Near-optimal solutions to large scale facility location problems

Francisco Barahona (barahon@watson.ibm.com)
IBM T.J. Watson Research Center,
PO Box 218, Yorktown Heights, New York

Fabián A. Chudak (chudak@watson.ibm.com)
IBM T.J. Watson Research Center,
PO Box 218, Yorktown Heights, New York

Abstract

We investigate the solution of large scale instances of the capacitated and uncapacitated facility location problems. Let n be the number of customers and m the number of potential facility sites. For the uncapacitated case we solved instances of size $m \times n = 3000 \times 3000$; for the capacitated case the largest instances were 1000×1000. We use heuristics that produce a feasible integer solution and use a Lagrangian relaxation to obtain a lower bound on the optimal value. In particular, we present new heuristics whose gap from optimality was generally below 1%. The heuristics combine the volume algorithm and randomized rounding. For the uncapacitated facility location problem, our computational experiments show that our heuristic compares favorably against DUALOC.

Keywords: Volume algorithm, randomized rounding, facility location.

1 Introduction

The study of location of facilities to serve clients at minimum cost has been one of the most studied themes in the field of Operations Research (see, e.g., [15]). In this paper, we focus on two variants of the problem: the *capacitated facility location problem* (CFLP) and the *uncapacitated facility location problem* (UFLP), both of which were extensively treated in the literature (see [8]). We present new heuristics for solving large scale instances of these problems and report on computational experience.

The capacitated facility location problem can be described as follows. There is a set of potential facility locations $\mathcal{F}$; building a facility at location $i \in \mathcal{F}$ has an

associated nonnegative fixed cost f_i, and has a capacity s_i of a certain commodity. There also is a set of customers or demand points $\mathcal{D}$ that require service; customer $j \in \mathcal{D}$ has a demand d_j that must be serviced from the open facilities. If a facility at location $i \in \mathcal{F}$ is used to satisfy part of the demand of client $j \in \mathcal{D}$, the service or transportation cost incurred is proportional to the distance from i to j, c_{ij} . The goal is to determine a subset of the set of potential facility locations at which to open facilities and an assignment of clients to these facilities without violating the capacity constraints so as to minimize the overall total cost, that is, the fixed costs of opening the facilities plus the total service cost. The uncapacitated facility location problem is the simple variant in which each open facility can provide an unlimited amount of commodity (i.e., $s_i = \infty$, for each $i \in \mathcal{F}$).

The uncapacitated facility location problem is known to be NP-complete and due to its widely broad area of applications many heuristics have been devised to solve it. Among these, the most recognized in the literature is the one due to Erlenkotter [10], called DUALOC, which combines simple dual heuristics in a branch and bound framework. Typically, the computational experience that has been reported dealt with problems with several hundreds of potential facility locations as well as several hundreds of customers. In contrast, in this paper we present a heuristic designed to deal with larger instances. We report our computational experience with problems with up to 3000 potential facility locations and similar number of customers.

Most of the previous computational work on the UFLP problem focused on finding *optimal* solutions. For the larger instances we investigated, however, we focused on finding *approximate* solutions, with a relative error, say, of no more than 1%. On one hand, in practice, the data itself is not error-free. On the other hand, enumerative methods such as branch and bound, may require a prohibitive amount of resources (such as time and/or memory). For example, DUALOC spent 60 hours to find an optimal solution to an instance with 1500 points, while our heuristic found a solution within 1% in about 20 minutes.

The state of the art of solving the capacitated facility location problem is less uniform in the sense that there is no one heuristic known to always work well in practice. Part of the reason is that the linear programming relaxation (see below) is known not to be tight (both theoretically [17], and experimentally for small special instances [9]). We refer the reader to the expositions of [9] and [1]. As in the UFLP, we focused on approximate solutions for the problem. In contrast with previous work, which dealt with smaller instances, we investigated instances with up to 1000 facility and demand points.

Our new heuristics are based on the *Volume Algorithm*, introduced in [4], to approximately solve a linear programming relaxation and *Randomized Rounding* [16] to find feasible integer solutions. For the uncapacitated case we use a sophisticated variant of randomized rounding, presented in [6, 7]. Our algorithm provides both an integer solution to the UFLP and a lower bound on the optimal value. Our results compare favorably against partial outputs of DUALOC. For the capacitated case,

we use a simpler variant of randomized rounding and also require a subroutine to solve a transportation problem. A feature of our heuristics is that they can be easily parallelized at almost optimal speed-up, thus they can be used to solve efficiently much larger instances than the ones reported here.

One of the simplest linear programming relaxations for the CFLP is the following (due to Balinsky [3] for the UFLP, and extended to the general case by many authors). For future reference we call it P for primal.

$$\text{Minimize} \quad \sum_{j\in\mathcal{D}}\sum_{i\in\mathcal{F}} d_j c_{ij} x_{ij} + \sum_{i\in\mathcal{F}} f_i y_i \tag{1}$$

$$\begin{aligned}
(\mathsf{P}) \qquad \text{subject to} \qquad \sum_{i\in\mathcal{F}} x_{ij} &= 1 \ , && \text{for each } j \in \mathcal{D}, && (2)\\
\sum_{j\in\mathcal{D}} d_j x_{ij} &\le s_i y_i \ , && \text{for each } i \in \mathcal{F}, && (3)\\
x_{ij} &\le y_i \ , && \text{for each } i \in \mathcal{F},\ j \in \mathcal{D}, && (4)\\
y_i &\le 1 \ , && \text{for each } i \in \mathcal{F}, && (5)\\
x_{ij} &\ge 0 \ , && \text{for each } i \in \mathcal{F},\ j \in \mathcal{D}. && (6)
\end{aligned}$$

Any feasible solution with 0-1 y_i values corresponds to a feasible solution to the capacitated facility location problem: $y_i = 1$ indicates that a facility at location $i \in \mathcal{F}$ is open, whereas x_{ij} is the fraction of the demand of client $j \in \mathcal{D}$ that is serviced by the facility built at location $i \in \mathcal{F}$. Inequalities (2) state that each demand point $j \in \mathcal{D}$ must be assigned among the facilities, whereas inequalities (4) say that clients can only be assigned to open facilities. The capacity constraints are guaranteed by inequalities (3). Thus the linear program P is indeed a relaxation of the problem. Throughout, we will use n to denote the number of clients (that is, $n = |\mathcal{D}|$), and m to denote the number of potential facility locations ($m = |\mathcal{F}|$).

For the UFLP the linear programming relaxation P is known to provide excellent lower bounds in practice. Our results seem to confirm this hypothesis for large instances: starting from a primal solution of P we derive a "close" to optimum integer solution. However, since there are nm inequalities (4), solving P becomes prohibitive for commercial LP solvers for instances with, say, $n, m \ge 500$. Many approaches have been taken to deal with this problem (see [8]), and one of the most successful ones is based on subgradient optimization to obtain tight lower bounds. Previous work using subgradient optimization, however, only provided lower bounds, more concretely, a "good" dual solution, failing to provide "good" primal solutions. To tackle this difficulty, the volume algorithm [4] not only provides primal solutions, but also exhibits enhanced convergence properties.

The second ingredient of our heuristic for the UFLP is based on randomized rounding (see [16]). To understand our approach suppose that (x^*, y^*) is an optimal solution to P. Consider the following very simple algorithm: for each facility $i \in \mathcal{F}$ open a

facility at location i with probability y_i^*; then assign each demand point to the closest open facility. This simple algorithm typically works well in practice. For our computations we used the variant of randomized rounding for the UFLP of [6, 7]. This new algorithm was shown to deliver a feasible solution within a factor of 1.74 of optimum for *any instance* of the problem. Interestingly, this algorithm, originally motivated by theoretical research, outperforms the simple randomized rounding described above.

For the CFLP, in contrast, the linear programming relaxation P can be far away from optimum and, thus, no worst-case performance guarantees can be derived just using P. Interestingly, it was widely reported that for the special instances that need to be solved in practice the linear programming relaxation provides very good lower bounds. Motivated by this and given that the difficulties of solving P are almost the same as for the UFLP, we have used the volume algorithm to approximately solve large instances of P. On top of it, motivated now by the success of our computations for the uncapacitated case, we used a simple randomized rounding procedure to obtain good integer feasible solutions.

2 Solving the linear programming relaxation

In this section we describe how to use the volume algorithm to approximately solve the linear programming relaxation P.

2.1 The uncapacitated case

In this case, we can remove the inequalities (3). Let u_j be a dual multiplier for equation j in (2), and $\overline{c}_{ij} = d_j c_{ij} - u_j$. If we dualize equations (2), a lower bound $L(u)$ is

$$L(u) = \text{Min} \quad \sum_{j \in \mathcal{D}} \sum_{i \in \mathcal{F}} \overline{c}_{ij} x_{ij} + \sum_{i \in \mathcal{F}} f_i y_i \tag{7}$$

$$\begin{aligned} \text{s.t.} \qquad & x_{ij} \leq y_i, && \text{for each } i \in \mathcal{F},\ j \in \mathcal{D}, && (8)\\ & y_i \leq 1, && \text{for each } i \in \mathcal{F}, && (9)\\ & x_{ij} \geq 0, && \text{for each } i \in \mathcal{F},\ j \in \mathcal{D}. && (10) \end{aligned}$$

One can observe that this decomposes into m independent problems (one for each $i \in \mathcal{F}$). After dropping the index i, their generic form is

$$\begin{aligned} &\text{Min} \quad fy + \sum_j \overline{c}_j x_j \\ &x_j \leq y, \quad j \in \mathcal{D}, \\ &0 \leq y \leq 1, \quad 0 \leq x. \end{aligned}$$

This can be solved as follows. If $\overline{c}_j > 0$ then x_j should be 0 ($j \in \mathcal{D}$). Let

$$\mu = \sum_{j:\ \overline{c}_j \leq 0} \overline{c}_j.$$

If $f + \mu < 0$ then we set $y = 1$ and $x_j = 1$ if $\overline{c}_j \leq 0$. If $f + \mu \geq 0$ then all variables should be 0.

2.2 The capacitated case

As before, we dualize equations (2). Let u be the associated dual multipliers and $\overline{c}_{ij} = d_j c_{ij} - u_j$. A lower bound $L(u)$ is given by the *subproblem*

$$L(u) = \text{Min} \quad \sum_{j \in \mathcal{D}} \sum_{i \in \mathcal{F}} \overline{c}_{ij} x_{ij} + \sum_{i \in \mathcal{F}} f_i y_i \tag{11}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{D}} d_j x_{ij} \leq s_i y_i, \qquad \text{for each } i \in \mathcal{F}, \tag{12}$$

$$x_{ij} \leq y_i, \qquad \text{for each } i \in \mathcal{F},\ j \in \mathcal{D}, \tag{13}$$

$$y_i \leq 1, \qquad \text{for each } i \in \mathcal{F}, \tag{14}$$

$$x_{ij} \geq 0, \qquad \text{for each } i \in \mathcal{F},\ j \in \mathcal{D}. \tag{15}$$

Again this decomposes into m independent problems, their generic form is

$$\begin{aligned} &\min fy + \sum_j \overline{c}_j x_j \\ &\sum_j d_j x_j \leq sy, \\ &x_j \leq y, \quad j \in \mathcal{D}, \\ &0 \leq y \leq 1, \quad 0 \leq x. \end{aligned}$$

This is easy to solve as below. First, any variable x_j with $\overline{c}_j > 0$ is set to 0. Then we can assume that the remaining variables are ordered so

$$\frac{\overline{c}_1}{d_1} \leq \frac{\overline{c}_2}{d_2} \leq \ \cdots \ \frac{\overline{c}_{n'}}{d_{n'}}.$$

Let k be the largest index such that $\sum_{j=1}^{j=k} d_j \leq s$. Let $b(k) = \sum_{j=1}^{j=k} d_j$ and $r = \frac{s - b(k)}{d_{k+1}}$. If $f + \sum_{j=1}^{j=k} \overline{c}_j + \overline{c}_{k+1} r \geq 0$, then we set $y = 0$ and $x_j = 0$ for all j. Otherwise, we set $y = 1$, $x_j = 1$, for $1 \leq j \leq k$, and $x_{k+1} = r$.

2.3 The Volume Algorithm

We have seen that computing $L(u)$ is very easy in both the uncapacitated and the capacitated case. One could use the subgradient method to improve $L(u)$, this gives a lower bound, but it does not give a primal solution. The *Volume Algorithm* was developed in [4] as an extension of the subgradient method [13] to produce primal solutions. Its name comes from a result on linear programming duality that says that one can derive a primal solution from the volumes below the faces that are active when maximizing $L(u)$. We describe this method below.

Volume Algorithm

Step 0. Start with a vector $\overline{u}$ and solve (7)-(10) for the UFLP (or (11)-(15) for the CFLP) to obtain $(\overline{x}, \overline{y})$ and $L(\overline{u})$. Set $t = 1$.

Step 1. Compute v^t, where $v_j^t = 1 - \sum_i \overline{x}_{ij}$, and $u^t = \overline{u} + sv^t$ for a step size s given by (17) below.
Solve (7)-(10) (or (11)-(15)) with u^t. Let (x^t, y^t) be the solution thus obtained. Then $(\overline{x}, \overline{y})$ is updated as

$$(\overline{x}, \overline{y}) \leftarrow \alpha(x^t, y^t) + (1 - \alpha)(\overline{x}, \overline{y}), \tag{16}$$

where α is a number between 0 and 1. This is discussed later.

Step 2. If $L(u^t) > L(\overline{u})$ update $\overline{u}$ as

$$\overline{u} \leftarrow u^t.$$

Let $t \leftarrow t + 1$ and go to Step 1.

Notice that in Step 2 we update $\overline{u}$ only if $L(u^t) > L(\overline{u})$, so this is an ascent method, it has some similarities with the bundle method [14], one difference is that we do not solve a quadratic problem at each iteration.

One difference with the subgradient algorithm is the use of formula (16). If $(x^0, y^0), \ldots, (x^t, y^t)$ is the sequence of vectors produced by (7)-(10) (or (11)-(15)), then

$$(\overline{x}, \overline{y}) = \alpha(x^t, y^t) + (1 - \alpha)\alpha(x^{t-1}, y^{t-1}) + \cdots + (1 - \alpha)^t(x^0, y^0).$$

The assumption that this sequence approximates an optimal solution of (1)-(6) is based on a theorem in linear programming duality that appears in [4]. Roughly speaking, it says that the primal variables can be obtained from the volumes below the faces of the dual polyhedron, with the use of formula (16) we are trying to approximate the computation of these volumes. Notice the exponential decrease of the coefficients of this convex combination, thus later vectors receive a much larger weight than earlier ones. At every iteration the direction depends upon this convex combination, so this is a method with "memory" that does not have the zig-zagging behavior of the subgradient method.

As for the subgradient method [13], the formula for the step size is

$$s = \lambda \frac{UB - L(\overline{u})}{\|v\|^2}, \tag{17}$$

where λ is a number between 0 and 2, and UB is an upper bound for the optimal value.

In order to set the value of λ we define three types of iterations:

- Each time that we do not find an improvement we call this iteration *red*. A sequence of red iterations suggests the need for a smaller step-size.
- If $L(u^t) > L(\overline{u})$ we compute $w_j = 1 - \sum_i x_{ij}^t$ for all j and

  $$d = v^t \cdot w.$$

  If $d < 0$ it means that a longer step in the direction v^t would have given a smaller value for $L(u^t)$, we call this iteration *yellow*.
- If $d \geq 0$ we call this iteration *green*. A green iteration suggests the need for a larger step-size.

At each green iteration we would multiply λ by 1.1. After a sequence of 20 consecutive red iterations we would multiply λ by 0.66.

In order to set the value of α in (16), we solve the following 1-dimensional problem:

$$\begin{aligned} &\text{minimize } \|\alpha w + (1-\alpha)v^t\| \\ &\text{subject to} \\ &\frac{b}{10} \leq \alpha \leq b. \end{aligned}$$

Here w is defined as $w_j = 1 - \sum_i x_{ij}^t$ for all j. We try to minimize the norm of the new vector v^{t+1}, while using bounds to control α. The value b was originally set to 0.1 and then every 100 iterations we would check if $L(u^t)$ had increased by at least 1%, if not we would divide b by 2. When b becomes less than 10^{-5} we keep it constant. Each time that we would decrease b we would notice a decrease in the sum of the primal infeasibilities. This choice of α bears great similarities with the one proposed in [18], the difference is in the bounds $b/10$ and b.

Now we show the typical behavior of the algorithm for the UFLP. In Figure 1 we plot the maximum violation of equations (2) by the primal vector. In Figure 2 we plot the value of the primal objective with bullets and the value of the dual objective with a continuous line.

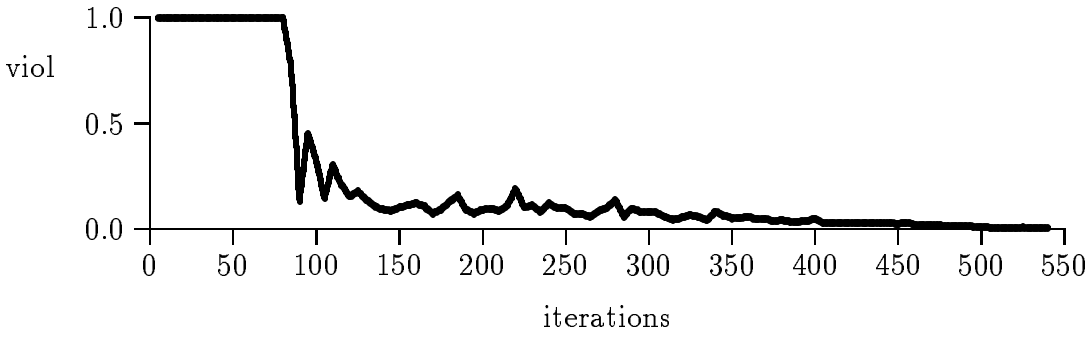

Figure 1

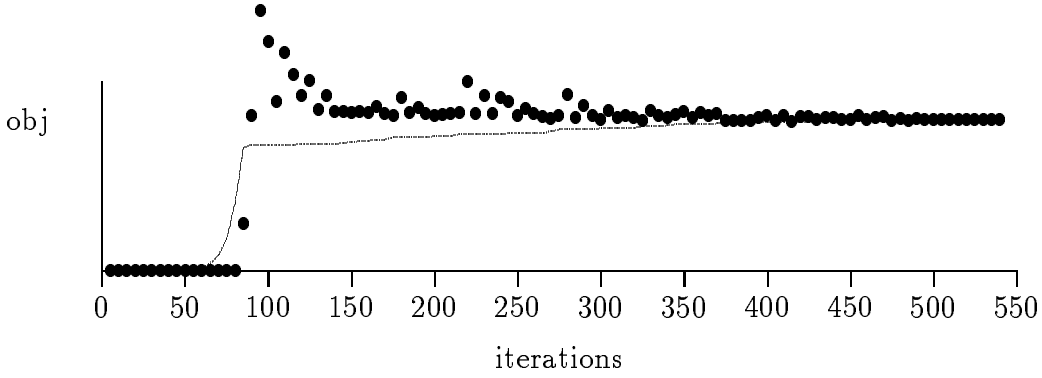

Figure 2

3 Randomized Rounding

In this section we describe the randomized rounding techniques we used to find feasible solutions to the problems.

3.1 Uncapacitated facility location

Here we review the randomized rounding ideas of [6, 7].

Suppose that (x^*, y^*) is an optimal solution to the linear programming relaxation P. Consider first a simple randomized rounding (RR) that opens facility $i \in \mathcal{F}$ with *probability* y_i^*, and assigns each demand point to the closest open facility. Notice that the expected facility cost is exactly $\sum_{i \in \mathcal{F}} f_i y_i^*$. Even though this algorithm performs well in practice, no worst-case performance guarantee is known for it. However, when the distance function c is a metric, a sophisticated variant of randomized rounding, *randomized rounding with clustering* (RRWC), presented in [6, 7], achieves a worst-case performance guarantee of 1.74, that is, it finds a feasible integer solution within a factor of 1.74 of optimum for *any* instance of the problem. Remarkably, this algorithm generally outperforms the simple randomized rounding (see Section 5, and [5]). The

success of the new algorithm relies on limiting the choices of randomized rounding by introducing dependencies in such a way that it uses additional structural information from the optimal solution to the linear programming relaxation P.

We briefly describe the algorithm of [6, 7]. As before, let (x^*, y^*) be an optimal solution to P. For each demand point $j \in \mathcal{D}$, the *neighborhood* of j, $\mathsf{N}(j)$, is the set of facilities $i \in \mathcal{F}$ for which $x^*_{ij} > 0$. The algorithm also makes use of the dual linear program, that can be written as follows: maximize $\{\sum_{j\in\mathcal{D}} v_j : \sum_{j\in\mathcal{D}} w_{ij} \le f_i \ (i \in \mathcal{F}); v_j - w_{ij} \le c_{ij} \ (i \in \mathcal{F}, j \in \mathcal{D}); w_{ij} \ge 0 \ (i \in \mathcal{F}, j \in \mathcal{D})\}$. Let (v^*, w^*) be an optimal dual solution.

First we break the set of points $\mathcal{F} \cup \mathcal{D}$ into clusters such that each demand point belongs to exactly one cluster, but facilities belong to at most one cluster. Also, each cluster has a center j such that all the facilities in the cluster are the neighbors of j; in addition, if k is any demand point in the cluster, and $i_\circ$ is any facility location in the cluster, the distance $c_{i_\circ k}$ is, essentially, at most $2v^*_k + \sum_{i\in\mathcal{F}} c_{ik}x^*_{ik}$. More concretely, the clustering procedure works as follows. Let $\mathcal{S}$ be the set of demand points that have not yet been assigned to any cluster; initially, $\mathcal{S} = \mathcal{D}$. We find the unassigned demand point $j_\circ$ with smallest $(v^*_j + \sum_{i\in\mathcal{F}} c_{ij}x^*_{ij})$-value and create a new cluster *centered* at $j_\circ$. Then all of the unassigned demand points that are fractionally serviced by facilities in the neighborhood of $j_\circ$ (that is, all the demand points $k \in \mathcal{S}$ with $\mathsf{N}(k) \cap \mathsf{N}(j_\circ) \neq \varnothing$) are assigned to the cluster centered at $j_\circ$; the set $\mathcal{S}$ is updated accordingly. We repeat the procedure until all of the demand points are assigned to some cluster (i.e., $\mathcal{S} = \varnothing$). The set of facilities that are in the neighborhood of some center are called "central" facilities, and all the others "noncentral".

Now the algorithm of [6, 7] is a modification of the simple randomized rounding, that makes sure that always there is a facility open in each cluster. If facility i is noncentral, we open facility i with probability y^*_i. Next we treat central facilities as follows. We open exactly one facility per cluster: if j is the center, open neighbor $i \in \mathsf{N}(j)$ with probability x^*_{ij} (note that $\sum_{i\in\mathsf{N}(j)} x^*_{ij} = 1$). Notice next that it is possible that for a central facility i that belongs to the cluster centered at j ,$x^*_{ij} < y^*_i$, and thus if facility i has not been opened by center j, we can open it now independently with probability $y^*_i - x^*_{ij}$. Finally, the algorithm assigns each demand point to the closest open facility. Once again, it is easy to verify that the expected facility cost is $\sum_{i\in\mathcal{F}} f_i y^*_i$. The rest of the analysis is more complicated and can be found in [6, 5, 7].

3.2 Capacitated facility location

For the capacited facility location problem, we use a simpler heuristic that is just based on randomized rounding. If (x^*, y^*) is an optimal solution to the linear programming relaxation P, again we open a facility at location $i \in \mathcal{F}$ with probability y^*_i, independently. Notice, now, that, in contrast with the uncapacitated case, it is possible that there is not enough total capacity to service all the demand. If this is the case, we repeat the random experiment and try again.

Observe next that once we determine which facilities are open, the cheapest assignment of clients can be easily found by solving the following transportation problem. The demand points are on one side of the partition; demand point $j \in \mathcal{D}$ has demand d_j. The open facilities are on the other side; an open facility $i \in \mathcal{F}$ is a source with a surplus s_i. The unit cost of assigning demand point j to an open facility i is c_{ij}. This transportation problem can be easily solved using any specialized network flow code.

Sometimes, we found it convenient to increase the probability of finding feasible solutions as follows. If we have not find any feasible solution after a fixed number of random trials, we multiply the fractional y_i^*'s values by a constant $\gamma \geq 1$, and use $\min\{\gamma y_i^*, 1\}$ as probabilities.

4 Combining the volume algorithm and randomized rounding

In this section we describe a new algorithm that combines both the volume algorithm of [4] and randomized rounding ([16], [6, 7]).

We start describing the new heuristic for the UFLP. The intuition behind the new algorithm is based on the following two simple observations:

- the randomized rounding based procedure described in Section 3.1 takes substantially less time than the volume algorithm
- if we find an integral solution that is sufficiently close to our current lower bound we can stop.

The new heuristic exploits these facts by just running the procedure of Section 3.1 whenever the volume algorithm finds a "good" primal fractional solution. Consider now randomized rounding with clustering (RRWC) of Section 3.1. It was originally devised to work with an optimal solution, but it can also be applied to *any* fractional solution. Thus we can run the randomized heuristic of Section 3.1 (RRWC) when the current primal solution violates equations (2) by less than 0.2. We call this new heuristic V&RRWC, that differs from RRWC in that it uses the algorithm of Section 3.1 many times not just once.

For the CFLP, the situation is not as simple as for the uncapacitated case. While running the volume algorithm, we now run the simple randomized rounding described in Section 3.2 with the current fractional primal solution. However solving a transportation problem does not take a negligible amount of time as in the uncapacitated case. Thus we again run the randomized rounding procedure only when the maximum violation is less than 0.2, but we cannot afford running the subroutine as often as for the UFLP.

5 Computational experiments

In this section we present a representative of our computational experiments.

All of our experiments were conducted on an IBM RISC 6000/7043P-240, with a cpu running at 166MHz.

5.1 The uncapacitated case

We implemented the volume algorithm as described in Section 2.1. Using the solution returned by the volume algorithm, we implemented the simple randomized rounding, RR, that opens facility i with probability y_i^* (here y_i^* is the value returned by the volume algorithm for variable y_i) and assigns each demand point to the closest open facility, and the more sophisticated randomized rounding algorithm described in Section 3.1, RRWC. For each we generated 4000 random trials and recorded the best solution.

To implement randomized rounding with clustering, we used the following observation. If (x^*, y^*) is an optimal primal solution, and if we set $u_j = \min_{i \in \mathcal{F}: x_{ij}^* > 0} c_{ij}$, $(j \in \mathcal{D})$, then the v^*'s in RRWC can be replaced by the u's and this variant of the algorithm also achieves a performance guarantee of 1.74 . Thus we just implemented this simpler algorithm that is only based on the primal fractional solution. In addition, to make sure that our primal solution was indeed feasible, we only extracted the y_i values from the volume algorithm and computed the x_{ij}'s from them in a straightforward way: for a demand point $j \in \mathcal{D}$, we assign j as much as possible to the closest fractionally open facilities respecting the inequalities (4).

We implemented the new heuristic of Section 4, V&RRWC, by running the algorithm of Section 3.1 inside the volume algorithm every 10 iterations, if the maximum violation of the current primal solution was less than 0.2. In each run of the algorithm of Section 3.1, we generated 1000 trials and kept the solution with smallest cost. Whenever the gap between the value of the current best integer solution and the current lower bound was less than 1% or the number of iterations of the volume algorithms reaches 2000, the algorithm stops.

We compared our algorithms against the two heuristics that are the key components of DUALOC [10]: the *dual ascent* procedure (ASCENT) and the more elaborated *dual adjustment* procedure (ADJUST). These two subroutines were extracted from DUALOC-II, provided to us by Erlenkotter [11]. We next describe the dual ascent procedure of Erlenkotter [10]. First note that the v_j's in the dual of P as in Section 3.1 completely determine a dual solution. Starting with $v_j = \min_{i \in \mathcal{F}} c_{ij}$ $(j \in \mathcal{D})$, the dual ascent procedure produces a maximal dual solution (v_j); that is, a solution for which none of the v_j's can be increased without losing feasibility; this is done by increasing one v_j value at a time, when v_j is changed from $c_{i'j}$ to the next largest $c_{i''j}$. Once a maximal dual solution is reached, all of the facilities that are tight in the dual are opened, and each client is assigned to the closest open facility.

In addition, some facilities are closed if the objective function value of the solution improves (see [10] and [11] for details). For the dual adjustment procedure, the idea is the following: given a maximal dual solution, decrease one of the v_j's to its previous value and try to recover the loss in the objective function value of the dual using other v_j's, through applications of the dual ascent procedure. If the amount recovered exceeds the loss, the dual objective function value improves (see [10], [11]). In our experiments, we repeat the dual adjustment procedure until all the demand points are examined.

Given that there are no large scale benchmark instances for the UFLP, we considered instances in which both facility and demand points are distributed uniformly at random in the unit square $[0,1] \times [0,1]$, all the demands d_j are 1, and the facility costs are all the same within some range of values. These instances are not very specialized, are easy to generate, are easy to solve for problems with up to 500 points, are typically considered difficult to solve in practice and exhibit some interesting properties as described by Ahn, Cooper, Cornuéjols and Frieze [2]. More concretely, n points are chosen independently uniformly at random in the unit square, and each point is simultaneously a potential facility location and a demand point. The distances correspond to the usual Euclidean distances in the plane. It was shown in [2] that, when n is large, any enumerative method based on the lower bound P would require the exploration of an exponential number of solutions; also the value of the linear programming relaxation P is, asymptotically in the number of points, about 0.998% of optimum. The goal of our experiments was to test both lower and upper bounds.

For each set of points, we set the fixed costs to be equal to $\sqrt{n}/10$ (type I), $\sqrt{n}/100$ (type II) and $\sqrt{n}/1000$ (type III). These different values provided instances with different properties.

Finally, to prevent numerical problems, we rounded the data to 4 significant digits and made all the data entries integer (this seemed to benefit DUALOC more than any other heuristics tested).

In Table 1, we report typical outputs of our experiments. The first column corresponds to the value of n, that is, the number of clients or facility locations. The second column corresponds to the value of each fixed cost f_i (before rounding up). The following two columns correspond to the values of the dual ascent and dual adjustment procedures of [11]. The next column corresponds to the lower bound provided by the volume algorithm. The following next two columns correspond to the simple randomized rounding algorithm and to the more sophisticated randomized algorithm of Section 3.1. Finally, the last column corresponds to our new heuristic of Section 4.

In Table 2, we report the relative errors of the heuristics tested (in average over 5 runs of each size/fixed cost). Notice that the difficulty of the problems changes from very hard for instances with large facility costs (type I), in which only few facilities can be opened, to fairly easy for instances with small facility costs (type III), in which almost all facilities can be opened.

number of clients n	fixed costs	ASCENT	ADJUST	LOWER BOUND	RR	RRWC	V&RRWC
500	$\sqrt{n}/10$	867381	836869	794686	794686	794686	796439
	$\sqrt{n}/100$	340219	325476	325270	328946	325618	326371
	$\sqrt{n}/1000$	99284	99064	99045	102089	102494	100410
1000	$\sqrt{n}/10$	1681731	1464966	1426160	1433870	1426820	1429330
	$\sqrt{n}/100$	631893	607111	603820	606043	607357	607372
	$\sqrt{n}/1000$	223019	221039	221004	228184	227572	224450
1500	$\sqrt{n}/10$	2476105	2126320	2009890	2095780	2060880	2022070
	$\sqrt{n}/100$	944951	888220	873405	890320	879189	880090
	$\sqrt{n}/1000$	341528	338282	337018	341672	342187	339323
2000	$\sqrt{n}/10$	3174067	2687037	2564950	2673720	2616440	2575990
	$\sqrt{n}/100$	1230267	1135976	1117760	1140000	1123860	1125150
	$\sqrt{n}/1000$	447213	440799	439569	444972	443732	443115
2500	$\sqrt{n}/10$	3790883	3326473	3087610	3201380	3165270	3098390
	$\sqrt{n}/100$	1511424	1375410	1352490	1388520	1368280	1364750
	$\sqrt{n}/1000$	550647	541157	539463	548989	547659	543897
3000	$\sqrt{n}/10$	4568080	3785751	3610970	3732960	3636350	3628860
	$\sqrt{n}/100$	1766589	1635448	1589360	1657590	1635740	1603490
	$\sqrt{n}/1000$	660188	644297	640693	655407	652844	645869

Table 1: Typical values of our experiments.

number of clients n	fixed costs	ASCENT	ADJUST	RR	RRWC	V&RRWC
500	$\sqrt{n}/10$	9.27%	4.01%	0.28%	0.06%	0.73%
	$\sqrt{n}/100$	6.28%	0.18%	1.37%	0.25%	0.52%
	$\sqrt{n}/1000$	0.10%	0.00%	3.26%	3.64%	1.77%
1000	$\sqrt{n}/10$	15.54%	4.18%	1.11%	0.36%	0.54%
	$\sqrt{n}/100$	6.51%	0.99%	0.96%	0.57%	0.56%
	$\sqrt{n}/1000$	1.05%	0.08%	2.92%	1.03%	1.47%
1500	$\sqrt{n}/10$	20.85%	5.27%	3.44%	1.81%	0.66%
	$\sqrt{n}/100$	9.87%	1.68%	2.38%	1.11%	0.93%
	$\sqrt{n}/1000$	1.82%	0.35%	1.68%	1.71%	0.79%
2000	$\sqrt{n}/10$	23.45%	4.85%	3.74%	2.09%	0.75%
	$\sqrt{n}/100$	10.62%	2.43%	1.74%	0.78%	0.76%
	$\sqrt{n}/1000$	02.34%	1.21%	1.68%	1.63%	0.90%
2500	$\sqrt{n}/10$	22.78%	7.12%	5.47%	3.99%	0.71%
	$\sqrt{n}/100$	12.40%	2.20%	3.64%	1.69%	1.06%
	$\sqrt{n}/1000$	2.79%	0.36%	1.58%	1.53%	0.79%
3000	$\sqrt{n}/10$	25.73%	16.79%	3.12%	1.74%	0.71%
	$\sqrt{n}/100$	12.79%	4.27%	3.42%	2.28%	0.93%
	$\sqrt{n}/1000$	3.62%	0.62%	1.79%	1.35%	0.85%

Table 2: Performance of the algorithms.

number of clients n	fixed costs	ADJUST	V&RRWC
500	$\sqrt{n}/10$	13s	91s
	$\sqrt{n}/100$	4s	32s
	$\sqrt{n}/1000$	13s	27s
1000	$\sqrt{n}/10$	42s	480s
	$\sqrt{n}/100$	27s	145s
	$\sqrt{n}/1000$	57s	142s
1500	$\sqrt{n}/10$	151s	1221s
	$\sqrt{n}/100$	37s	364s
	$\sqrt{n}/1000$	129s	308s
2000	$\sqrt{n}/10$	481s	2513s
	$\sqrt{n}/100$	719s	2827s
	$\sqrt{n}/1000$	721s	1748s
2500	$\sqrt{n}/10$	1281s	4784s
	$\sqrt{n}/100$	1376s	3528s
	$\sqrt{n}/1000$	1318s	2604s
3000	$\sqrt{n}/10$	6911s	7512s
	$\sqrt{n}/100$	4074s	5510s
	$\sqrt{n}/1000$	1962s	3922s

Table 3: Comparing running times.

A comparison of running times between the dual adjustment and the heuristic of Section 4 is given in Table 3 (time is measured in seconds).

In Table 4, we show how the number of iterations of the volume algorithm is reduced by the heuristic of Section 4. The stopping criteria for the volume algorithm is when the maximum violation of equations (2) is less that 0.02 and the difference between the lower bound and the primal value is less than 1%. When we run V&RRWC, we stop when the difference between the lower bound and the value of the heuristic solution is less than 1%.

5.2 The capacitated case

We implemented the algorithm described in Section 3.2. As mentioned earlier, solving a transportation problem takes some computational effort. Thus, we only generated 50 random trials every 75 iterations, and only when the maximum violation is less than 0.2. In addition, to increase the chance that our integer solutions are feasible (i.e., have enough capacity), we subsequentially increased the probabilities of opening facilities by a factor of 1.05 if after 25 random trials no feasible solution was found

number of clients n	fixed costs	VOLUME	V&RRWC
500	$\sqrt{n}/10$	750	350/7
	$\sqrt{n}/100$	603	305/18
	$\sqrt{n}/1000$	535	535/45
1000	$\sqrt{n}/10$	990	490/10
	$\sqrt{n}/100$	610	390/19
	$\sqrt{n}/1000$	525	545/46
1500	$\sqrt{n}/10$	1102	590/11
	$\sqrt{n}/100$	702	465 /24
	$\sqrt{n}/1000$	550	340/25
2000	$\sqrt{n}/10$	1110	692/16
	$\sqrt{n}/100$	753	480/21
	$\sqrt{n}/1000$	549	320/22
2500	$\sqrt{n}/10$	1200	850/17
	$\sqrt{n}/100$	870	782/25
	$\sqrt{n}/1000$	555	330/23
3000	$\sqrt{n}/10$	1200	710/12
	$\sqrt{n}/100$	1200	590/24
	$\sqrt{n}/1000$	1200	330/23

Table 4: Reduction on the number of iterations.

or the quality of the solution was not within 2% of our current lower bound. All the transportation problems were solved using an algorithm of Goldberg [12]. Since some of the problems were harder, we used a different stopping criteria. We first looked for an integral solution within 1% of the lower bound. But if after 1500 iterations we have not succeeded, we allow then a gap of 2%.

As for the UFLP there are no large scale benchmark instances for the CFLP. We used instances generated as in [9] (as in [1]), that are as follows. Once again we generate the demand and facility points uniformly at random in $[0,1]\times[0,1]$. The unit transportation costs correspond to the Euclidean distance scaled by 10. The demands are generated from U[5,35], i.e., the uniform distribution in the interval [5,35). The capacities s_i are generated from U[10,160] and for the fixed costs we used the formula $f_i = U[0,90] + U[100,110]\sqrt{s_i}$ to take into account the economies of scale. Then we rescaled the capacities so that $\sum_i s_i / \sum_j d_j$ takes the value of a parameter *factor*. The parameter then is set to $factor = 1.5, 2, 3, 5$ or 10. To make the instances more challenging, we further adjusted the fixed cost by another parameter, v. Then v is 2 for the first two values of *factor* and 1 for the rest. Thus we considered 5 type of instances, exactly as in [9]. The largest instance they considered was $n \times m = 50 \times 50$. In [1], the largest instance is 100×75. In contrast we dealt with instances of size 300×300, 500×500, 800×800, 1000×1000.

A sample of our computational results is in Table 5. Each of the last three columns represents the average on runs of the algorithm in 5 different instances of the same type.

As pointed out in the introduction, the linear programming relaxation P can sometimes fail to provide enough information to tackle the difficulties of the problem. In our experiments, out of 100 trials, we only failed to find feasible solutions within our iteration bounds in 3 of them (with *factor*=1.5, two instances with $n = 500$ and one with $n = 800$).

6 Final Remarks

In this paper we focused on two of the simplest facility location models: the capacitated and uncapacitated facility location problems. We developed a heuristic to approximately solve the problems, providing a feasible solution together with a lower bound on the optimum. Our methods are based on the volume algorithm to solve a linear programming relaxation to the problem, together with variants of randomized rounding to obtain feasible solutions. We point out here that these ideas can be extended easily to solve other location problems with more or different complicating constraints, such as the k-median problem (see [5]).

number of clients n	factor	relative error	iterations	time
300	1.5	0.87%	1908	497s
	2	0.68%	864	240s
	3	0.74%	720	241s
	5	0.85%	906	274s
	10	0.93%	1158	310s
500	1.5	0.13%	3200	2306s
	2	0.65%	954	758s
	3	0.67%	786	730s
	5	0.86%	792	752s
	10	1.25%	1886	1694s
800	1.5	0.65%	2756	5541s
	2	0.50%	1098	2191s
	3	0.64%	816	1709s
	5	0.68%	758	1891s
	10	1.02%	1398	3462s
1000	1.5	1.07%	2748	9311s
	2	0.73%	1060	3291s
	3	0.46%	834	2605s
	5	0.67%	866	3354s
	10	0.93%	938	3495s

Table 5: Computational results for capacitated facility location instances .

References

[1] K. Aardal. Capacitated facility location: separation algorithms and computational experience. *Mathematical Programming*, 81:149–175, 1998.

[2] S. Ahn, C. Cooper, G. Cornuéjols, and A.M. Frieze. Probabilistic analysis of a relaxation for the k-median problem. *Mathematics of Operations Research*, 13:1–31, 1988.

[3] M.L. Balinski. Integer programming: methods, uses, computation. *Management Science*, 12(3):253–313, 1965.

[4] F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87:385–399, 2000.

[5] F.A. Chudak. *Improved approximation algorithms for the uncapacitated facility location problem.* PhD thesis, Cornell University, 1998.

[6] F.A. Chudak. Improved approximation algorithms for uncapacitated facility location. In *Proceedings of the 6th IPCO Conference*, pages 180–194, 1998.

[7] F.A. Chudak and D.B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. In preparation, 1999.

[8] G. Cornuéjols, G.L. Nemhauser, and L.A. Wolsey. The uncapacitated facility location problem. In P. Mirchandani and R. Francis, editors, *Discrete Location Theory*, pages 119–171. John Wiley and Sons, Inc., New York, 1990.

[9] G. Cornuéjols, R. Sridharan, and J.M. Thizy. A comparison of heuristics and relaxations for the capacitated plant location problem. *European Journal of Operational Resesarch*, 50:280–297, 1991.

[10] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26:992–1009, 1978.

[11] D. Erlenkotter, 1991. Program DUALOC – Version II. Distributed on request.

[12] A.V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. Technical Report STAN-CS-92-1439, Stanford University, 1992.

[13] M. Held, P. Wolfe, and H.P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 49:62–88, 1991.

[14] C. Lemaréchal. Nondifferential optimization. In G.L. Nemhauser, A.H.G. Rinnoy Kan, and M.J. Todd, editors, *Optimization*, Handbooks in Operations Research, pages 529–572. North Holland, 1989.

[15] P. Mirchandani and R. Francis, eds. *Discrete Location Theory.* John Wiley and Sons, Inc., New York, 1990.

[16] P. Raghavan and C.D. Thompson. Randomized rounding. *Combinatorica*, 7:365–374, 1987.

[17] D.B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[18] P. Wolfe. A method of conjugate subgradients for minimizing nondifferentiable functions. *Mathematical Programming Study*, 3:145–173, 1975.