Journal of Global Optimization 7: 209–227, 1995.

A D.C. Optimization Method for Single Facility Location Problems*

HOANG TUY[1], FAIZ AL-KHAYYAL[2] and FANGJUN ZHOU[2]
[1]*Institute of Mathematics, P.O. Box 631, Bo Ho, Hanoi, Vietnam;*
[2]*School of Industrial and Systems Engineering, Georgia Institute of Technology Atlanta, GA 30332, U.S.A. e-mail:faiz@isye,gatech.edu*

(Received: 16 January 1995; accepted: 28 April 1995)

Abstract. The single facility location problem with general attraction and repulsion functions is considered. An algorithm based on a representation of the objective function as the difference of two convex (d.c.) functions is proposed. Convergence to a global solution of the problem is proven and extensive computational experience with an implementation of the procedure is reported for up to 100,000 points. The procedure is also extended to solve conditional and limited distance location problems. We report on limited computational experiments on these extensions.

Key words: Facility location, d.c optimization, global optimization, nondifferentiable optimization.

1. Introduction

The origins of location theory can be traced to Fermat's challenge, in his 17th century essay on maxima and minima, to find a point in the plane such that the sum of its distances to three given points is a minimum. The problem and its generalization to weighted sums of distances remained in the province of mathematicians until the early 20th century when Weber [20] used the theory for the location of industries. Today, the simplest version of Weber's problem (sometimes called the Fermat–Weber problem) is to locate a single facility in the plane that minimizes the sum of weighted Euclidean distances to the locations of n known users. The historical developments and many contributions to the problem and its extensions are well documented in the literature (see, e.g., Love *et al.* [8]).

When all of the weights are positive, the objective function is convex and the problem is easily solved by an iterative procedure proposed by Weiszfeld [21]. The problem becomes much harder when some of the weights are taken to be negative. In this case, when minimizing the objective function, users having positive weights are *attracted* to the facility and those having negative weights are *repelled* by the facility. For example, a neighborhood school or civic center is attractive to most residents of a community, but those with contiguous properties may understandably object because of increased noise and traffic, and decreased

* This research was supported in part by the National Science Foundation Grant DDM-91-14489.

residential property values. The first exact algorithm for finding a global solution to the general problem was proposed by Chen *et al.* [2]. The procedure exploited recent results from d.c. programming and was implemented to solve randomly generated problems with up to 1000 users. In addition, Chen *et al.* [2] extend their procedure to exponentially decaying repulsion and to constraining the facility to be located in one of a finite number of disjoint convex polygons. In further extensions to their work, Chen *et al.* [3] propose exact procedures for the multisource Weber problem, the conditional multisource Weber problem, and the limited distance location problem (see Section 7 below for definitions of these problems). Recently, Maranas and Floudas [9] developed a procedure for locating a single facility in a rectangle, containing all users, that directly exploits the problem structure. The authors report computational results on solving randomly generated problems with up to 10,000 users.

The location problems discussed above were all variations and extensions of the classical Weber problem. Generalizations to the Weber problem give rise to harder problems that capture more applications. One such problem was considered by Idrissi *et al.* [7] involving the *maximization* of the sum of decreasing convex functions of arbitrary distance metrics. The problem is formulated only for attraction points, but each user can have a different metric which need not be symmetric, so that both norms and gauges are possible. Idrissi *et al.* [7] develop a procedure based on solving a sequence of parameterized Weber problems for finding a *local* solution to the problem. Tuy and Al-Khayyal [18] proposed the first algorithm for finding global solutions to the problem by reducing it to the solution of a sequence of unconstrained nondifferentiable convex minimization problems and specializing a polyhedral annexation procedure for the case when the distance metric is a polyhedral norm. Computational results were not reported.

In this paper we extend the work of Tuy and Al-Khayyal [18] to the case of both attraction and repulsion, and develop an entirely new procedure based on a d.c. reformulation of the problem. We implement our procedure and report on computational experiments on more than three-thousand randomly generated test problems. We also discuss extensions of the basic procedure to limited distance and conditional location problems. The remainder of the paper is organized as follows. The problem under consideration is formally presented in Section 2 and its d.c. reformulation is described in Section 3. The global optimization algorithm is presented in Section 4 and its convergence is treated in Section 5. The results of computational experiments on test problems with up to 100,000 users are reported in Section 6. Extensions to the limited distance and conditional location problems are discussed in Section 7, including computational results on these problems. We also include an appendix on how to compute the subgradients called for by the procedure. In the interest of brevity, the remainder of the paper assumes a basic knowledge of the fundamental concepts in deterministic global optimization as detailed in Horst and Tuy [6].

2. Facility Location in the Presence of Attraction and Repulsion

The following problem has been studied in [7] and [18]:

A facility must be constructed in some area M in the plane (a convex polygon in R^2) to serve n users located at points $a^j \in M$. If the facility is located at $x \in M$, then the attraction of the facility to user j is $q_j(h_j(x))$, where $h_j(x) = \|x - a^j\|$ is the distance from x to a^j and $q_j : R_+ \to R_+$ is a convex decreasing function (the farther x is away from a^j the less attractive it looks to user j). We want to determine the location of the facility so as to achieve maximal total attraction, i.e.

$$\text{maximize } \sum_{j=1}^{n} q_j[h_j(x)] \quad \text{s.t. } x \in M. \tag{1}$$

In this model only attraction points are considered. In practice, aside from attraction points there may exist repulsion points as well ([5], [2], [9]). For example, in the presence of garbage dumps, sewage plants, or nuclear plants in the area, one may wish the facility to be located as far away from these points as possible. If J_1 is the set of a ttraction points and J_2 the set of repulsion points, then instead of (1) one should seek to maximize the function

$$F(x) := \sum_{j \in J_1} q_j[h_j(x)] - \sum_{j \in J_2} q_j[h_j(x)]. \tag{2}$$

In typical situations we have

$$q_j[h_j(x)] = \begin{cases} \alpha_j - w_j\|x - a^j\| & j \in J_1, \\ w_j e^{-\theta_j\|x-a^j\|} & j \in J_2. \end{cases}$$

where $\alpha_j > 0$ is the maximal attraction of point $j \in J_1$, $\theta_j > 0$ is the rate of decay of the repulsion of point $j \in J_2$ and $w_j > 0$ for all j (the classical Weber's problem corresponds to the case $J_2 = \emptyset$).

Frequently, $\|\cdot\|$ is taken to be the usual Euclidean norm, but in the general case, a different norm or even a gauge function can be associated with each point a^j.

Thus the general single facility location problem can be formulated as:

$$\max \left\{ \sum_{j \in J_1} q_j[h_j(x)] - \sum_{j \in J_2} q_j[h_j(x)] : \quad x \in M \right\}, \qquad (P)$$

where M is a convex polygon in R^2 and, for every $j = 1, \ldots, n$, $q_j : R_+ \to R_+$ is a convex decreasing function, while $h_j : R^2 \to R_+$ is a convex function such that $h_j(a^j) = 0$, $h_j(x) > 0$ for $x \neq a^j$ and $h_j(x) \to \infty$ as $\|x\| \to +\infty$ (see [16]).

Since the objective function (to be maximized) is not concave, problem (P) is a nonconvex global optimization problem for which many local maximizers may exist which are not global solutions. However, under mild assumptions, we will show in the next section that (2) is in fact a d.c. function, so that (P) is a d.c.

maximization problem over M. Since the dimension of x is small ($M \subset R^2$,) this problem can be solved by recently developed d.c. optimization methods (see e.g. [16]), even for fairly large values of n.

3. D.C. Reformulation of the Location Problem

In the results below (see [17]), the function $q^+(t)$ represents the *right derivative* of $q(t)$ (see [10]).

LEMMA 1. *Let $h\colon M \to R_+$ be a convex function on a compact convex subset of R^m. If $q\colon R_+ \to R$ is a convex nonincreasing function such that $q^+(0) > -\infty$, then $q[h(x)]$ can be expressed as the d.c. function on M*

$$q[h(x)] = g(x) - Ch(x),$$

where $g(x)$ is a convex function and C is a positive constant satisfying $C \geq |q^+(0)|$.

Proof. We have $q^+(0) \leq q^+(t) \leq 0 \quad \forall t \geq 0$, therefore $\tilde{q}(t) = q(t) + Ct$ satisfies $\tilde{q}^+(t) = q^+(t) + C \geq q^+(0) + C \geq 0 \quad \forall t \geq 0$, which implies that the convex function $\tilde{q}$ is nondecreasing. For any $x, x', \alpha \in [0,1]$ we then have $\tilde{q}[h(\alpha x + (1-\alpha)x')] \leq \tilde{q}[\alpha h(x) + (1-\alpha)h(x')] \leq \alpha \tilde{q}[h(x)] + (1-\alpha)\tilde{q}[h(x')]$. Consequently, the function $g(x) := \tilde{q}[h(x)]$ is convex and $g(x) = q[h(x)] + Ch(x)$, as was to be proved. □

LEMMA 2. *Let $h\colon M \to R_+$ be a convex function on a compact convex subset of R^m. If $q\colon R_+ \to R$ is a concave nondecreasing function such that $q^+(0) < \infty$, then $q[h(x)]$ can be expressed as the d.c. function on M*

$$q[h(x)] = g(x) + Ch(x),$$

where $g(x)$ is a concave function and C is a positive constant satisfying $C \geq |q^+(0)|$.

PROPOSITION 1. *Assume*

$$q_j^+(0) > -\infty \quad \forall j. \tag{3}$$

Then the function $F(x)$ defined by (2) can be expressed as:

$$F(x) = G(x) - H(x),$$

with $G(x)$ and $H(x)$ being convex functions defined by

$$\begin{aligned} G(x) &:= \sum_{j \in J_1} g_j(x) + \sum_{j \in J_2} C_j h_j(x) \\ H(x) &:= \sum_{j \in J_2} g_j(x) + \sum_{j \in J_1} C_j h_j(x) \\ g_j(x) &= q_j[h_j(x)] + C_j h_j(x), \quad j = 1, \ldots, n. \end{aligned}$$

and C_j being constants satisfying $C_j \geq |q_j^+(0)|$.

Proof . By Lemma 1,

$$F(x) = \left[\sum_{j\in J_1} g_j(x) + \sum_{j\in J_2} C_j h_j(x)\right] - \left[\sum_{j\in J_2} g_j(x) + \sum_{j\in J_1} C_j h_j(x)\right],$$

whence the result. □

Thus problem (P) can be rewritten as the d.c. maximization problem:

$$\max\{G(x) - H(x) : \ x \in M\}. \qquad (Q)$$

REMARK. The single facility location problem also has the equivalent cost formulation

$$\min\left\{\sum_{j\in J_1} q_j[h_j(x)] - \sum_{j\in J_2} q_j[h_j(x)] : \quad x \in M\right\},$$

where each q_j is now a concave increasing function.

4. A D.C. Maximization Algorithm

Several methods are available for solving the d.c. maximization problem (Q) (see [6], [16] and references therein). For instance, one can convert (Q) into the convex maximization problem [6]

$$\max\{G(x) - t : \ H(x) \leq t, \ x \in M\}$$

and solve the latter by outer approximation (see e.g. [2]). Alternatively, one can solve (Q) by a branch and bound method using rectangular subdivision as in [9]. However, the outer approximation method needs an additional variable t, while a rectangular branch and bound method would require for bounding the computation of a concave envelope of $G(x) - H(x)$ over each rectangle (recall that this is a maximization problem), and has to use a non-adaptive exhaustive subdivision process defined independently of the problem conditions at every current iteration. In addition, both methods assume special forms of the functions $q_j(\cdot), j = 1, \ldots, n$.

To better exploit the structure of (Q), in the sequel we propose to solve (Q) using a triangular branch and bound method in which upper bounds are computed as in ([15]) (see also [16] (Section 9.1.1, Remark (ii)), while branching follows a *normal* triangular subdivision scheme [15] (see also [6]). The bounding and subdivision operations of the procedure are explained below before the formal presentation of the algorithm.

I. *Bound computation.* At iteration k of the procedure we have in hand a convex piecewise affine minorant $L_k(x)$ with $|I_k|$ linear pieces $l_i(x)$ resulting

from the previous iterations. For each newly generated triangle S, let $G_S(x)$ be the affine function that agrees with $G(x)$ at the vertices of S (because of the convexity of $G(x)$, $G_S(x)$ is a majorant of $G(x)$ on S). Then an upper bound for $\max\{G(x) - H(x) : x \in S \cap M\}$ is given by the optimal value of the linear program

$$\text{LP}(L_k, S) \qquad \alpha(S) := \max\{G_S(x) - t : l_i(x) \leq t \ (i \in I_k),\ x \in S \cap M\}$$

where $l_i(x)$ are linear functions such that, for every k, $\{x \in S : l_i(x) \leq t, i \in I_k\} \supseteq \{x \in S : H(x) \leq t\}$. The latter program is equivalent to $\max\{G_S(x) - L_k(x) : x \in S \cap M\}$.

The bound computed as above depends upon the function $L_k(x)$ used for underestimating $H(x)$ on S. To ensure asymptotic exactness (consistency) of this bound, the underestimator $L_k(x)$ must be refined at each iteration k. The refinement is achieved by computing, at each iteration k, a linearization $l_k(x)$ of $H(x)$ at an appropriate point x^k and defining $L_{k+1}(x) = \sup\{L_k(x), l_k(x)\}$. The point x^k, at which the linearization $l_k(x)$ of $H(x)$ should be constructed, is a basic optimal solution of the linear program $\text{LP}(L_k, S_k)$ associated with the most promising triangle S_k at this stage. We prove later that this will ensure convergence of the sequence $H(x^k) - L_k(x^k) \to 0$.

As a linearization of $H(x)$ at x^k we take

$$l_k(x) = \langle p^k, x - x^k \rangle + H(x^k),$$

where p^k is a subgradient of $H(x)$ at x^k. Although $H(x)$ results from the compositions of a number of functions, a subgradient of it at any point can be computed easily (see Appendix for the details of this computation).

II. *Subdivision.* Given any triangle S, and a point $y \in S$ which is not a vertex of S, we can join y to the vertices of S to define a *radial subdivision* of S in subtriangles. When y is the midpoint of the longest side of S the subdivision is called a *bisection.* A basic result of the theory of simplicial subdivision processes is the following [13].

PROPOSITION 2. *Let $\{S_\nu\}$ be a nested sequence of m-simplices in R^m and for each ν let x^ν be any point in S_ν. Assume that*

(1) *For infinitely many ν, $S_{\nu+1}$ is a subsimplex of S_ν in a bisection;*
(2) *For all other ν's, $S_{\nu+1}$ is a subsimplex of S_ν in a radial subdivision via x^ν.*

Then at least one cluster point of the sequence $\{x^\nu\}$ is a vertex of the simplex $S_\infty = \cap_{\nu=1}^{\infty} S_\nu$.

Now in the proposed branch and bound algorithm, at every iteration k a triangle S_k is selected which must be further subdivided. As S_k should be in a sense the most promising among the triangles still of interest in the current partition, we

choose S_k to be the triangle S with maximal $\alpha(S)$, among all mentioned triangles. Let x^k be a basic optimal solution of the linear program $\mathrm{LP}(L_k, S_k)$ used for computing the upper bound $\alpha(S_k)$. Then, from the definition of $\mathrm{LP}(L_k, S_k)$ and the fact $\alpha(S_k) \in \alpha(S)$ for all triangles S in the current partition we can easily see that

$$G_{S_k}(x^k) - L_k(x^k) \geq G(x) - L_k(x) \geq G(x) - H(x) \quad \forall x \in M.$$

Therefore, if x^k is a vertex of S_k (so that $G_{S_k}(x^k) = G(x^k)$) and if $L_k(x^k) = H(x^k)$, then the above inequality shows that x^k is an optimal solution of (Q) and the algorithm terminates. Thus, the algorithm is continued only in either of the following situations:

(1) x^k is not a vertex of S_k;
(2) x^k is a vertex of S_k while $L_k(x^k) < H(x^k)$.

Furthermore, the above considerations suggest that the algorithm should try to bring x^k closer and closer to a vertex of S_k, while reducing the difference $H(x^k) - L_k(x^k)$ to zero (the latter fact can be ensured by an appropriate selection of the sequence $L_k(\cdot)$). On the basis of Proposition 4 we can ensure that x^k comes closer and closer to a vertex of S_k by generating a subdivision process such that: at every iteration k, S_k is either bisected, or is radially subdivided via x^k, while any infinite nested sequence of triangles generated by the subdivision process involves infinitely many bisections. A subdivision process satisfying this condition is called *normal.*

Practically, to generate a normal subdivision process, every triangle S generated during the process is assigned a generation index $\tau(S)$ such that any initial triangle S (triangles of first generation) has index $\tau(S) = 1$, and every son S' of a triangle S has index $\tau(S') = \tau(S) + 1$. Then for any (arbitrary) infinitely increasing sequence Δ of natural numbers, we seek to

bisect S_k if $k \in \Delta$, and subdivide S_k via x^k, otherwise.

The normality of such a rule is immediate. It is also obvious that the sequence Δ expresses how often we use bisections in the subdivision process.

The proposed algorithm can now be described as follows.

Algorithm

Step 0. Start with a set $\mathcal{S}_1$ of triangles covering M or a region known to contain at least a global optimal solution (e.g. take $\mathcal{S}_1 = \{S_1\}$, where S_1 is a triangle containing M), together with a point $x^0 \in M$, the best feasible point available. Let $l_0(x) = \langle p^0, x - x^0 \rangle + H(x^0)$, where $p^0 \in \partial H(x^0)$. Set $L_1(x) = l_0(x)$, $\bar{x} = x^0$, $\gamma = G(\bar{x}) - H(\bar{x})$, $\mathcal{R}_1 = \mathcal{P}_1 = \mathcal{S}_1$, $\tau(S) = 1$ for every $S \in \mathcal{S}_1$. Set $k = 1$.

Step 1. For each $S \in \mathcal{P}_k$, solve the linear program $\mathrm{LP}(L_k, S)$ to obtain the optimal value $\alpha(S)$ and a basic optimal solution $x(S)$.

Step 2. Update $\bar{x}$ and γ. Delete every $S \in \mathcal{S}_k$ such that $\alpha(S) \leq \gamma$ and let $\mathcal{R}_k$ be the set of remaining simplices. If $\mathcal{R}_k = \emptyset$, then terminate: $\bar{x}$ solves (Q).

Step 3. Select $S_k \in \text{argmax}\{\alpha(S) : S \in \mathcal{R}_k\}$. Let $x^k = x(S_k)$, $t^k = L_k(x^k)$.

3a. If $t^k = H(x^k)$ and x^k is a vertex of S_k then terminate: x^k solves (Q).

3b. If $t^k = H(x^k)$ and x^k is not a vertex of S_k then set $L_{k+1}(x) = L_k(x)$.

3c. Otherwise, define $l_k(x) = \langle p^k, x - x^k \rangle + H(x^k)$ with $p^k \in \partial H(x^k)$ and let $L_{k+1}(x) = \sup\{L_k(x), l_k(x)\}$.

Step 4. Bisect S_k or subdivide S_k via x^k, according to a normal rule.

Let $\mathcal{P}_{k+1}$ be the partition of S_k. Set $\tau(S) = \tau(S_k) + 1$ for every $S \in \mathcal{P}_{k+1}$, $\mathcal{S}_{k+1} = (\mathcal{R}_k \setminus \{S_k\}) \cup \mathcal{P}_{k+1}$, $k \leftarrow k + 1$ and go back to Step 1.

REMARK. The sequence Δ in the normal subdivision rule can be chosen once at the beginning. For example we can take a natural number N (typically $N = 5$ as in many computational experiments with branch and bound algorithms of concave minimization) and define $\Delta = \{N, 2N, 3N, \ldots, \}$ (sequence of multiples of N). Alternatively, since theoretically Δ can be arbitrary, a more efficient strategy may be to define the elements of Δ one by one, as needed, in the course of the algorithm, and use Δ to monitor, to some extent, the speed of convergence of the algorithm. In general, one may try to use as many subdivisions via x^k as possible, and have recourse to a bisection when the algorithm seems to slow down.

5. Convergence

If the algorithm terminates at Step 3a then $L_k(x^k) = H(x^k)$ (because $t^k = L_k(x^k)$), while $G_{S_k}(x^k) = G(x^k)$ and, as we saw above, x^k is indeed a global optimal solution of (Q).

Suppose now that the algorithm is infinite, so that an infinite sequence $\{x^k\}$ is generated. Write $G_k(x)$ for $G_{S_k}(x)$.

LEMMA 3. *There exists a subsequence $x^{k_\nu} \to \tilde{x}$ such that $G_{k_\nu}(x^{k_\nu}) \to G(\tilde{x})$.*

Proof. By a standard argument, the algorithm, if infinite, must generate at least an infinite sequence of nested triangles. Let $\{S_{k_\nu}\}$ be such a sequence. Since the subdivision process is normal, by Proposition 4, at least one cluster point of the sequence $\{x^{k_\nu}\}$ coincides with a vertex $\tilde{x}$ of the limit triangle $S_\infty = \cap_\nu S_{k_\nu}$. By taking a subsequence if necessary, we can assume that $x^{k_\nu} \to \tilde{x}$. Let $v^{k,i}, i = 1, 2, 3$, be the vertices of S_k, so that $x^k = \sum_{i=1}^3 \lambda_{k,i} v^{k,i}$ with $\lambda_{k,i} \geq 0$, $\sum_{i=1}^3 \lambda_{k,i} = 1$, and $G_k(x^k) = \sum_{i=1}^3 \lambda_{k,i} G(v^{k,i})$. We may assume $v^{k_\nu,i} \to \bar{v}^i, i = 1, 2, 3$, so that S_∞ is the convex hull of $\bar{v}^1, \bar{v}^2, \bar{v}^3$. Since $\tilde{x}$ is a vertex of S_∞ we have $\tilde{x} \in \{\bar{v}^1, \bar{v}^2, \bar{v}^3\}$, for instance, $\tilde{x} = \bar{v}^1$, i.e. $v^{k_\nu,1} \to \tilde{x}$. The latter fact implies that, by taking a subsequence if necessary, $\lambda_{k_\nu,1} \to 1$, $\lambda_{k_\nu,i} \to 0$ $(i \neq 1)$, and hence,

$G_{k_\nu}(x^{k_\nu}) \to G(\tilde{x})$. □

LEMMA 4. *For any subsequence* $x^{k_\nu} \to \tilde{x}$ *we have* $L_{k_\nu}(x^{k_\nu}) \to H(\tilde{x})$.

Proof. Since the sequence $\{x^k\}$ is bounded, then the sequence $\{p^k\}$ is also bounded (see [10]). We can, therefore, assume that $p^{k_\nu} \to p \in \partial H(\tilde{x})$. Now for $\nu > \mu$ we have

$$H(x^{k_\nu}) \geq t_{k_\nu} = L_{k_\nu}(x^{k_\nu}) \geq l_{k_\mu}(x^{k_\nu}) = \langle p^{k_\nu}, x^{k_\nu} - x^{k_\mu} \rangle + H(x^{k_\mu}).$$

Letting $\nu \to \infty$, we obtain

$$H(\tilde{x}) \geq \limsup L_{k_\nu}(x^{k_\nu}) \geq \liminf L_{k_\nu}(x^{k_\nu}) \geq \langle p^{k_\mu}, \tilde{x} - x^{k_\mu} \rangle + H(x^{k_\mu}),$$

and letting $\mu \to \infty$ yields the desired result. □

Let $\bar{x}^k$ be the current best feasible solution at iteration k and $\gamma_k = F(\bar{x}^k)$.

THEOREM 1. *If the above Algorithm is infinite, then at least one cluster point of the sequence* $\{x^k\}$ *is a global optimal solution. Hence,* γ_k *tends to the optimal value of problem (Q) and any cluster point of the sequence* $\bar{x}^k$ *is a global optimal solution of (Q).*

Proof. By Lemmas 2 and 3, there exists a subsequence $x^{k_\nu} \to \tilde{x}$ such that

$$G_{k_\nu}(x^{k_\nu}) \to G(\tilde{x}), \quad K_{k_\nu}(x^{k_\nu}) \to H(\tilde{x}).$$

Since for every ν, $G_{k_\nu}(x^{k_\nu}) - L_{k_\nu}(x^{k_\nu}) = G_{k_\nu}(x^{k_\nu}) - t^{k_\nu} \geq G(x) - H(x)\ \forall x \in M$, it follows that

$$G(\tilde{x}) - H(\tilde{x}) \geq G(x) - H(x) \quad \forall x \in M.$$

Therefore, $\tilde{x}$ solves (Q). The rest is clear. □

Consequence *Given a tolerance* $\varepsilon > 0$ *if we stop the Algorithm at iteration* k *such that*

$$G_k(x^k) - t_k \geq \gamma_k - \varepsilon$$

then x^k *is a global* ε*-optimal solution (in the sense that* $G(x^k) - H(x^k) \leq \max \{G(x) - H(x) : x \in M\} - \varepsilon$).

6. Computational Experience

The above method is a branch and bound algorithm in R^2, where branching is by triangular subdivision and bounding involves solving linear programs LP(L, S) in only three variables. Computational experience with branch and bound algorithms

for d.c. optimization suggests that such algorithms should be efficient even for fairly large n.

In the special case of Weber's problem with attraction and repulsion, our branch and bound algorithm differs from the one in [9] not only in the subdivision method (triangular rather than rectangular) but also in the subdivision rule (normal rather than exhaustive).

In order to evaluate the efficiency and robustness of our algorithm, we implemented the procedure in double precision FORTRAN 77 on a SUN SPARC 20 Model 50, rated at 143.2 *MIPS*, 30.6 *MFLOPS*, and 78.3 *SPECfp92*. The user points are generated randomly from a uniform distribution in the unit square $(0,1)\times(0,1)$. The maximal attraction, the rate of decay and weight for each point are selected from a uniform distribution in $(0,30)$, $(0,3)$ and $(0,2)$, respectively. We set the convergence tolerance to $\epsilon = 10^{-5}$ and use E04MBF in the NAG library as our LP solver.

In the following tables, we report the total number of points n, the number of repulsion points n^-, the average number of iterations μ_{Iter} and CPU times μ_{CPU}, and their standard deviations $\sigma_{\text{Iter}}, \sigma_{\text{CPU}}$, respectively. For each (n, n^-) pair, we randomly generated *thirty* problems according to the parameters specified above.

Tables 6.1 and 6.2 list the results for the Euclidean attraction and repulsion case: $q_j(h_j(x)) = \alpha_j - \omega_j\|x - a^j\|$, $j \in J_1 \cup J_2$. These tables include all of the (n, n^-) combinations of Tables 5 and 6 in [9].

Because the points were uniformly distributed on the square and the attraction and repulsion forces are both functions of Euclidean distances, the solution point $\tilde{x}$ is very close to the center $(1/2, 1/2)$ when $n^-/n \leq 1/2$ and it is very close to a vertex of the unit square otherwise. Thus, the problem is most difficult when $n^-/n = 1/2$ and becomes considerably easier when $n^-/n > 1/2$. This is reflected by the sharp drop in the run times in the latter case. An intuitive explanation for this behavior is the following. When $n^-/n > 1/2$, the repulsion force exceeds the attraction force and pushes the solution to the boundary. Since the repulsion force diminishes with distance, the expected repulsion to the corners is less than that to other boundary points. Thus all four corners are good solutions for n large. When $n^-/n \leq 1/2$, the attraction forces control the solution. If we view the location of the facility as fixed and the locations of the points as random, then the distances from the points to the facility are random variables and the minimum total expected distance will occur if the fixed point is in the center. Asymptotically, as $n \to \infty$, the actual behavior should be close to the expected behavior.

We note that our procedure does not *a priori* exploit the knowledge of where the solution is likely to be. In order to show that our algorithm does not favor problems with center solutions, we generated user points as follows: one half of the total points are generated uniformly in $(0, 0.5) \times (0, 0.5)$, and the others are in $(0,1) \times (0,1)$. The results are exhibited in Table 6.3. Notice that the run times are comparable to those in Table 6.1, while the solution points were not in the center

Table 6.1. Computational results for $100 \leq n \leq 500$ points

n	n^-	μ_{Iter}	σ_{Iter}	μ_{CPU}	σ_{CPU}
100	10	57.27	20.88	0.62	0.22
100	20	77.70	22.24	0.86	0.24
100	30	112.53	44.74	1.28	0.53
100	40	180.13	64.76	2.10	0.79
100	50	165.87	154.13	2.00	1.96
100	75	8.67	1.21	0.11	0.02
200	10	42.20	11.92	0.71	0.19
200	20	55.60	11.83	0.92	0.20
200	30	67.13	16.33	1.11	0.27
200	40	81.17	18.21	1.34	0.30
200	50	91.47	25.26	1.52	0.42
200	60	106.67	34.57	1.78	0.58
200	80	188.40	53.69	3.22	0.94
200	100	245.30	151.64	4.28	2.68
200	150	9.07	0.78	0.18	0.02
500	10	43.70	12.21	1.48	0.41
500	20	46.40	13.75	1.57	0.46
500	30	47.40	10.70	1.60	0.35
500	40	49.97	13.56	1.66	0.43
500	50	56.43	12.69	1.91	0.42
500	100	81.30	21.91	2.76	0.74
500	150	106.30	38.88	3.62	1.31
500	200	214.83	74.23	7.37	2.56
500	250	338.07	214.20	11.83	7.61
500	375	9.70	0.70	0.37	0.03

of the square when $n^-/n \leq 1/2$. As expected, even for these problems, the corners solved the problems when $n^-/n > 1/2$.

Our approach allows the incorporation of more general attraction and repulsion functions. Table 6.4 exhibits the results for the exponential attraction and repulsion case: $q_j(h_j(x)) = \omega_j \, \mathrm{e}^{-\theta_j \|x - a^j\|}$, $j \in J_1 \cup J_2$. Note that the run times are from 3 to 5 times longer, on average, than the Euclidean case in Table 6.1. This is due to the increased number of linear functions needed to approximate the convex exponential functions.

We also tested the procedure on the mixed case: $q_j(h_j(x)) = \alpha_j - \omega_j \|x - a^j\|$, $j \in J_1$, and $q_j(h_j(x)) = \omega_j \, \mathrm{e}^{-\theta_j \|x - a^j\|}$, $j \in J_2$. Table 6.5 exhibits the results. Note that when $n^-/n \leq 1/2$, the run times are comparable to the all Euclidean case in Table 6.1. However, when $n^-/n > 1/2$, the run times are comparable to the all exponential case in Table 6.4. This suggests that the solution difficulty is governed by the distance measure of the dominant points.

Table 6.2. Computational results for $1,000 \leq n \leq 100,000$ points

n	$\bar{n}$	μ_{Iter}	σ_{Iter}	μ_{CPU}	σ_{CPU}
1,000	10	35.87	9.93	2.28	0.61
1,000	20	40.43	12.96	2.56	0.81
1,000	40	47.13	9.86	2.97	0.61
1,000	50	50.03	9.71	3.15	0.60
1,000	100	54.80	13.50	3.46	0.83
1,000	200	79.07	21.70	4.96	1.33
1,000	250	88.30	19.62	5.55	1.23
1,000	300	99.10	21.00	6.21	1.30
1,000	400	210.43	38.39	13.23	2.42
1,000	500	367.30	195.09	23.53	12.60
1,000	750	9.70	0.70	0.68	0.05
5,000	10	34.13	11.12	10.12	3.20
5,000	50	37.33	7.82	11.00	2.25
5,000	100	40.93	7.86	12.05	2.25
5,000	500	55.20	11.57	16.12	3.31
5,000	1,000	83.57	20.90	24.37	6.04
5,000	1,500	105.67	25.08	30.66	7.23
5,000	2,000	234.80	57.44	67.97	16.49
5,000	2,500	609.60	237.81	178.59	69.52
5,000	3,750	10.03	0.72	3.20	0.21
10,000	10	29.60	11.29	17.46	6.44
10,000	50	29.97	8.43	17.61	4.81
10,000	100	34.63	9.33	20.30	5.30
10,000	500	49.40	10.65	28.69	6.10
10,000	1,000	54.50	9.28	31.54	5.26
10,000	2,000	89.07	28.19	51.16	16.06
10,000	5,000	967.70	546.97	556.89	313.91
10,000	7,500	9.93	0.64	6.26	0.36
100,000	1,000	33.20	8.33	192.38	46.82

Table 6.3. Computational results for different user points strucutre

n	$\bar{n}$	μ_{Iter}	σ_{Iter}	μ_{CPU}	σ_{CPU}
100	10	71.03	21.53	0.78	0.24
100	20	87.67	24.89	0.98	0.28
100	50	165.53	98.97	1.96	1.19
100	75	6.60	1.71	0.09	0.03
500	50	50.30	28.72	1.72	0.95
500	100	106.13	22.87	3.60	0.78
500	250	389.00	257.80	13.55	9.13
500	375	6.30	0.47	0.26	0.02

Table 6.4. Computational results for exponential functions

n	n^-	μ_{Iter}	σ_{Iter}	μ_{CPU}	σ_{CPU}
100	10	286.77	102.30	3.67	1.36
100	20	328.13	110.82	4.26	1.49
100	80	42.20	10.00	0.57	0.13
200	20	259.13	78.69	5.07	1.58
200	40	326.63	103.31	6.48	2.09
200	150	57.07	13.89	1.17	0.28
500	50	300.03	81.42	12.14	3.32
500	100	398.53	109.82	16.32	4.60
500	400	49.57	7.45	2.13	0.32

Table 6.5. Computational results for mixed functions

n	n^-	μ_{Iter}	σ_{Iter}	μ_{CPU}	σ_{CPU}
100	10	66.17	21.21	0.73	0.23
100	20	78.77	23.25	0.88	0.26
100	80	58.07	16.03	0.75	0.21
200	20	59.03	17.82	1.00	0.29
200	40	85.93	22.47	1.48	0.39
200	150	128.37	60.93	2.55	1.24
500	50	66.03	16.42	2.28	0.56
500	100	82.30	15.56	2.95	0.56
500	400	75.07	15.58	3.29	0.67

From the above tables, we can see that the CPU time depends on the total number of points n and the ratio n^-/n. When n is fixed, more repulsion points will increase the CPU time until $n = 2n^-$. When n^-/n is fixed, the average number of iterations is almost the same for all cases, while the CPU time increases with n.

7. Extensions

In this section we briefly report on the extension of our problem (P) to more general location problems; namely, limited distance and conditional location problems ([1], [3], [4]). These location problems can be formulated as

$$\max \left\{ F'(x) = \sum_{j \in J_1} q_j[d_j(x)] - \sum_{j \in J_2} q_j[d_j(x)] : \; x \in M \right\}, \qquad (P')$$

where $d_j(x) = \min\{h_j(x) = \|x - a^j\|, d^j\}$.

Table 7.1. Computational results for limited distance location problems

n	n^-	μ_{Iter}	σ_{Iter}	μ_{CPU}	σ_{CPU}
50	10	163.60	69.66	1.82	0.93
50	20	155.90	51.54	1.70	0.62
50	25	166.63	61.25	1.83	0.77
50	30	145.17	53.85	1.61	0.69
100	10	183.90	60.19	2.59	0.94
100	20	196.27	66.53	2.73	1.10
100	30	209.47	69.87	2.89	1.12
100	40	213.40	57.17	2.93	0.91
100	50	272.27	90.38	3.88	1.46
100	80	130.93	40.89	1.76	0.60
500	50	326.23	80.62	12.80	3.33
500	100	365.90	101.74	14.53	4.34
500	200	521.87	155.53	21.34	6.92
500	250	556.30	152.82	22.61	6.64
500	400	129.10	13.45	5.02	0.53

Table 7.2. Computational results for conditional location problems

n	n^-	μ_{Iter}	σ_{Iter}	μ_{CPU}	σ_{CPU}
50	10	159.33	68.87	1.68	0.83
50	20	202.37	56.11	2.18	0.70
50	25	228.87	81.19	2.53	1.02
50	30	223.53	85.92	2.50	1.09
100	10	196.13	74.56	2.72	1.20
100	20	216.40	79.82	3.05	1.28
100	30	241.37	89.30	3.45	1.45
100	40	285.17	92.53	4.15	1.54
100	50	323.73	129.41	4.81	2.26
100	80	420.70	192.31	6.96	4.24
500	50	251.30	71.74	9.80	2.89
500	100	327.37	99.99	12.98	4.23
500	200	569.00	208.26	23.57	9.53
500	250	879.03	267.94	37.48	12.71
500	400	636.57	109.61	26.90	5.29

For the limited case, $d^j > 0$ $(j \in J_1 \cup J_2)$ is a given constant. For the conditional case, $d^j = \min\{\|y^i - a^j\| : i = 1, 2, \ldots, m\}$, where y^i $(i = 1, \ldots, m)$ are given points at which existing facilities are located.

For these problems we can prove the following result.

LEMMA 5. *Let* $h(x) = \|x - a\|$, $d(x) = \min\{h(x), d\}$, *where* $d > 0$ *is constant. If* $q : R_+ \to R$ *is a convex nonincreasing function such that* $q^+(0) > -\infty$, *then* $q[d(x)]$ *can be expressed as the d.c. function:*

$$q[d(x)] = g(x) - Ch(x),$$

where $g(x)$ *is a convex function and* C *is a positive constant satisfying* $C \geq |q^+(0)|$.

Hence, we can extend our algorithm to limited distance and conditional location problems. Below we report on our preliminary computational experience with these models.

Table 7.1 exhibits the results for limited distance location problems with the d^j $(j = 1, \ldots, n)$ generated uniformly in $(0.25, 0.5)$. Note that when $n^-/n \leq 1/2$, the run times are within 25% (lower and higher) of the times for the all exponential case in Table 6.4. However, when $n^-/n > 1/2$, the run times are 2 to 4 times longer than the all exponential case. For these latter problems, even though they still exhibit near corner solutions, the problem is more difficult than the unlimited distance location case. This can be explained as follows. When iterates are far away from users near the optimal location, the attraction force is constant for those users and stronger for users near the current iterate. A similar statement can be made for repulsion forces. Thus more iterations are needed to approach the region of attraction of the optimal solution. Note that problems with small d^j values take longer to solve. We report the results for the conditional location problems in Table 7.2. In this case, we choose $m = 5$ and the existing facilities points are selected uniformly from $(0, 1) \times (0, 1)$. Note that when $n^-/n \leq 1/2$, the run times are within 60% (mostly higher) of the run times for the limited distance case in Table 7.1. However, when $n^-/n > 1/2$, the run times are from 50% to 5 times longer than the limited distance case. It appears then that the most difficult problem for our algorithm is the conditional location problem. This can be explained by the fact that the constants d^j, which depend on randomly generated points y^i $(i = 1, \ldots, m)$, were smaller, on average, than those in Table 7.1.

8. Concluding Remarks

In this paper, we proposed a general algorithm for solving single facility location problems using d.c. optimization. The procedure was implemented and tested on several thousand problems and the details of these experiments are reported above. We demonstrated that the procedure is robust with respect to the location of the optimal solution and can handle more general objective functions than Euclidean distances. We also extended the procedure to solve limited distance and conditional location problems, and conclude from our computational experience that conditional location problems are the most difficult to solve. Finally, because our procedure generates upper bounds, we can imbed it in a search procedure for locating the single facility in the union of a finite number of convex polygons in the spirit of Chen *et al.* [2].

Appendix: Computation of Subgradients

The above algorithm involves in Step 3c the computation of a vector $p^k \in \partial H(x^k)$, where, for $C_j \geq -q_j^+(0) > 0$, we have

$$H(x) = \sum_{j \in J_2} g_j(x) + \sum_{j \in J_1} C_j h_j(x),$$
$$g_j(x) = q_j[h_j(x)] + C_j h_j(x),$$

so that

$$H(x) = \sum_{j \in J_2} \{q_j[h_j(x)] + C_j h_j(x)\} + \sum_{j \in J_1} C_j h_j(x).$$

In this Appendix we discuss how to compute this subgradient. Below, we use the notation $\langle x, y \rangle$ for inner product $x^T y$ between two vectors x and y.

LEMMA 6. *A subgradient of the function $h(x) = \|x - a\|$ at point x^0 is the vector*

$$\pi = \begin{cases} \dfrac{x^0 - a}{\|x^0 - a\|} & \text{if } x^0 \neq a, \\ 0 & \text{if } x^0 = a. \end{cases}$$

Proof. If $x^0 \neq a$ then the function $h(x)$ is differentiable at x^0, so its subgradient at x^0 is just its gradient at this point. Since $h^2(x) = (x_1 - a_1)^2 + (x_2 - a_2)^2$, we have, for $i = 1, 2$, $2h(x^0)\frac{\partial h(x^0)}{\partial x_i} = 2(x_i^0 - a_i)$. Hence $\frac{\partial h(x^0)}{\partial x_i} = \dfrac{x_i^0 - a_i}{h(x^0)}$, and so

$$\nabla h(x^0) = \frac{x^0 - a}{\|x^0 - a\|}.$$

If $x^0 = a$, then $h(a) = 0$, so $h(x) - h(a) = h(x) \geq 0 \ \forall x$, and hence, $h(x) - h(a) \geq \langle 0, x - a \rangle \ \forall x$. This implies that the vector 0 is a subgradient of $h(x)$ at a $(0 \in \partial h(a))$. □

LEMMA 7. *Let $q : R_+ \to R$ be a convex nonincreasing function such that $q^+(0) > -\infty$, and let $g(x) = q[h(x)] + Ch(x)$ where $C \geq |q^+(0)|$. If $\pi \in \partial h(x^0)$, then*

$$[q^+(h(x^0)) + C]\pi \in \partial g(x^0)$$

where $q^+(t)$ denotes, as usual, the right derivative of $q(t)$ at t.

Proof. Let $t_0 = h(x^0)$. By convexity of the function $\tilde{q}(t) = q(t) + Ct$ (see the proof of Lemma 1), we have $\tilde{q}^+(t_0)(t - t_0) \leq \tilde{q}(t) - \tilde{q}(t_0) \ \forall t$, i.e. $(q^+(t_0) + C)(t - t_0) \leq (q(t) + Ct) - (q(t_0) + Ct_0), \ \forall t$. Hence, letting $t = h(x)$, we obtain

$$[q^+(h(x^0)) + C](h(x) - h(x^0)) \geq g(x) - g(x^0) \quad \forall x. \tag{4}$$

But, $\pi \in \partial h(x^0)$ implies that

$$\langle \pi, x - x^0 \rangle \leq h(x) - h(x^0),$$

and since $q^+(h(x^0)) + C \geq q^+(0) + C \geq 0$, we have

$$[q^+(h(x^0)) + C]\langle \pi, x - x^0 \rangle \leq [q^+(h(x^0)) + C](h(x) - h(x^0)).$$

This, together with (4), yields

$$\langle [q^+(h(x^0)) + C]\pi, x - x^0 \rangle \leq g(x) - g(x^0) \quad \forall x,$$

as was to be proved. □

EXAMPLE I. If $q(t) = \alpha - wt$, with $w > 0$, then for $C \geq w$, a subgradient of the function $g(x) = q(\|x - a\|) + C\|x - a\|$ at point $x^0 \in M$ is

$$p = \begin{cases} (C - w)\dfrac{x^0 - a}{\|x^0 - a\|} & \text{if } x^0 \neq a, \\ 0 & \text{if } x^0 = a. \end{cases}$$

EXAMPLE II. If $q(t) = w\,\mathrm{e}^{-\theta t}$, with $w > 0$ and $\theta > 0$, then for $C \geq \theta w$, a subgradient of the function $g(x) = w\,\mathrm{e}^{-\theta\|x-a\|} + C\|x - a\|$ at point $x^0 \in M$ is

$$p = \begin{cases} [C - \theta w\,\mathrm{e}^{-\theta\|x^0 - a\|}]\dfrac{x^0 - a}{\|x^0 - a\|} & \text{if } x^0 \neq a, \\ 0 & \text{if } x^0 = a. \end{cases}$$

LEMMA 8. *Let $U(x), V(x)$ be convex functions and $A(x) = U(x) + V(x)$. If $p = u + v$, with $u \in \partial U(x^0)$, $v \in \partial V(x^0)$, then $p \in \partial A(x^0)$.*

Proof. Immediate. □

Using the above results it is easy to compute a subgradient of $H(x)$ at $x^0 \in M$ if, for every j, $h_j(x) = \|x - a^j\|$ and $q_j(t)$ is as in Example I or II. For instance:

PROPOSITION 3. *Assume that for every j : $h_j(x) = \|x - a^j\|$, while $q_j(t) = \alpha_j - w_j t$ with $w_j > 0$. Then a subgradient of $H(x)$ at x^0 is*

$$p = \sum_{j \in J_2 \cap N(x^0)} [C_j - w_j]\frac{x^0 - a^j}{\|x^0 - a^j\|} + \sum_{j \in J_1 \cap N(x^0)} C_j \frac{x^0 - a^j}{\|x^0 - a^j\|},$$

where $N(x^0) = \{j : x^0 \neq a^j\}$.

PROPOSITION 4. *Assume that for every* j : $h_j(x) = \|x - a^j\|$, *while* $q_j(t) = w_j \mathrm{e}^{-\theta_j t}$ *with* $w_j > 0$ *and* $\theta_j > 0$ *is the rate of decay of its attraction. Then a subgradient of* $H(x)$ *at* x^0 *is*

$$p = \sum_{j \in J_2 \cap N(x^0)} [C_j - \theta_j w_j e^{-\theta_j \|x^0 - a^j\|}] \frac{x^0 - a^j}{\|x^0 - a^j\|} + \sum_{j \in J_1 \cap N(x^0)} C_j \frac{x^0 - a^j}{\|x^0 - a^j\|}.$$

REMARK. In some models, polyhedral gauges are used instead of the Euclidean norm.

Let $h(x) = \varphi(x - a)$, where $\varphi(x)$ is a polyhedral gauge, i.e. a nonnegative-valued function of the form $\varphi(x) = \max_{i \in I} \langle c^i, x \rangle$, $|I| < +\infty$, such that $B = \{x : \varphi(x) \leq 1\}$ is a polytope containing 0 in its interior. It can be proved that $0 \in \partial h(a)$, while for $x \neq a$, we have $\partial h(x) = \mathrm{conv}\{c^i : h(x) = \langle c^i, x - a \rangle\}$.

References

1. Chen, R. (1988), Conditional minisum and minimax location-allocation problems in Euclidean space, *Transportation Science* **22**, 157–160.
2. Chen, P., Hansen, P., Jaumard B. and Tuy, H. (1992), Weber's problem with attraction and repulsion, *Journal of Regional Science* **32**, 467–486.
3. Chen, P., Hansen, P., Jaumard, B. and Tuy, H. (1994), Solution of the multisource Weber and Conditional Weber Problems by D.-C. Programming, Technical Report # G-92-35, Revised March 1994, GERAD, University of Montreal, Montreal, Canada.
4. Drezer, Z., Mehrez, A. and Wesolowsky, G.(1991), The facility location problem with limited distances, *Transportation Science* **25**, 183–187.
5. Drezner, Z., and Wesolowsky, G. (1990), The Weber problem on the plane with some negative weights, *INFOR* **29**, 87–99.
6. Horst, R. and Tuy, H. (1993), *Global Optimization*, Kluwer Academic Press, second edition, Dordrecht, The Netherlands.
7. Idrissi, H., Loridan, P. and Michelot, C. (1988), Approximation of solutions for location problems, *Journal of Optimization Theory and Applications* **56**, 127–143.
8. Love, R. E., Morris, J. G. and Wesolowsky, G. O. (1988) *Facilities Location: Models and Methods*, North-Holland, Amsterdam.
9. Maranas, C. D. and Floudas, C. A. (1994) A global optimization method for Weber's problem with attraction and repulsion, in *Large Scale Optimization: State of the Art*, eds. W. W. Hager, D. W. Hearn and P.M. Pardalos, Kluwer Academic Publishers, Dordrecht, The Netherlands, 259–293.
10. Rockafellar, R. T. (1970), *Convex Analysis*, Princeton University Press, Princeton, NJ.
11. Tuy, H. (1987), Global minimization of a difference of convex functions, *Mathematical Programming Study* **30**, 150–182.
12. Tuy, H. (1990), On a polyhedral annexation method for concave minimization, in *Functional Analysis, Optimization and Mathematical Economics*, eds. L.J. Leifman and J.B. Rosen, Oxford University Press, Oxford, 248–260.
13. Tuy, H. (1991), Effect of the subdivision strategy on convergence and efficiency of some global optimization algorithms, *Journal of Global Optimization* **1**, 23–36.
14. Tuy, H. (1991), Polyhedral annexation, dualization and dimension reduction technique in global optimization, *Journal of Global Optimization* **1**, 229–244.
15. Tuy, H. (1992), On nonconvex optimization problems with separated nonconvex variables, *Journal of Global Optimization* **2**, 133–144.
16. Tuy, H. (1993), D.C. Optimization: theory, methods and algorithms, Preprint, Institute of Mathematics, Hanoi.

17. Tuy, H. (1994), A general d.c. approach to location problems, Preprint, Institute of Mathematics, Hanoi.
18. Tuy, H. and Al-Khayyal, F. A. (1992), Global optimization of a nonconvex single facility problem by sequential unconstrained convex minimization, *Journal of Global Optimization* **2**, 61–71.
19. Tuy, H. and Thuong, N. V. (1988), On the global minimization of a convex function under general nonconvex constraints, *Applied Mathematics and Optimization* **18**, 119–142.
20. Weber, A. (1909), *Ueber den Standort der Industrien,* Tübingen (English translation: C.J Friedrich (translator), 1929, Theory of the Location of Industries, University of Chicago Press, Chicago.
21. Weiszfeld, E. (1937), Sur le point pour lequel la somme des distances de n points donnés est minimum, *Tôhoku Mathematical Journal* **43**, 355–386.