

An efficient algorithm for the single facility location problem with polyhedral norms and disk-shaped demand regions

André Berger¹ · Alexander Grigoriev¹  ·
Andrej Winokurow¹

Received: 25 January 2017 / Published online: 12 August 2017
© The Author(s) 2017. This article is an open access publication

Abstract The single facility location problem with demand regions seeks for a facility location minimizing the sum of the distances from n demand regions to the facility. The demand regions represent sales markets where the transportation costs are negligible. In this paper, we assume that all demand regions are disks of the same radius, and the distances are measured by a rectilinear norm, e.g. ℓ_1 or ℓ_∞ . We develop an exact combinatorial algorithm running in time $O(n \log^c n)$ for some c dependent only on the space dimension. The algorithm is generalizable to the other polyhedral norms.

Keywords 1-median · Single facility location problem · Rectilinear norm · Polyhedral norm · Exact algorithm

1 Introduction

In the conventional facility location problem each customer is associated with a single point in the space. Often in applications, the number of points is too large to consider them individually and the clients are combined in demand regions, see Dinler et al. [6]. In the basic problem considered in the present paper, we assume the transportation costs to a demand region are proportional to the shortest distance between any point in

✉ Alexander Grigoriev
a.grigoriev@maastrichtuniversity.nl

André Berger
a.berger@maastrichtuniversity.nl

Andrej Winokurow
a.winokurow@maastrichtuniversity.nl

¹ P.O. Box 616, 6200 MD Maastricht, The Netherlands

the region and the facility, also referred to as a *center*. The transportation costs within the demand region are negligible. While Brimberg and Wesolowsky [3] have used the same cost function with the regions being polytopes, we consider a special case where the demand regions are disks (balls) of radius $R \geq 0$ and the distances are measured by a rectilinear norm, e.g., ℓ_1 or ℓ_∞ . Notice, under ℓ_1 - and ℓ_∞ -norm a disk shape is not a conventional disk but a diamond and a square, respectively.

A special case of the problem with $R = 0$ and distances defined by the Euclidean norm ℓ_2 is the well-known *geometric 1-median* problem. It has been shown that for the geometric 1-median problem, neither a closed form formula, nor an exact algorithm involving only ruler and compass can be constructed, see Bajaj [1]. Kumar et al. [9] presented a randomized sampling algorithm, which for any $\varepsilon > 0$ with probability at least $1/2$ finds a $(1 + \varepsilon)$ -approximation to the geometric k -median problem in $O(2^{(k/\varepsilon)^{O(1)}} dn)$ time, where d is the space dimension and n is the number of points. Very recently, Cohen et al. [5] presented a deterministic algorithm computing the geometric median to arbitrary precision in nearly linear time. For $R = 0$ and under ℓ_1 - or ℓ_∞ -norm, Kalcsics et al. [7] constructed an algorithm solving the 1-median problem in $O(n \log^d n)$ time. In that work the authors utilized a general framework of Cohen and Megiddo [4] developed for minimization of a broad class of convex functions over polytopes.

In this paper we develop an *exact* algorithm for the single facility location problem with disk-shaped demand regions in \mathbb{R}^d under rectilinear distance measures. The running time of our exact algorithm matches the running time of the algorithm by Kalcsics et al. [7], but solves the problem for arbitrary $R \geq 0$. Then, we generalize the algorithm to generic polyhedral norms.

2 Problem definition

First, we derive the results for the ℓ_1 norm. Let $\rho(x) = \|x\|_1$ be the ℓ_1 -norm. Since a disk in \mathbb{R}^d under ℓ_1 -norm has a diamond shape, we refer to the disk-shaped demand regions as the *diamond-shaped* regions.

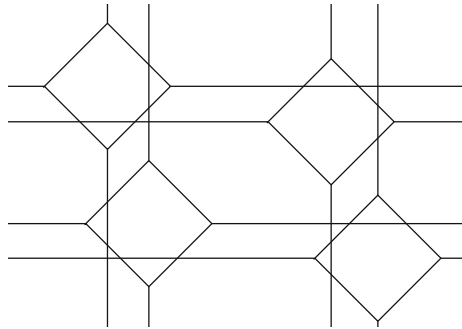
Given an input set $X = \{x_1, x_2, \dots, x_n\} \subseteq \mathbb{R}^d$ and $R \geq 0$, the *single facility location problem* (SFLP) with diamond-shaped regions is to find a center $c \in \mathbb{R}^d$, which minimizes

$$f(c) = \sum_{i=1}^n (\rho(x_i - c) - R)_+, \quad (1)$$

where $(x)_+ = \max\{0, x\}$, $x \in \mathbb{R}$. This problem can be reformulated as a linear program as follows. Let index $j \in \{1, 2, \dots, d\}$ specify the j th coordinate of a point. Then, the linear program reads

$$\min_{c, y, z} \sum_{i=1}^n z_i$$

Fig. 1 Linear regions for $d = 2, n = 4$



subject to

$$\begin{aligned}
 &y_{i,j} \geq x_{i,j} - c_j, \quad \forall i \in \{1, 2, \dots, n\}, j \in \{1, 2, \dots, d\} \\
 &y_{i,j} \geq -x_{i,j} + c_j, \quad \forall i \in \{1, 2, \dots, n\}, j \in \{1, 2, \dots, d\} \\
 &z_i \geq \sum_{j=1}^n y_{i,j} - R, \quad \forall i \in \{1, 2, \dots, n\}, \\
 &z_i \geq 0, \quad \forall i \in \{1, 2, \dots, n\},
 \end{aligned}$$

where for each data point $i \in \{1, 2, \dots, n\}$, variable $y_{i,j}$ is the distance between c and x_i projected on the j th coordinate, and z_i is the distance between c and x_i minus R if this value is non-negative, otherwise it equals 0.

This linear program has $nd + n + d$ variables and $2nd + n$ constraints. Vaidya [14] have shown that the time needed to solve such a linear program is $O((nd)^{2.5}L)$, where L is a parameter characterizing the constraint matrix. In this paper we introduce a simple combinatorial algorithm as an alternative to the solution of the above linear program. Moreover, our algorithm outperforms the classical linear programming algorithm.

The idea of the algorithm is purely geometrical. First, we observe, that the objective function $f(c)$ is convex and piecewise linear. Second, we can define polynomially many polytopal regions, such that the objective function is linear in each region; see Fig. 1 for an illustration of a two-dimensional case with four data points. Third, the algorithm runs a binary search in each dimension over these polytopal regions and finds an optimal solution.

3 A two-dimensional ℓ_1 case

To simplify the presentation, we first describe the algorithm for the planar case, i.e. for $d = 2$. For one data point x , the basic division of the space into regions, where f is linear, is depicted in Fig. 2a. This *basic drawing* is centered at x . For the case of more than one data point, say for x_1, x_2, \dots, x_n , we embed such basic drawings in the plane centering them at points x_1, x_2, \dots, x_n (Fig. 1), respectively. Similar to the graph drawing terminology, we call the union of the basic drawings an *embedding*. For

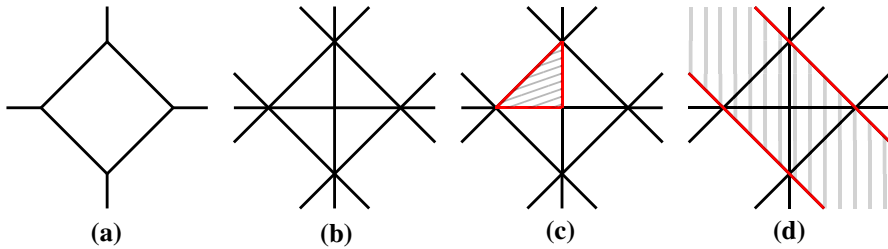


Fig. 2 Division of the space in linearity regions. **a** Linear regions, **b** Grid, **c** cell and **d** block of cells

simplicity of further calculations, we extend the line segments to the straight lines; see Fig. 2b. This creates a complex *grid* with two types of lines: vertical-horizontal and turned by 45° . Let a *cell* be an inclusion maximal polyhedron in the plane, not crossed by any line of the embedding. This is similar to a *face* of a planar graph embedding. Finally, we refer to the union of cells between any two consecutive parallel grid lines as a *block of cells*, see Fig. 2d.

Notice, division of the space into the regions where the objective function is linear is very well-known approach in Location Theory, see e.g. [10–13]. These regions are often referred to as *linearity regions*, *domains of linearity*, or *cells*, see the above references. For simplicity, in this paper we follow the latter notion of cells introduced in Nickel et al. [12]. The most important property of the problem is that within any cell of the embedding the function f is linear. Therefore the evaluation of the function in any point of a cell takes only $O(n)$ time.

In the two-dimensional case there are only four different line directions and, therefore, any cell has at most eight facets and eight vertices. Thus the evaluation of the function in all corner points of a cell takes $O(n)$ time and finding a minimum in a cell also takes linear time. Despite the fact that there are in general $O(n^2)$ cells and naive brute force would take cubic time, we are able to construct an almost linear time algorithm to the problem. Instead of enumerating over a quadratic number of cells, we run binary searches along four possible line directions, corresponding to four different types of blocks. Since the function f is convex, binary search correctly determines a cell containing a minimizer of the function. Binary search requires at most $O(\log n)$ steps per line direction, making the total number of steps $O(\log^4 n)$ for the four line directions of any two-dimensional case. At each step of this high dimensional binary search, the function f is evaluated, making the overall computation time of the algorithm at most $O(n \log^4 n)$. Details on the algorithm are deferred to the next section where we discuss the more general high dimensional case.

4 Higher dimensions and polyhedral norms

4.1 Notation and definitions

Notice that under the ℓ_1 -norm for any $x \in \mathbb{R}^d$ we have $\rho(x) = \max_{a \in \{-1, 1\}^d} a^T x$, where $\{-1, 1\}^d$ is the set of d -dimensional vectors consisting entries 1 and -1 . Therefore, we rewrite the objective function f as follows:

$$f(c) = \sum_{i=1}^n \left(\max_{a \in \{-1, 1\}^d} a^T(x_i - c) - R \right)_+ .$$

Similar to the two-dimensional case, we define the notions of grid, cells and blocks in \mathbb{R}^d . We first define the set of directions A , which is the set of orthogonal vectors to the norm defining hyperplanes. Let

$$A = \left\{ a \in \{-1, 0, 1\}^d \mid a \neq \{0\}^d \text{ and the first non-zero entry of } a \text{ is } 1 \right\} . \tag{2}$$

Notice, $|A| = (3^d - 1)/2$. For simplicity of notations we assume all vectors in A are indexed: $A = \{a_1, a_2, \dots, a_{|A|}\}$. Let us remind that $X = \{x_1, x_2, \dots, x_n\}$. For every vector $a_k \in A$, we define

$$B_k = \left\{ b_1^k, b_2^k, \dots, b_{|B_k|}^k \right\} = \begin{cases} \{a_k^T x \pm R \mid x \in X\} , & \text{if } a_k \in \{-1, 1\}^d; \\ \{a_k^T x \mid x \in X\} , & \text{otherwise,} \end{cases} \tag{3}$$

being the set of positions/levels of the hyperplanes orthogonal to $a_k \in A$. The intuition for the set B_k is that the hyperplanes $a_k^T x = b_j^k$, $x \in \mathbb{R}^d$, $b_j^k \in B_k$, are the delimiters of the linearity regions (cells) in the space, and these hyperplanes will be used in the construction of the high dimensional grid, blocks and cells, similarly to the two-dimensional case in Sect. 3. Again, without loss of generality, assume $b_1^k \leq b_2^k \leq \dots \leq b_{|B_k|}^k$, otherwise we sort the positions in the preprocessing step in $O(n \log n)$ time. For convenience, let us introduce $b_0^k = -\infty$ and $b_{|B_k|+1}^k = +\infty$. Now, we define the grid, cells and blocks of cells for a general high dimensional case.

Definition 1 The hyperplanes $\{x \in \mathbb{R}^d \mid a_k^T x = b_j^k, a_k \in A, b_j^k \in B_k\}$ form the *grid*. A *cell* is an inclusion maximal polyhedron in \mathbb{R}^d not crossed by a grid hyperplane. A set of indices $\{j_1, j_2, \dots, j_{|A|}\}$, where $j_k \in \{0, 1, \dots, |B_k| + 1\}$ for any $k \in \{1, 2, \dots, |A|\}$, completely specifies the cell:

$$C(0; j_1, j_2, \dots, j_{|A|}) = \left\{ x \in \mathbb{R}^d \mid b_{j_k}^k \leq a_k^T x \leq b_{j_k+1}^k, a_k \in A \right\} . \tag{4}$$

Leaving out l constraints in the definition of a cell, we arrive at the definition of a *block* of cells parameterized by l :

$$C(l; j_1, j_2, \dots, j_{|A|-l}) = \left\{ x \in \mathbb{R}^d \mid b_{j_k}^k \leq a_k^T x \leq b_{j_k+1}^k \text{ for any } k \in \{1, 2, \dots, |A| - l\} \right\} . \tag{5}$$

4.2 Observations

The following observations are necessary ingredients of the algorithm and its further analysis. First of all, notice that for any cell C and for any $x \in X$, either the cell is within the range R from x and then

$$\left(\max_{a \in \{-1,1\}^d} a^T(c-x) - R \right)_+ = 0, \quad \forall c \in C, \tag{6}$$

or there exists a vector $a_0 \in \{-1, 1\}^d$ such that

$$\left(\max_{a \in \{-1,1\}^d} a^T(c-x) - R \right)_+ = a_0^T(c-x) - R, \quad \forall c \in C. \tag{7}$$

In other words, every individual contribution of a data point $x \in X$ to the objective function f is linear in any cell C . As the sum of linear functions is a linear function, f is linear in any cell C .

The second observation is that every individual contribution of a data point $x \in X$ to the objective function f is convex as it is a maximum of two linear functions. Since the sum of convex functions is convex, the function f is convex.

The third observation is that for any cell C the minimum of f over C can be found in $O(dn + d^{4.5}3^d)$ time. By linearity of the function f within a cell, we have to compute d coefficients of that linear function. Computation of any coefficient requires at most $O(n)$ time. Thus, specification of the linear objective function in a cell is obtained in $O(dn)$ time. Then, solving the linear program minimizing f on polyhedron cell C takes at most $O(d^{4.5}3^d)$ time, e.g., using Karmarkar’s Algorithm [8].

4.3 High dimensional recursive binary search algorithm

Now, we are ready to present the algorithm and to analyze its running time.

Algorithm 1:

```

Data:  $X = \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$ .
Result:  $c^* = \arg \min_{c \in \mathbb{R}^d} f(c)$ .
Algorithm main()
    // Define the grid by the sets  $A$  and  $B_k$ ,  $a_k \in A$ , according to the
    // Definitions (2) and (3).
    // For every  $a_k \in A$ , sort elements of  $B_k$  in non-decreasing order.
    return minimize( $C(|A|; \emptyset)$ ).
Procedure minimize( $C(l; j_1, j_2, \dots, j_{|A|-l})$ )
    // This procedure returns the point  $c_l^* \in C(l; j_1, j_2, \dots, j_{|A|-l})$  where
    // the minimum of the objective function in the block is
    // attained, i.e.,  $f(c_l^*) \leq f(c)$  for all  $c \in C(l; j_1, j_2, \dots, j_{|A|-l})$ .
    // Consider the segment of integers  $S_l = \{0, 1, \dots, |B_{|A|-l+1}| + 1\}$ .
    // Let  $c_{l-1,q}^*$  be the point returned in
    // minimize( $C(l-1; j_1, j_2, \dots, j_{|A|-l}, q)$ ),  $q \in S_l$ .
    // For  $l \geq 1$ , using binary search on  $q \in S_l$  find an integer  $q^*$ 
    // minimizing  $f(c_{l-1,q}^*)$ . Let  $c_l^* = c_{l-1,q^*}^*$ .
    // For  $l = 0$ , solve the linear program minimizing function  $f$  on
    // polyhedron cell  $C(0; j_1, j_2, \dots, j_{|A|})$ . Let  $c_l^*$  be a solution to the
    // linear program.
    return  $c_l^*$ .

```

Theorem 1 Algorithm 1 returns an optimal solution $c^* \in \mathbb{R}^d$ to the single facility location problem with diamond-shaped regions in $O\left(dn \log^{3^d} n + d^{4.5} 3^d \log^{3^d} n\right)$ time.

Proof By convexity of the function f , the binary search correctly computes the minimum objective function values and the corresponding solution points in the blocks of cells. Given that the binary search for a single parameter q takes at most $O(\log n)$ time and the recursion depth in the algorithm is at most 3^d , we have that the binary search takes $O(\log^{3^d} n)$ steps in total. In every leaf of the binary search tree, the algorithm evaluates a minimum of the function f in a cell. By the observation above, this evaluation is done by solving a linear program in $O(dn + d^{4.5} 3^d)$ time. Hence, the claimed time complexity follows. \square

Algorithm 1 works also for l_∞ and for more general polyhedral norms. We use the definition of a polyhedral norm as in Barvinok et al. [2]. Let P be a centrally symmetric bounded polyhedron in \mathbb{R}^d with the origin in its center. By symmetry, the number of facets is even. Let the number of facets be $2m$. Then, there is a set $H_P = \{h_1, h_2, \dots, h_m\}$ of points in \mathbb{R}^d such that P is the intersection of a collection of half-spaces defined by H_P :

$$P = \left(\bigcap_{i=1}^m \{x \in \mathbb{R}^d \mid h_i^T x \leq 1\} \right) \cap \left(\bigcap_{i=1}^m \{x \in \mathbb{R}^d \mid h_i^T x \geq -1\} \right).$$

Then, the polyhedral norm is defined by $\rho(x) = \max_{1 \leq i \leq m} |h_i^T x|$. For instance, for ℓ_1 in \mathbb{R}^2 we can take $H_P = \{(1, 1), (-1, 1)\}$ and for ℓ_∞ in \mathbb{R}^2 we can take $H_P = \{(0, 1), (1, 0)\}$.

For a polyhedral norm, the only adjustment we have to make in the Algorithm 1 is to generalize sets A and B in the definition of the grid. For this generalization, let

$$A_1 = \{\pm h_i \mid h_i \in H_P, 1 \leq i \leq m\}, \tag{8}$$

$$A_2 = \{h_i - h_j \mid h_i, h_j \in H_P, 1 \leq i, j \leq m, i \neq j\}, \tag{9}$$

$$A = A_1 \cup A_2. \tag{10}$$

Again, we assume all vectors in A are indexed: $A = \{a_1, a_2, \dots, a_{|A|}\}$. For every vector $a_k \in A$, we define

$$B_k = \left\{ b_1^k, b_2^k, \dots, b_{|B_k|}^k \right\} = \begin{cases} \{a_k^T x \pm R \mid x \in X\}, & \text{if } a_k \in A_1; \\ \{a_k^T x \mid x \in X\}, & \text{if } a_k \in A_2. \end{cases} \tag{11}$$

Now, to adjust Algorithm 1 to a generic polyhedral norm we define the grid by the sets A and B_k , $a_k \in A$, according to the definitions (10) and (11). The rest of the algorithm remains intact.

Finally, let us estimate the run-time of Algorithm 1 adjusted to a polyhedral norm. The new set A determines the types of the grid hyperplanes. For each $a_k \in A$ the set B_k determines all hyperplanes of type a_k . Notice, $|A| = O(m^2)$ as the size of A_2 is quadratic in the number of facets. By definition, for every $a_k \in A$, the size of the set B_k is at most $2n$. Thus, the number of grid cells created in this construction is $n^{O(m^2)}$. Using the adjusted number of grid cells and following literally the same arguments as in Theorem 1, we generalize the result as follows.

Theorem 2 *Under a polyhedral norm determined by a polyhedron with $2m$ facets, the adjusted Algorithm 1 returns an optimal solution $c^* \in \mathbb{R}^d$ to the single facility location problem with disk-shaped demand regions in $O\left((dn + d^{4.5}m^2) \log^{O(m^2)} n\right)$ time.*

Since $2m \geq d + 1$ for any bounded polyhedron in \mathbb{R}^d , we have the following straightforward corollary:

Corollary 1 *Under a polyhedral norm determined by a polyhedron with $2m$ facets, the adjusted Algorithm 1 returns an optimal solution to the single facility location problem with disk-shaped demand regions in $O\left((mn + m^{6.5}) \log^{O(m^2)} n\right)$ time.*

Acknowledgements We thank anonymous reviewers for the thorough reading of this paper, providing us with many relevant references, and for very useful suggestions which helped us to improve the presentation.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Bajaj, C.: Proving geometric algorithms nonsolvability: an application of factoring polynomials. *J. Symb. Comput.* **2**(1), 99–102 (1986)
2. Barvinok, A., Johnson, D.S., Woeginger, G.J., Woodroof, R.: The maximum traveling salesman problem under polyhedral norms. In: Bixby, R.E. et al. (Eds.) *Integer Programming and Combinatorial Optimization*, 6th International IPCO Conference, Houston, Texas, USA, June 22–24, 1998, Proceedings. *Lecture Notes in Computer Science* 1412, Springer, Berlin-Heidelberg 1998, pp. 195–201 (1998)
3. Brimberg, J., Wesolowsky, G.O.: Minisum location with closest Euclidean distances. *Ann. Oper. Res.* **111**(1), 151–165 (2002)
4. Cohen, E., Megiddo, N.: Maximizing concave functions in fixed dimension. In: Pardalos, P.M. (ed.) *Complexity in Numerical Optimization*, pp. 74–87. World Scientific, Singapore (1993)
5. Cohen, M., Lee, Y.T., Miller, G., Pachocki, J., Sidford, A.: Geometric median in nearly linear time. In: *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC) 2016*, Cambridge, MA, USA, pp. 9–21 (2016)
6. Dinler, D., Tural, M.K., Iyigun, C.: Heuristics for a continuous multi-facility location problem with demand regions. *Comput. Oper. Res.* **62**, 237–256 (2015)
7. Kalcsics, J., Nickel, S., Puerto, J., Tamir, A.: Algorithmic results for ordered median problems. *Oper. Res. Lett.* **30**(3), 149–158 (2002)
8. Karmarkar, N.: A new polynomial-time algorithm for linear programming. *Combinatorica* **4**(4), 373–395 (1984)
9. Kumar, A., Sabharwal, Y., Sen, S.: Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM* **57**(2), 1–32 (2010)

10. Nickel, S., Puerto, J.: *Location Theory: A Unified Approach*. Springer, Berlin (2005)
11. Nickel, S., Puerto, J., Rodriguez-Chia, A.M.: An approach to location models involving sets as existing facilities. *Math. Oper. Res.* **28**(4), 693–715 (2003)
12. Nickel, S., Puerto, J., Rodriguez-Chia, A.M., Weißler, A.: Multicriteria planar ordered median problems. *J. Optim. Theory Appl.* **126**(3), 657–683 (2005)
13. Puerto, J., Fernandez, F.R.: Geometrical properties of the symmetric single facility location problem. *J. Nonlinear Convex Anal.* **1**(3), 321–342 (2000)
14. Vaidya, P.M.: Speeding-up linear programming using fast matrix multiplication. In: *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (SFCS) 1989*, pp. 332–337, Research Triangle Park, North Carolina, USA (1989)