *For every k, the sequence $\{\|x^k - x^*\|^2\}$ converges.*

Proof. Using the definition of the Euclidean norm we can write

$$\left\|x^k - x^*\right\|^2 = \left\|x^{k+1} - x^k\right\|^2 - 2\left\langle x^k - x^{k+1}, x^* - x^{k+1} \right\rangle + \left\|x^{k+1} - x^*\right\|^2.$$

Thus

$$\left\|x^{k+1} - x^*\right\|^2 = \left\|x^k - x^*\right\|^2 - \left\|x^{k+1} - x^k\right\|^2 + 2\left\langle x^k - x^{k+1}, x^* - x^{k+1} \right\rangle. \tag{4.6}$$

Note that from (4.5) we have

$$\langle \alpha_k g^k, x^k - x^{k+1} \rangle \leq \beta_k \left\| x^k - x^{k+1} \right\| \leq \beta_k^2. \tag{4.7}$$

Then from (4.6) and (4.7), it follows that

$$\begin{aligned} \left\| x^{k+1} - x^* \right\|^2 &\leq \left\| x^k - x^* \right\|^2 - \left\| x^{k+1} - x^k \right\|^2 + 2 \langle \alpha_k g^k, x^* - x^{k+1} \rangle \\ &\leq \left\| x^k - x^* \right\|^2 + 2 \langle \alpha_k g^k, x^* - x^{k+1} \rangle \\ &= \left\| x^k - x^* \right\|^2 + 2 \langle \alpha_k g^k, x^* - x^k \rangle + 2 \langle \alpha_k g^k, x^k - x^{k+1} \rangle \\ &\leq \left\| x^k - x^* \right\|^2 + 2\alpha_k \langle g^k, x^* - x^k \rangle + 2\beta_k^2. \end{aligned} \tag{4.8}$$

Since $g^k \in \partial d(x^k, C)$, we have

$$\langle g^k, x^* - x^k \rangle \leq d(x^*, C) - d(x^k, C). \tag{4.9}$$

Substituting (4.9) into (4.8) to obtain

$$\left\| x^{k+1} - x^* \right\|^2 \leq \left\| x^k - x^* \right\|^2 + 2\alpha_k \left(d\left(x^*, C\right) - d\left(x^k, C\right) \right) + 2\beta_k^2. \tag{4.10}$$

Since x^* is an optimal solution, $d(x^k, C) \geq d(x^*, C)$, and therefore from (4.10) we have

$$\left\| x^{k+1} - x^* \right\|^2 \leq \left\| x^k - x^* \right\|^2 + 2\beta_k^2,$$

from which, by the assumption $\sum_{k=0}^{\infty} \beta_k^2 < +\infty$, it follows from Lemma 2.8 that the sequence $\{\|x^k - x^*\|^2\}$ is convergent. □

Claim 4.5. *One has*

$$\limsup_{k \to +\infty} (d(x^k, C) - d(x^*, C)) = 0. \tag{4.11}$$

Proof. From 4.10, we can write

$$0 \leq 2\alpha_k \left(d\left(x^k, C\right) - d\left(x^*, C\right) \right) \leq \left\| x^k - x^* \right\|^2 - \left\| x^{k+1} - x^* \right\|^2 + 2\beta_k^2. \tag{4.12}$$

Summing up both sides of the inequality above, we obtain

$$\begin{aligned} 0 \leq 2 \sum_{k=0}^{m} \alpha_k \left(d\left(x^k, C\right) - d\left(x^*, C\right) \right) &\leq \left\| x^0 - x^* \right\|^2 - \left\| x^{m+1} - x^* \right\|^2 + 2 \sum_{k=0}^{m} \beta_k^2 \\ &\leq \left\| x^0 - x^* \right\|^2 + 2 \sum_{k=0}^{m} \beta_k^2. \end{aligned}$$

Letting $m \to +\infty$ we get

$$0 \leq 2 \sum_{k=0}^{+\infty} \alpha_k \left(d\left(x^k, C\right) - d\left(x^*, C\right) \right) \leq \left\| x^0 - x^* \right\|^2 + 2 \sum_{k=0}^{+\infty} \beta_k^2. \tag{4.13}$$

Since $\sum_{k=0}^{+\infty} \beta_k^2 < +\infty$, we have

$$\sum_{k=0}^{+\infty} \alpha_k \left(d\left(x^k, C\right) - d\left(x^*, C\right)\right) < +\infty. \tag{4.14}$$

On the other hand, since the sequence $\{x^k\}$ is bounded, the sequence $\{g^k\}$ is bounded too ([12], Thm. 24.7, p. 237). Thus, there exists $L > 0$ such that $\|g^k\| \leq L < \infty$ for every $k \in \mathbb{N}$. Let $L_0 := \max\{\rho, L\}$, then, by definition of α_k, we have

$$\alpha_k = \frac{\beta_k}{\max\{\rho, \|g^k\|\}} \geq \frac{\beta_k}{L_0}, \tag{4.15}$$

which together with (4.14) implies

$$\frac{1}{L_0} \sum_{k=0}^{+\infty} \beta_k \left(d\left(x^k, C\right) - d\left(x^*, C\right)\right) \leq \sum_{k=0}^{+\infty} \alpha_k \left(d\left(x^k, C\right) - d\left(x^*, C\right)\right) < +\infty. \tag{4.16}$$

Since $\sum_{k=0}^{+\infty} \beta_k = +\infty$, we can deduce that

$$\limsup_{k \to +\infty} \left(d\left(x^k, C\right) - d\left(x^*, C\right)\right) = 0. \tag{4.17}$$

□

Now using the just proved claims, we can prove assertion (ii) of the Theorem 4.2. In fact, according to the definition of lim sup, there exists a subsequence $\left\{x^{k_j}\right\}$ of the sequence $\{x^k\}$ such that

$$\lim_{j \to +\infty} \left(d\left(x^{k_j}, C\right) - d\left(x^*, C\right)\right) = \limsup_{k \to +\infty} \left(d\left(x^k, C\right) - d\left(x^*, C\right)\right) = 0.$$

Since $\left\{x^{k_j}\right\}$ is bounded, we may assume that

$$\lim_{j \to +\infty} x^{k_j} = \bar{x}.$$

Then

$$\begin{aligned} d\left(x^*, C\right) - d\left(\bar{x}, C\right) &= \lim_{j \to +\infty} \left(d\left(x^*, C\right) - d\left(x^{k_j}, C\right)\right) \\ &= -\lim_{j \to +\infty} \left(d\left(x^{k_j}, C\right) - d\left(x^*, C\right)\right) \\ &= -\limsup_{k \to +\infty} \left(d\left(x^k, C\right) - d\left(x^*, C\right)\right) \\ &= 0, \end{aligned}$$

which shows that $\bar{x}$ is also an optimal solution. Keeping in mind that x^* is the unique solution of problem (P), we have $x^* = \bar{x}$. Since the sequence $\left\{\|x^k - x^*\|\right\}$ is convergent and the subsequence $\{x^{k_j}\}$ of $\{x^k\}$ converges to x^*, we can write

$$\lim_{k \to +\infty} x^k = \lim_{j \to +\infty} x^{k_j} = x^*.$$

Thus the whole sequence $\{x^k\}$ must converge to x^*. □

Remark 4.6.

(i) As we have seen, if either $g^k = 0$ or $x^{k+1} = x^k$, then x^k is an exact solution. In numerical computation, for an approximate solution, we can terminate the algorithm if either $\|g^k\| \leq \varepsilon$ or $\|x^{k+1} - x^k\| \leq \max\{\|x^k\|, 1\}\epsilon$, where $\epsilon > 0$ is a given tolerance.

(ii) Convergence of the sequence $\{x^k\}$ generated by the above algorithm has been proved in [13], Theorem 7.3 for unconstrained problems. The proof above is quite different from the proof in [13].

(iii) The problem to be solved by the above algorithm is a special case of the minimax problem considered in [7] when the feasible regions is (topologically) connected and the distance is the Euclidean. Thanks to this particular case, the model can be formulated as a nonsmooth convex program that can be much more easily handled than general cases considered in the above mentioned papers. Of course, the just described subgradient algorithm cannot be used for models involving nonconvex objective functions and/or multi-feasible regions. However, the algorithms that used combinatorial and/or global optimization techniques proposed in the above papers, for instance in [7,8,14], are not suitable for our convex model setting.

5. Computational aspects and results

In this section we discuss computational experiments and results on the model setting in two-dimensional spaces.

We suppose that the cardinality of the set C of users is very large (often in practical models) and that the set D where we want to locate the facility is a polyhedral convex set given as

$$D = \left\{x \in \mathbb{R}^2 \mid Ax \leq b\right\},$$

where A is an $m \times 2$ matrix of full rank, b is a vector in $\mathbb{R}^m$.

As presented above, for minimizing the function $d(x, C)$ one needs only to know the vertices of the convex hull of C. There are some effective convex hull algorithms, see *e.g.* Akl *et al.* [1], O'Rourke [10], An [2]. We applied the modification of Quickhull algorithm proposed by Dung and Linh [6] for finding the convex hull of the set C. For convenience of the reader, first we outline the Quickhull algorithm [5,10] and its modification for finding the convex hull on plane.

5.1. The Quickhull algorithm and its modification

It is well-known that the convex hull of a set of finite planar points in is a polygon. Since any point in the convex hull can be expressed as a convex combination of its vertices, for simplicity, in the sequel we use the phrase "convex hull" to mean "the set of vertices of the convex hull".

The Quickhull algorithm [5,10] for finding the convex hull of a set P of n-points in plane processes as follows. The first step is to find two extreme points p and q.

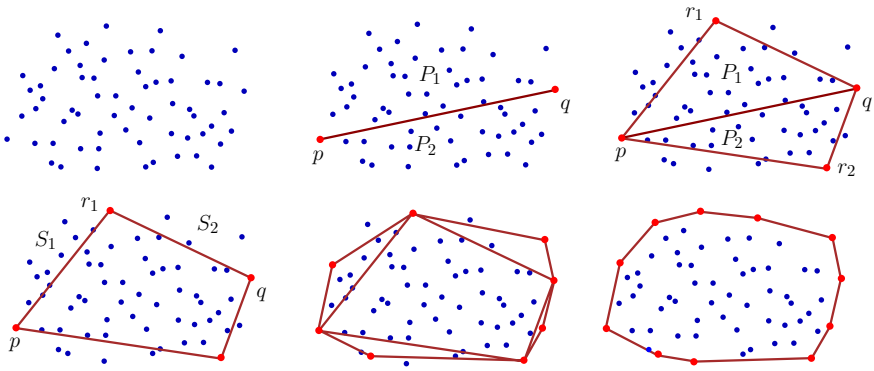

FIGURE 1. The Quickhull algorithm.

The line pq divides $(n-2)$ other points into two groups P_1 and P_2. Then, from P_1 find the farthest point r_1 from the line pq. Three points p, q, and r_1 partition the set P_1 into three subsets S_0, S_1, and S_2, where S_0 consists of the points inside the triangle pr_1q, S_1 consists of the points lying to the left of pr_1, and S_2 consists of the points lying to the left of r_1q. Replace the line segment pq by pr_1 and r_1q and continue recursively the algorithm. The same process is applied for the set P_2 (Fig. 1).

The complexity of the Quickhull algorithm is $O(n \log n)$ in average and $O(n^2)$ in the worst case. But it is similar to the Quicksort algorithm, which runs in practice much faster than in the worst case. In [6], the authors presented three techniques, namely revised Quickhull algorithm, using orienting vectors, preprocessing and problem separation. They combined them with Quickhull algorithm and call it is *the new Quickhull algorithm* (NQH). The new Quickhull algorithm reduces the computational time of the original Quickhull algorithm by a factor of three on average.

5.2. Computational aspects and results

We applied the new Quickhull algorithm and implemented this algorithm in C programming language. Algorithm 4.1 for solving the location problem was implemented with C programming. The programs were executed on a PC Core 2Duo 2×2.0 GHz, RAM 2GB.

We tested the programs on various instances of the set C, which are created randomly inside a square and a circle, which have different sizes, by the random.cpp program. The two examples for the set D are D_1, D_2 defined by

$$A_1 = \begin{bmatrix} 8 & 0 & -1 & -10 & -3 & 14 \\ 5 & 1 & 1 & -4 & -15 & -7 \end{bmatrix}^T,$$

$$b_1 = (103, 11, 17, 142, 155, 133)^T,$$

TABLE 1. Numerical experiments corresponding to the set D_1 (time in s).

$\|C\|$ (1)	$\|V_C\|$ (2)	The average ratio $\frac{\|V_C\|}{\|C\|}$ (3)	Time to find conv(C) (4)	Computational time using NQH (5)	Total time (6) = (4) + (5)	Computational time without using NQH (7)
1000	25		$<10^{-4}$	0.0011	0.0011	0.0021
1000	39		$<10^{-4}$	0.0014	0.0014	0.0030
1000	35	2.76%	$<10^{-4}$	0.0014	0.0014	0.0030
1000	22		$<10^{-4}$	0.0011	0.0011	0.0022
1000	17		$<10^{-4}$	0.0010	0.0010	0.0020
10000	48		0.0010	0.0018	0.0028	0.0127
10000	81		0.0010	0.0077	0.0087	0.0129
10000	100	2.75%	0.0010	0.0111	0.0121	0.0128
10000	27		$<10^{-4}$	0.0013	0.0013	0.0127
10000	19		$<10^{-4}$	0.0011	0.0011	0.0126
100000	94		0.0150	0.0101	0.0251	0.0312
100000	124		0.0150	0.0118	0.0268	0.0313
100000	155	0.414%	0.0161	0.0121	0.0282	0.0314
100000	23		0.0010	0.0015	0.0025	0.0310
100000	18		0.0010	0.0012	0.0022	0.0301

and

$$A_2 = \begin{bmatrix} 1 & 3 & 2 & 1 & 2 & 1 & 1 & -1 & -1 & -1 & -2 & -1 & -4 \\ 0 & 1 & 1 & -1 & 3 & 3 & 4 & 4 & 3 & 2 & 3 & 1 & 3 \\ -3 & -4 & -1 & -2 & -1 & 0 & 1 & 2 & 1 & 2 & 3 & & \\ 1 & -1 & -1 & -5 & -7 & -1 & -4 & -3 & -1 & -1 & -1 & & \end{bmatrix}^T,$$

$$b_2 = (15, 48, 36, 21, 52, 44, 57, 55, 42, 30, 49, 20, 73, 51, 68, 20, 64, 77, 11, 47, 44, 18, 31, 36)^T.$$

In implementing Algorithm 4.1 we set $\varepsilon = 10^{-3}, \alpha_k = \beta_k = \frac{1}{k+1}$. The results of testing the programs on these instances are reported in Tables 1 and 2. Running time of executing the programs is recorded in seconds.

Columns (6) in Tables 1 and 2 contain the time for solving the problem when applying the preprocessing step for finding the convex hull of the set C by using the new Quickhull (shortly NQH), while columns (7) contain the time to solve the problem without using the new Quickhull for finding the convex hull of the set C. Comparing (6) and (7) in both Tables, we realize that exploring the convex property of the distance function is an essential step in solving our location problem, since this step helps us to dramatically reduce the number of points in the set C which need to be considered as the cardinality of C gets larger. The results in Tables 1 and 2 show that the proposed algorithm can be efficiently applied to the single facility minimax location problems in question with planar instances, which arise frequently from real life.

TABLE 2. Numerical experiments corresponding to the set D_2 (time in s).

$\lvert C\rvert$ (1)	$\lvert V_C\rvert$ (2)	The average ratio $\frac{\lvert V_C\rvert}{\lvert C\rvert}$ (3)	Time to find conv(C) (4)	Computational time using NQH (5)	Total time (6) = (4) + (5)	Computational time without using NQH (7)
1000	25		$<10^{-4}$	0.0014	0.0014	0.0030
1000	39		$<10^{-4}$	0.0019	0.0019	0.0031
1000	35	2.76%	$<10^{-4}$	0.0016	0.0016	0.0032
1000	22		$<10^{-4}$	0.0012	0.0012	0.0031
1000	17		$<10^{-4}$	0.0009	0.0011	0.0031
10000	48		0.0010	0.0018	0.0028	0.0151
10000	81		0.0010	0.0079	0.0089	0.0156
10000	100	2.75%	0.0010	0.0117	0.0127	0.0157
10000	27		$<10^{-4}$	0.0017	0.0017	0.0151
10000	19		$<10^{-4}$	0.0015	0.0015	0.0151
100000	94		0.0151	0.0023	0.0174	0.0402
100000	124		0.0155	0.0027	0.0182	0.0413
100000	155	0.414%	0.0165	0.0037	0.0202	0.0432
100000	23		0.0010	0.0017	0.0027	0.0403
100000	18		0.0010	0.0017	0.0027	0.0401

Acknowledgements. We would like to thank the referees for their useful remarks and comments that helped us very much in revising the paper.

REFERENCES

[1] S.G. Akl and G.T. Toussaint, A fast convex hull algorithm, *Inform. Process. Lett.* **7** (1978) 219–222.
[2] P.T. An, Method of orienting curves for determining the convex hull of a finite set of points in the plane. *Optimization* **59** (2010) 175–179.
[3] J.L. Bentley, H.T. Kung, M. Schkolnick and C.D. Thompson, On the average number of maxima in a set of vectors and application. *J. Assoc. Comput. Machine* **25** (1978) 536–543.
[4] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press (2004).
[5] M.M. David, *Computation geometry*. Department of Computer Science (2002).
[6] H.N. Dung and N.K. Linh, Quicker than Quickhull. *Viet. J. Math.* (2014) DOI:10.1007/s10013-014-0067-1.
[7] P. Hansen, D. Peeters, D. Richard and J.F. Thisse, The minisum and minimax location problems revisited. *Oper. Res.* **33** (1985) 1251–1265.
[8] M. Kon and S. Kushimoto, A single facility minisum location problem under the A-distance. *J. Oper. Res. Soc. Jpn* **40** (1997) 10–20.
[9] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Kluwer Academic Publishers (2004).
[10] J. O'Rourke, *Computational geometry in C*, 2nd edn. Cambridge University Press (1998).
[11] F. Plastria, The generalized big square small square method for planar single facility location. *Eur. J. Oper. Res.* **62** (1992) 163–174.
[12] R.T. Rockafellar, *Convex Analysis*, Princeton University Press (1970).
[13] A.P. Ruszczynski, *Nonlinear Optimization*, Princeton University Press (2006).
[14] H. Tuy, A general d.c. approach to location problems. In *State of the art in global optimization: Computational methods and applications*, edited by C.A. Floudas and P.M. Pardalos. Kluwer (1996), 413–432.
[15] H.K. Xu, An iterative approach to quadratic optimization. *J. Optim. Theor. Appl.* **116** (2003) 659–678.