A HEURISTIC METHOD FOR LARGE–SCALE MULTIFACILITY LOCATION PROBLEMS

YURI LEVIN AND ADI BEN-ISRAEL

ABSTRACT. A heuristic method for solving large-scale multi–facility location problems is presented. The method is analogous to Cooper's method [3], using the authors' single facility location method [20] as a parallel subroutine, and reassigning customers to facilities using the heuristic of *Nearest Center Reclassification*. Numerical results are reported.

1. INTRODUCTION

We consider two related problems, the *multiple facility location problem*, and the *clustering problem*.

The *clustering problem* is to partition a given set of objects with d known *attributes* to clusters of *similar* objects, so that objects in different clusters are *dissimilar*. The objects are usually represented as points in the Euclidean space $\mathbb{R}^d$ with the coordinates representing the d attributes in question, and *similar* objects correspond to *nearby* points. In general, clustering problems have difficult, non–convex, objective functions, modeling the similarity within clusters and the dissimilarity between clusters, see, e.g., [15].

The *multiple facility location problem* (abbreviated *MFLP*) is to locate certain *facilities* so as to serve *optimally* a given set of *customers*, whose *locations* and *requirements* are known. The simplest MFLP is where the n *demand points* (customers) and the number of *facilities* m are given[1], and it is required to:

(a) determine the locations of the m facilities (*location* decision), and

(b) assign customers to facilities (*allocation* or *assignment*),

so as to minimize the sum of weighted distances from facilities to assigned points[2]. We assume no interactions between facilities.

The MFLP is identified as a special clustering problem if the sets of customers served by the same facility are considered as clusters. The location stage, i.e. locating each facility optimally within its cluster, is part of the MFLP but in general does not appear in the clustering problem.

[1]See Remark 7 below.

[2]Because *allocation* is part of the problem, the MFLP is often called the *location–allocation problem*, [3], as distinguished from the single facility location problem.

Another difference: the MFLP is usually 2 dimensional (customers and facilities are points in the plane) but clustering problems are d–dimensional, where d, the number of attributes, can be any positive integer. We therefore state our results for general dimension d, although $d = 2$ would suffice for MFLP.

The clustering literature is rich, we mention in particular [10], [16], [11] and [15], for surveys and references. Methods for solving MFLP's are surveyed by Drezner [8], Ghosh and Rushton [13] and Love, Morris and Wesolowsky [21]. Approaches for solving the MFLP include dynamic programming (for one-dimensional problems), Drezner's Algorithm for 2-facility problems, and heuristic methods (see, e.g., [21], [8]). Comparison of heuristics for solving the multisource Weber problem is given in [1].

Because costs are assumed proportional to distances, at optimum each customer is *assigned* to the nearest facility. Exceptions to this rule include the case where facilities have limited capacities (*capacitated MFLP*), and it may be necessary to "split" customers between several facilities. Drezner and Wesolowsky [9] consider shipping costs that depend also on the number of customers served by the facility in question. In [6], Drezner assumes different priorities for the facilities. Hakimi [14], Dobson and Karmakar [5], Darzentas and Bairaktaris [4] consider competitive facilities (facilities competing for customers). In all these cases, customers are not necessarily assigned to the nearest facility.

The best known parallel heuristic method for solving the MFLP is Cooper's method, [3, p. 46]. The method is based on the *Nearest Center Reclassification Algorithm* described in Section 2, with the Weiszfeld method [22] for location single facilities as a subroutine, that can be done in parallel. We call it here the *Cooper–Weiszfeld method*, abbreviated *Cooper–W*.

The authors' *Newton–Bracketing (NB)* method for convex minimization, [20], has been used successfully for solving single facility location problems. The NB method is presented in Section 3. Numerical experiments show that the NB method is better (computes better values in less time) than Weiszfeld's method, see results reported in Section 4.

This suggests an analog of Cooper–W, using the NB method (instead of Weiszfeld's) as a parallel subroutine for solving the single location problems. The proposed method, called the *Cooper-NB method*, is described in Section 5. Numerical results reported in Section 6 illustrate its advantage over the Cooper–W method.

A mathematical programming formulation of the MFLP, given in Section 7, provide interpretation of the proposed method.

The results of this paper are stated for Euclidean distances. However, our methods are applicable to distances derived from other norms (e.g. $\| \ \|_1$ or $\| \ \|_\infty$). In fact, the NB method [20] applies to general convex minimization.

Synonyms used in the literature:

- facilities: *sources, supply points,*

- customers: *destinations, demand points, mass points,*
- requirements: *demands, masses,*
- location problem: *Fermat–Weber problem.*

2. The nearest center reclassification algorithm

By a *center* of a set of points we mean a central location of that set – this concept is vague on purpose, allowing it to mean *average, mass center, optimal facility location,* etc.

Given a set of points $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n\}$, and a positive integer m, it is required to partition $\mathcal{A}$ into m nonempty clusters $\{\Omega_1, \ldots, \Omega_m\}$, so that each cluster Ω_i consists of those points nearer to its center than to the centers of the other clusters $\Omega_k, k \neq i$.

The *Nearest Center Reclassification Algorithm* (abbrevieted *NCRA*) is an iterative method, well–known under the name *Nearest Mean Reclassification,* see, e.g., [12]. The k <u>th</u>–iteration begins with a *partition* (or *clustering*)

$$\Omega^k = \{\Omega_1^k, \ldots, \Omega_m^k\}.$$

The initial partition Ω^0 is selected randomly.

The center $\mathbf{x}_i^k$ of each Ω_i^k is computed (Algorithm 1, step 2), and points $\mathbf{a}_j \in \Omega_i^k$ are reassigned to other clusters if closer to their centers than to $\mathbf{x}_i$, (Algorithm 1, step 4). The algorithm stops (if no reassignments are possible) or proceeds with the new partition $\Omega^{k+1} = \{\Omega_1^{k+1}, \ldots, \Omega_m^{k+1}\}$ reflecting the reassignments.

The general iteration is described as follows:

Algorithm 1 (Nearest Center Reclassification Algorithm. Iteration k).

Given a partition $\Omega^k = \{\Omega_1^k, \ldots, \Omega_m^k\}$ of the set $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$.			
1	**set**	$r := 0$	(the number of reassignments)
2	**compute**	**for** $i = 1, \ldots, m$	the center $\mathbf{x}_i^k$ of Ω_i^k
3	**compute**	**for** $j = 1, \ldots, n$	distances $d_{ji} := \|\mathbf{a}_j - \mathbf{x}_i^k\|$, $i = 1, \ldots, m$
4		**for** $j = 1, \ldots, n$	**if** $\mathbf{a}_j \in \Omega_p^k$ **and** $d_{j\ell} = \min\limits_{i=1,\cdots,m} d_{ji} < d_{jp}$ **then**
			$\Omega_\ell^k := \Omega_\ell^k \cup \{\mathbf{a}_j\}$, $\Omega_p^k := \Omega_p^k \setminus \{\mathbf{a}_j\}$ (reassign $\mathbf{a}_j$)
			$r := r + 1$
			endif
5	**if**	$r = 0$	**stop**
	endif	$\Omega^{k+1} := \Omega^k$	
		$k := k + 1$	**go to 1**

Remark 1. Step 4 (reassignment) in Algorithm 1 may leave a cluster Ω_p^k empty, having "lost" all its customers to nearer facilities. The next partition Ω^{k+1} may therefore have fewer than m nonempty clusters.

3. LOCATION OF A SINGLE FACILITY

The set of customers is denoted by $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n\}$. Distances are Euclidean and costs are proportional to distances. The objective is to locate the facility so as to minimize the total transportation cost.

Let:

- $\mathbf{a}_j = (a_j^1, a_j^2, \ldots, a_j^d)$ be the location of the j th–customer, $j = 1, \ldots, n$,
- c_j = the unit cost of shipping for one mile between the facility and the j th–customer, $j = 1, \ldots, n$,
- q_j = the requirement of the j th–customer.

Denote by $\mathbf{x} = (x^1, x^2, \ldots, x^d)$ the location selected for the facility. Then the (*Euclidean*) *distance* between the facility and the j th–customer is

$$d_j = \sqrt{\sum_{i=1}^{d} (x^i - a_j^i)^2} \quad \text{denoted by} \quad d_j = \|\mathbf{x} - \mathbf{a}_j\| \ .$$

The optimal facility location is the minimizer of the the *total transportation cost*

$$C(\mathbf{x}) = \sum_{j=1}^{n} w_j \|\mathbf{x} - \mathbf{a}_j\| \ , \tag{1}$$

with *weights* w_j given by

$$w_j = c_j\, q_j \ , \quad j = 1, \ldots, n \ . \tag{2}$$

The gradient of C

$$\nabla C(\mathbf{x}) = \sum_{j=1}^{n} w_j \frac{\mathbf{x} - \mathbf{a}_j}{\|\mathbf{x} - \mathbf{a}_j\|} \tag{3}$$

exists for all $\mathbf{x} \notin \mathcal{A}$. If C is differentiable at $\mathbf{x}^*$, then $\mathbf{x}^*$ is an optimal location if and only if [3]

$$\nabla C(\mathbf{x}^*) = \sum_{j=1}^{n} w_j \frac{\mathbf{x}^* - \mathbf{a}_j}{\|\mathbf{x}^* - \mathbf{a}_j\|} = \mathbf{0}. \tag{4}$$

It follows from (4) that $\mathbf{x}^*$ is a convex combination of the points of $\mathcal{A}$,

$$\mathbf{x}^* = \sum_{j=1}^{n} \lambda_j(\mathbf{x}^*)\, \mathbf{a}_j \ , \tag{5}$$

with weights

$$\lambda_j(\mathbf{x}) = \frac{w_j \|\mathbf{x} - \mathbf{a}_j\|^{-1}}{\sum\limits_{k=1}^{n} w_k \|\mathbf{x} - \mathbf{a}_k\|^{-1}} \ . \tag{6}$$

---

[3]Or, more generally, $\mathbf{0} \in \partial C(\mathbf{x}^*)$, using the terminology of convex analysis

The *Weiszfeld Method* [22] for solving this problem is an iterative method, giving the *next iterate* $\mathbf{x}_+$ as a convex combination of $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$,

$$\mathbf{x}_+ := \sum_{j=1}^{n} \lambda_j(\mathbf{x}) \, \mathbf{a}_j \, , \tag{7}$$

with weights $\lambda_j(\mathbf{x})$ computed by (6) for the *current iterate* $\mathbf{x}$. If the Weiszfeld iterates converge to a point $\mathbf{x}^*$, then $\mathbf{x}^*$ is optimal by (4). The method breaks down if $\mathbf{x} \in \mathcal{A}$, where the weights (6) are undefined.

The Weiszfeld method is the standard, and best–known, method for solving problem (1). The convergence of the Weiszfeld method was studied in [18], [17] and [2].

In [20] we proposed an alternative, the *Newton Bracketing* (NB) method, for the approximate solution of the problem (1). The method, based on the *directional Newton method* of [19], works by improving bounds on the minimum value $C_{\min}$ of $C(\mathbf{x})$, rather than by approximating a solution $\mathbf{x}^*$ satisfying the optimality condition (4). The NB method, used here as a subroutine, is outlined next.

An iteration begins with an interval $[L^k, U^k]$, called a *bracket*, containing the minimum value $C_{\min}$,

$$L^k \leq C_{\min} \leq U^k \, . \tag{8}$$

If the bracket is sufficiently small, say

$$U^k - L^k < \epsilon \tag{9}$$

for an acceptable tolerance $\epsilon > 0$, then the current $\mathbf{x}^k$ is declared optimal, and computation stops. Else, select a value M^k inside the bracket, i.e.

$$M^k := \alpha \, U^k + (1 - \alpha) \, L^k \, , \text{ for some } 0 < \alpha < 1, \tag{10}$$

and do one directional Newton iteration [19] for solving

$$C(\mathbf{x}) = M^k \, , \tag{11}$$

to get $\mathbf{x}^{k+1}$. This is Step 3 in Algorithm 2 below. Now there are two cases:

- $C(\mathbf{x}^{k+1}) < C(\mathbf{x}^k)$ in which case the upper bound is updated $U^{k+1} := C(\mathbf{x}^{k+1})$, see Step 4 of Algorithm 2 below, or
- $C(\mathbf{x}^{k+1}) \geq C(\mathbf{x}^k)$ in which case the lower bound is updated $L^{k+1} := M^k$ and the old solution is retained, $\mathbf{x}^{k+1} := \mathbf{x}^k$, see Step 5 of Algorithm 2.

The general iteration of the NB method is then described as follows:

Algorithm 2 (The NB Method. Iteration k)**.**

Given an **iterate** $\mathbf{x}^k$ and a **bracket** $[L^k, U^k]$.			
1	**if**	$U^k - L^k < \epsilon$	**then solution** := $\mathbf{x}^k$, **stop**
2	**endif**	**select**	$M^k := \alpha\, U^k + (1 - \alpha)\, L^k$
3		**do**	$\mathbf{x}^{k+1} := \mathbf{x}^k - \dfrac{C(\mathbf{x}^k) - M^k}{\|\nabla C(\mathbf{x}^k)\|^2}\, \nabla C(\mathbf{x}^k)$
4	**if**	$C(\mathbf{x}^{k+1}) < C(\mathbf{x}^k)$	**set** $U^{k+1} := C(\mathbf{x}^{k+1})$, $L^{k+1} := L^k$
5		**else**	**set** $L^{k+1} := M^k$, $U^{k+1} := U^k$, $\mathbf{x}^{k+1} := \mathbf{x}^k$
	endif	$k := k + 1$	**return**

An initial bracket $[L^0, U^0]$ is given by:

$$U^0 \;=\; C(\mathbf{x}^0) \text{ where } \mathbf{x}^0 \text{ is the initial iterate, and}$$

$$L^0 \;=\; \|\mathbf{a}_i - \mathbf{a}_j\| \min \{w_i, w_j\} \text{ for any two points in } \mathcal{A}.$$

Remark 2.

(a) The NB method is guaranteed to work if the level sets of the convex function $C(\mathbf{x})$ are not "too narrow". One interpretation of this statement is that the condition number[4] of $C(\mathbf{x})$ is less than $13.92 = 1/(7 - \sqrt{48})$, see [20, Theorems 4–5]. Remarkably, the level sets of the function $C(\mathbf{x})$ for location problems tend to be circular as the number of points increases, see e.g. [20, Figure 4]. For random data sets with n points, the condition number of $C(\mathbf{x})$ tends to 1 as $n \to \infty$.

(b) The gradient (3) exists at all points $\mathbf{x} \notin \mathcal{A}$. If the point $\mathbf{x}$ coincides[5] with one of the n points $\mathbf{a}^i \in \mathcal{A}$, then the index i can be dropped from the summation in RHS(3), which means replacing the non-existing gradient $\nabla \left(\|\mathbf{x} - \mathbf{a}^i\| \right)$ by the vector $\mathbf{0}$ in the subgradient $\partial \|\mathbf{x} - \mathbf{a}^i\|$.

(c) There is no need to recompute $\nabla C(\mathbf{x})$ if $C(\mathbf{x}^{k+1}) > C(\mathbf{x}^k)$, i.e. if Step 4 in Algorithm 2 is skipped, in which case only the lower bound L^k is updated. This presents considerable economy since, in our experience, such iterations occur about 75% of the time.

(d) The brackets $[L^k, U^k]$ are on the average halved at each iteration, so that after 10 iterations the bracket has been reduced to about 0.001 of its initial size.

4. Comparison of running times of the NB and Weiszfeld methods

The NB and Weiszfeld methods were compared in [20] and the numerical results reported in [20, Tables 1–3], giving the number of iterations in both methods until a problem is "solved". This comparison seemed sufficient

---

[4]The *condition number* of $C(\mathbf{x})$ is here the ratio $\lambda_{\max}/\lambda_{min}$ of the extreme eigenvalues of the matrix Q corresponding to a quadratic function $\mathbf{x}^T Q \mathbf{x}$ + linear term, approximating $C(\mathbf{x})$ near its minimum

[5]An event of zero probability if the algorithm is used with random data $\mathcal{A}$ and random initial iterate $\mathbf{x}^0$.

n	Weiszfeld	NB
100	0.02	0.01
1000	0.09	0.06
5000	0.27	0.22
10000	0.70	0.46
20000	1.28	0.91
50000	4.15	2.56
100000	7.31	5.04
300000	20.81	15.38
500000	49.95	28.30
1000000	72.50	58.87

TABLE 1. Results of Experiment 1: The CPU times (in Seconds) of the Weiszfeld and NB methods for 10 random problems with n points in $\mathbb{R}^2$.

since the work (number of operations) per iteration in both methods is $O(n\,d)$ (n = number of points, d = dimension of space).

In this section we report the CPU times of the NB and Weiszfeld methods for problems with random data. The comparison is problematic since the Weiszfeld method does not have a natural stopping rule like (9) for the NB method. To get around this difficulty we ran two experiments:

Experiment 1: Run both methods until they give the same value and report the CPU times. The stopping value is the approximate optimal value obtained by the NB method using a tolerance of 10^{-3}. In other words: solve a problem using the NB method, stopping when the bracket $U - L < 10^{-3}$, then solve again using the Weiszfeld method with the given value as the stopping rule. The same initial solutions were used for both methods. The CPU times for 10 problems with random data of size n between 100 and 1000000 are reported in Table 1.

Experiment 2: Run both methods for a fixed number of iterations from the same initial solution, and report the average CPU time per iteration. This experiment gives a direct comparison of the work per iteration in both methods. The results are reported in Table 2; the 4th column gives the percentage of increase of Weiszfeld over NB.

In the above experiments, both the Weiszfeld and NB methods were programmed in MATLAB 6 Version 12, and run on a PC. The codes are available from the authors.

A comparison of the Weiszfeld and NB methods should also note the erratic behavior of the former, illustrated by [20, Example 6], a location

n	Weiszfeld	NB	increase
100	0.0005	0.0005	0 %
1000	0.0030	0.0020	50 %
5000	0.0095	0.0085	12 %
10000	0.0225	0.0180	25 %
20000	0.0386	0.0316	22 %
50000	0.1095	0.0840	30 %
100000	0.2150	0.1770	21 %
300000	0.6490	0.5240	24 %
500000	1.1030	0.9100	21 %
1000000	43.370	35.200	23 %

TABLE 2. Results of Experiment 2: The average CPU time per iteration of the Weiszfeld and NB methods, in 20 iterations, for 10 random problems with n points in $\mathbb{R}^2$.

problem with five points $\mathbf{a}_i$ and corresponding weights w_i given as follows:

$$\mathbf{a}_1 = (-1, -1) \ , \ w_1 = 1$$
$$\mathbf{a}_2 = (-1, 1) \quad , \ w_2 = 1$$
$$\mathbf{a}_3 = (1, -1) \quad , \ w_3 = 1$$
$$\mathbf{a}_4 = (1, 1) \qquad , \ w_4 = 1$$
$$\mathbf{a}_5 = (100, 0) \quad , \ w_5 = 4$$

This simple example, based on [7, p. 277], is difficult for the Weiszfeld method, requiring about a million iterations from the initial solution $\mathbf{x}^0 = (80, 0)$ (or similarly located points on the x–axis). In contrast, the NB algorithm converges reasonably fast, as does Drezner's accelerated Weiszfeld method, [7].

5. A PARALLEL HEURISTIC METHOD FOR SOLVING THE MFLP

Recall the MFLP: Given n *customers* and m, the number of *facilities*, determine the locations of the facilities, and assign the customers to these facilities, so as to minimize the sum of weighted Euclidean distances from facilities to assigned points.

The best known heuristic method for solving the MFPL was proposed by Cooper in [3, p. 46]. It uses the *Nearest Center Reclassification Algorithm* of Section 2, with centers computed using the *Weiszfeld method*. We call it the *Cooper–Weiszfeld method* or *Cooper–W* for short. The Cooper–W Method is inherently parallel since the cluster centers can be computed in parallel.

We propose here an analogous parallel heuristic method, using the *NB Method* for minimization of convex functions (Algorithm 2 above) as the

subroutine for computing the cluster centers. The proposed method is called *Cooper–NB*.

Each iteration begins with a partition of the set of customers $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ into m nonempty disjoint sets (clusters), $\{\Omega_1, \ldots, \Omega_m\}$, where all customers in a cluster are assigned to the same facility (the number of facilities $m \leq n$, the number of customers).

The optimal location $\mathbf{x}_i$ of the facility serving the customers in the $i\underline{\text{th}}$ cluster is then computed for $i = 1, \ldots, m$. This computation can be done in parallel.

Finally, given the location of the facilities $\mathbf{x}_1, \ldots, \mathbf{x}_m$, it may happen that a customer is better off by switching to another cluster whose facility is nearer. We compute for each customer the nearest facility, and if it is different from the currently assigned facility, switch the customer.

We write a (feasible) solution as $\{\Omega, \mathbf{X}\}$ where

$\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_m\}$, a partition,

$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, the locations of facilities for the clusters in Ω.

The general iteration of the Cooper–NB method is described as follows:

Algorithm 3 (The Cooper–NB method. Iteration k).

Given a partition $\Omega^k = \{\Omega_1^k, \ldots, \Omega_m^k\}$ of the set $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$.			
1	**set**	$r := 0$	(the number of reassignments)
2	**compute**	**for** $i = 1, \ldots, m$	the location of facility $\mathbf{x}_i^k$ serving Ω_i^k using Algorithm 2
3		Step 3 of Algorithm 1	
4		Step 4 of Algorithm 1	
5	**if**	$r = 0$	**solution** is $\{\Omega^k, \mathbf{x}^k\}$, **stop**
	endif	$\Omega^{k+1} := \Omega^k$ $k := k + 1$	**go to 1**

Remark 3.
(a) The algorithm starts with an arbitrary partition $\Omega^0 = \{\Omega_1^0, \ldots, \Omega_m^0\}$ of the set $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$. Our numerical experience shows that in most cases the algorithm is not sensitive to the choice of the initial partition, see Section 6.2.
(b) Given the optimal locations $\mathbf{X}^* = \{\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*\}$, the boundaries of the optimal clusters $\{\Omega_1^*, \ldots, \Omega_m^*\}$ are the corresponding piecewise linear bisectors (Voronoi diagram) of these points, see the last diagrams in Figures 1 and 2.

Remark 4. A heuristic method is not guaranteed to work in a controlled fashion in all instances, and the method proposed here is no exception. There are two cases where it may not converge to an optimal solution.

Case 1: Step 4 (reassignment) in Algorithm 3 may leave a cluster Ω_p^k empty, see Remark 1. If the constraint that there should be exactly m facilities is rigid, one or more of the clusters in Ω^{k+1} may be sub–partitioned to give again a partition with m clusters.

Case 2: Step 5 is heuristic in the sense that the condition $r = 0$ (no reassignments) is necessary, but not sufficient, for optimality of $\{\Omega^k, \mathbf{X}^k\}$. Thus the algorithm may stop at a non–optimal solution, see Section 6.2.

Remark 5. Algorithm 3 may be accelerated in two ways.

Option 1: Do less work in Step 2, where m single facility location problems are solved using Algorithm 2, by either relaxing the tolerance ϵ in the stopping rule (9), or by limiting the number of iterations. Indeed, good approximations of the optimal facility locations $\{\mathbf{x}_1^*, \dots, \mathbf{x}_m^*\}$ are needed only at the end, when the partitions $\{\Omega_1, \dots, \Omega_m\}$ are near optimality.

Option 2: Do less work in Steps 3–4, since one needs a nearer, not the nearest, facility to the point $\mathbf{a}_i$ in question.

Remark 6. In most MFLP's, the speed of computation is secondary to the quality of the solution, as an approximation of the optimal decision. On the other hand, there are dynamic location problems where configurations change, and fast decisions are required. In such problems the speed of computation is of primary concern. The method proposed here can be tuned to achieve any balance between speed of computation and quality of solution, see Remark 5.

Remark 7. There are applications where the number of facilities m is not given a priori, but is determined as the minimal number that guarantees a certain level of service. In this case, the algorithm can be run for different m, until the desired level of service is attained.

6. Numerical Experience

This section has four parts.

The performance of the Cooper–NB method (Algorithm 3) is illustrated in the examples of § 6.1.

The stability of Cooper–NB method under different initial partitions is illustrated in § 6.2.

Running–times of the Cooper–NB method and the Cooper–W method are reported in § 6.3.

In § 6.4 we compare the Cooper–NB method (Algoritm 3) with the *Nearest Mean Reclassification Algorithm* (NMRA), which is the NCRA (Algorithm 1) with facilities located at the mean (centroid) of each cluster.

Remark 8. When reporting on a heuristic method, we use *optimal value* for the final value determined by the method, which is at best an approximate optimal value.

FIGURE 1. Illustration of Example 1
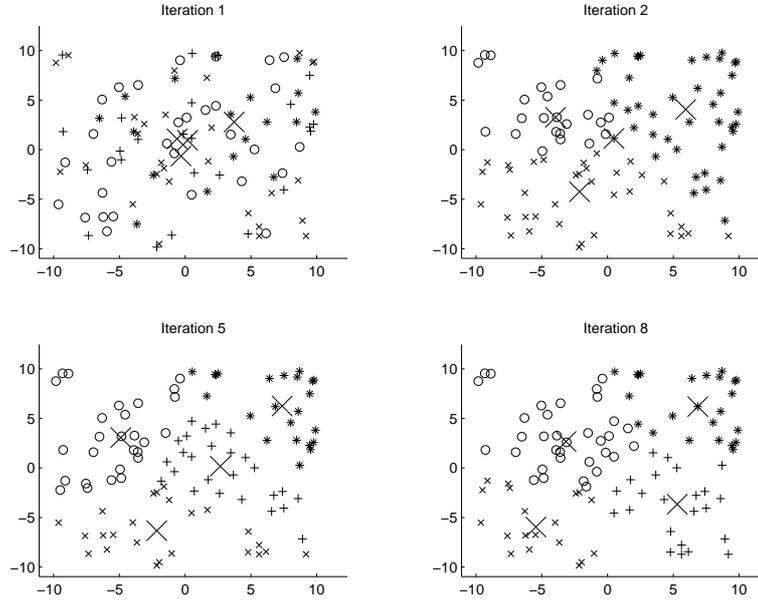
6.1. **Examples.** We illustrate the performance of the Cooper–NB method on three examples. All customers are assumed to have equal demand, taken to be 1.

Example 1. Here $n = 100$ random points in $[-10, 10]^2 \in \mathbb{R}^2$ and $m = 4$ facilities. The algorithm starts with 4 random clusters. Iterations 1,2,5 and 8 of the Cooper–NB method are shown in Figure 1, using different symbols: $\circ, \times, +, *$ for the 4 clusters in each iteration. The location of the facility in each cluster is denoted by large $\times$. The algorithm stopped at iteration 9, when no further reassignments were possible.

Example 2. Here $n = 1000$ random points in $[-10, 10]^2 \in \mathbb{R}^2$ and $m = 4$ facilities. The algorithm starts with 4 random clusters. Iterations 1, 2, 11, 20 are shown in Figure 2, illustrating the progress from the initial chaos to order.

Example 3. ([3, p. 47]). Here $m = 3$ facilities and $n = 15$ demand points (customers) given as follows

Customer	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x–coordinate	5	5	5	13	12	13	28	21	25	31	39	39	45	41	49
y–coordinate	9	25	48	4	19	39	37	45	50	9	2	16	22	30	31

The Cooper–NB method gives an optimum solution to this problem with the minimum cost = 143.1981. The following table shows the optimal solution (location of facilities, and allocation of customers)

FIGURE 2. Illustration of Example 2
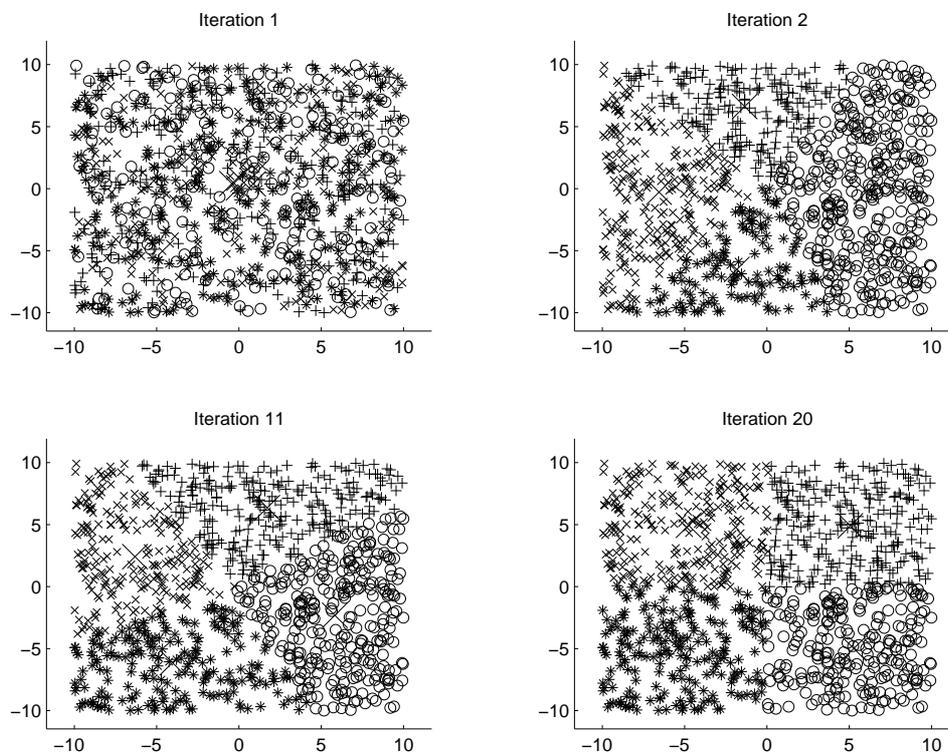
Facility	Location	Assigned Customers
1	(8.9463,14.6382)	1,2,4,5,
2	(20.9989,44.9994)	3,6,7,8,9
3	(40.1947,17.6930)	10,11,12,13,14,15

Iterations 1,2,3,4 are shown in Figure 3.

6.2. **Sensitivity to the initial partition.** Recall that the NCRA (Algorithm 1) and the Cooper–NB method (Algorithm 3) which is a special case, may stop at a non-optimal solution. For example, consider the case of $m = 2$ facilities to serve $n = 1000$ customers, of which 990 live in a city (near one another) and 10 live in a far away village. If we begin with the partition {city,village} the algorithm may stop there, locating one facility in the village, whereas the optimal solution may have both facilities in the city. This known feature of the NCRA is less likely to occur in large–scale problems since the probability of a "bad" random initial partition is small. We illustrate this for Examples 2–3. As expected, the sensitivity to the initial conditions decreases with the problem size.

Example 2 (continued). We tested the Cooper–NB method on this problem 100 times, starting each time with a random initial partition. In all 100 runs the same optimal cost = 3773.4 was obtained, illustrating the stability

FIGURE 3. Illustration of Example 3
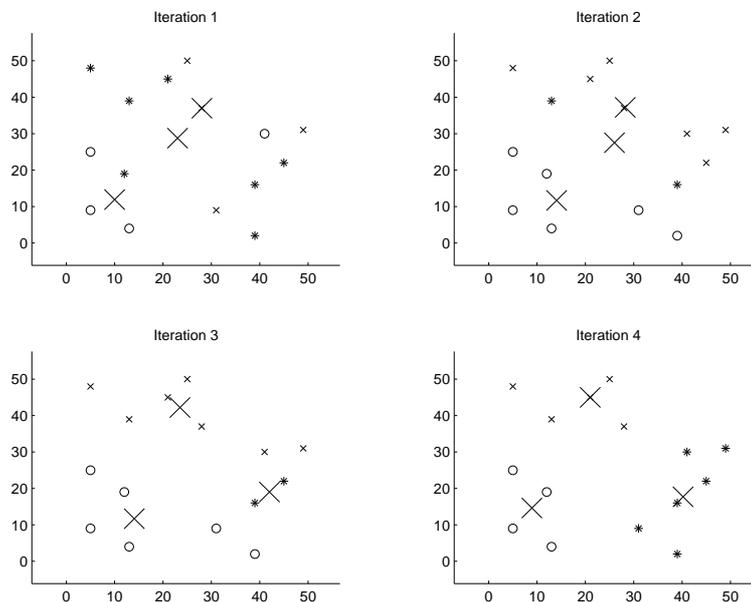
of Cooper–NB method under initial partitions, for large problems.

Example 3 (continued). We tested the Cooper–NB method on this problem 100 times, starting each time with a random initial partition. In 84 runs the optimal total cost = 143.1981 was repeated, and the remaining 16 runs gave values ≤ 145. This shows that even for small problems, the Cooper–NB method is not very sensitive to the initial partition.

6.3. A running–time comparison of Cooper–NB and the Cooper–W methods.

The running–time of a parallel method can be approximated using a single processor by replacing the running–times $\{T_1, T_2, \ldots, T_m\}$ of the parallelizable tasks with

$$T = \max \{T_i : i = 1, \ldots, m\}, \tag{12}$$

called the *equivalent parallel time* of $\{T_1, T_2, \ldots, T_m\}$. We use this device to approximate the running times of parallel implementations of the Cooper–NB method and the Cooper–W method.

Experiment 3: The procedure is as follows:

(a) Generate random problems for different values of n (number of customers) and $m = 5$ (number of facilities).

(b) Solve each problem by the Cooper–NB method and the Cooper–W method until a stable partition (no further reassignments possible) is obtained. The computation of the facility locations for each cluster uses the stopping rule of 20 iterations (of the NB and Weiszfeld methods, respectively).

n (no. of customers)	Cooper–W	Cooper–NB
100	0.061	0.060
1000	0.553	0.533
5000	4.070	3.590
10000	3.545	3.146
20000	17.26	12.97
50000	29.30	24.79

TABLE 3. Results of Experiment 3: Equivalent parallel CPU times (in Seconds) of Cooper–NB and Cooper–W methods for 5 problems with $m = 5$.

(c) Compute the CPU time for each method by adding the equivalent parallel time of the parallelizable tasks in Step 2 of Algorithm 1 to the CPU times of the other steps of Algorithm 1.

The results for 5 random problems are reported in Table 3. In all cases the optimal value obtained by Cooper–NB was better than, or equal to, the value found by the Cooper–W method. These results were repeated with regularity in hundreds of test problems of all sizes.

6.4. **Comparison of Cooper–NB and NMRA.** We compare here the Cooper–NB method with another method based on the Nearest Center Reclassification Algorithm (NCRA), where the center (Step 2 of Algorithm 1 is computed as the cluster mean (or centroid). We call this method the *Nearest Mean Reclassification Algorithm* (NMRA).

We report results for middle–size problems (Example 4) and for the small problem of Example 3. In all tests, the Cooper–NB method and NMRA were started at the same, randomly generated, initial partition.

NMRA requires much less work than the Cooper–NB, and consequently runs faster. In all cases, NMRA brings one very close to optimum.

Example 4. We solved 10 problems with $n = 1000$ random points in $[-10, 10]^2 \in \mathbb{R}^2$ and $m = 4$ facilities. The results (values and iteration counts) are given below.

Cooper–NB		NMRA	
Optimal value	No. of iterations	Optimal value	No. of iterations
3772.3	16	3774.9	14
3855.5	12	3858.9	13
3830.8	13	3831.4	16
3867.5	8	3869.9	6
3815.2	11	3816.9	9
3841.7	12	3848.1	16
3825.7	12	3832.3	8
3756.5	14	3758.9	11
3835.1	9	3837.2	8
3774.8	12	3776.9	14

It is remarkable how close to optimum are the NMRA results.

Example 3 (continued). This problem was solved 5 times by the above methods (starting with 5 random initial partitions). The results, presented in the following table, show again that the NMRA comes close to optimum (in fact, the NMRA values are better than the value obtained in [3] using the Cooper–W method.)

Cooper–NB		NMRA	
Optimal value	No. of iterations	Optimal value	No. of iterations
143.1981	3	144.8724	4
143.1981	3	144.8724	4
143.1981	2	144.8724	4
143.1981	5	144.8724	4
143.1981	2	144.8724	4

Since the NMRA iterations are very cheap, and give an almost–optimal value, a *hyrbid method* is suggested, where one starts with several iterations of NMRA, to be followed by the Cooper–NB method. We tested one such hybrid method (use NMRA until it stops, then continue with Cooper–NB, until it stops too) and it proved superior to Cooper–NB (obtaining the same optimal value quicker) in hundreds of random test problems of varying sizes.

Further research and testing are needed in order to determine the best hybrid method.

7. A MATHEMATICAL PROGRAMMING FORMULATION

We interpret the above results in the context of mathematical programming. Indeed, the location–allocation problem can be formulated as a mixed

integer programming problem:

$$\min_{\mathbf{x}, \lambda} \sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_{ij} \, w_i \, \|\mathbf{x}_j - \mathbf{a}_i\| \qquad (13)$$

subject to

$$\sum_{j=1}^{m} \lambda_{ij} = 1 \, , \ i = 1, \ldots n,$$

$$\lambda_{ij} \in \{0, 1\}, \ i = 1, \ldots n, \ j = 1, \ldots m \, .$$

where
- $\mathbf{x}_j$ is the location of the j th–facility;
- $\mathbf{a}_i$ is the location of the i th–customer;
- w_i are the given weights, and
- $\lambda{ij} = \begin{cases} 1, & \text{if point } i \text{ is assigned to facility } j \\ 0, & \text{otherwise.} \end{cases}$

The coefficients λ_{ij} satisfying the constraints of Problem (13) represent a partition $\Omega = (\Omega_1, \ldots, \Omega_m)$ in the sense that

$$\lambda_{ij} = 1 \iff \mathbf{a}_i \in \Omega_j \, . \qquad (14)$$

Problem (13) can be decomposed into two problems.
Location subproblem (X–problem)
This is Step 2 of Algorithm 3. We denote by $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ the locations of m facilities. Given a partition Ω, or equivalently given the coefficients λ_{ij}, Problem (13) reduces to

$$\min_{\mathbf{X}} \sum_{j=1}^{m} \sum_{\{i : \mathbf{a}_i \in \Omega_j\}} w_i \, \|\mathbf{x}_j - \mathbf{a}_i\| \, , \qquad (15)$$

which decomposes into m single facility location problems,

$$\min_{\mathbf{x}} \sum_{\{i : \mathbf{a}_i \in \Omega_j\}} w_i \, \|\mathbf{x} - \mathbf{a}_i\| \, , \quad i = 1, \ldots, m \, , \qquad (16)$$

that are solved in parallel, using Algorithm 2.
Allocation subproblem (λ–problem)
This corresponds to Step 4 in Algorithm 3. Given the locations $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, the coefficients

$$c_{ij} := w_i \, \|\mathbf{x}_j - \mathbf{a}_i\| \, , \quad i = 1, \ldots n \, , , \quad j = 1, \ldots m \, ,$$

are determined, and Problem (13) reduces to

$$\min_{\lambda} \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}\, \lambda_{ij} \tag{17}$$

$$\sum_{j=1}^{m} \lambda_{ij} = 1\,,\ i = 1,\dots n \tag{18}$$

$$\lambda_{ij} \in \{0,1\}. \tag{19}$$

Because all $c_{ij} \geq 0$, it follows that the binary constraints $\lambda_{ij} \in \{0,1\}$ can be replaced by simpler nonnegativity constraints $\lambda_{ij} \geq 0$. The greedy algorithm can therefore be used to solve the λ–problem (17)–(19). This observation underscores the fact that Step 2 (the $\mathbf{X}$–problem) is the major effort in the proposed method.

Acknowledgement

This paper and the authors benefited from helpful comments by the referees, which are gratefully acknowledged.

References

[1] J. Brimberg, P. Hansen, N. Mladenovic and E. Taillard, Improvements and comparison of heuristics for solving the multisource Weber problem, *Operations Research* **48**(2000), no. 3, p. 444–460

[2] R. Chandrasekaran and A. Tamir, Open questions concerning Weiszfeld's algorithm for the Fermat–Weber location problem, *Math. Prog.* **44**(1989), no. 3 (ser. A), 293–295

[3] L. Cooper, Heuristic methods for location–allocation problems, *SIAM Review* **6** (1964), 37–53

[4] J. Darzentas and P. Bairaktaris, On the set partitioning type formulation for the discrete location problem, *Computers and Operations Research* **13**(1986), 671–679

[5] G. Dobson and U. Karmarkar, Competitive location on a network, *Operations Research* **35**(1987), 565–574

[6] Z. Drezner, Locating a single new facility among existing, unequally attractive facilities, *Journal of Regional Science* **34**(1994), 237–252

[7] Z. Drezner, A note on accelerating the Weiszfeld procedure, *Location Science* **3** (1995), no. 4, 275–279

[8] Z. Drezner, Facility Location: A Survey of Applications and Methods, *Springer-Verlag*, New York, 1995

[9] Z. Drezner and G. Wesolowsky, Allocation of discrete demand with changing costs, *Computers and Operations Research* **26**(1999), 1335–1349

[10] B. Duran and P. Odell, Cluster Analysis, *Springer-Verlag*, Berlin, 1974

[11] B. Everitt, Cluster Analysis, 3nd edition, *Edward Arnold*, London, 1993

[12] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd edition, *Academic Press Inc.*, Boston, MA, 1990

[13] A. Ghosh, G. Rushton, editors, Spatial Analysis and Location–Allocation Models, *New York: Van Nostrand Reinhold*, 1987

[14] S. Hakimi, On locating new facilities in a competitive environment, *European Journal of Operations Research* **12**(1983), 29–35

[15] P. Hansen, B. Jaumard, Cluster analysis and mathematical programming, *Math. Programming* **79**(1997), 191–215

[16] M. Jambu and M. Lebeaux, Cluster Analysis and Data Analysis, *North-Holland Publishing Co.*, Amsterdam, 1983

[17] I.N. Katz, Local convergence in Fermat's problem, *Math. Prog.* **6**(1974), 89–104

[18] H. Kuhn, A note on Fermat's problem, *Math. Prog.* **4**(1973), 98–107

[19] Y. Levin and A. Ben-Israel, Directional Newton methods in n variables, *Math. of Comput.,* **71**(2002), 251-262

[20] Y. Levin and A. Ben-Israel, The Newton Bracketing method for convex minimization, *Computational Optimization and Applications,* **21**(2002), 213-229

[21] R. Love, J. Morris and G. Wesolowsky, Facilities Location: Models & Methods, *North-Holland Publishing Co.*, New York, 1988

[22] E. Weiszfeld, Sur le point par lequel la somme des distances de n points donnés est minimum, *Tohoku Math. J.* **43** (1937), 355–386

YURI LEVIN <**ylevin@business.queensu.ca**>
SCHOOL OF BUSINESS
QUEENS UNIVERSITY
KINGSTON, ONTARIO, CANADA K7L 3N6

ADI BEN-ISRAEL <**bisrael@rutcor.rutgers.edu**>
RUTCOR–RUTGERS CENTER FOR OPERATIONS RESEARCH
RUTGERS UNIVERSITY
640 BARTHOLOMEW RD
PISCATAWAY, NJ 08854-8003, USA