Improved approximation algorithms for capacitated facility location problems*

Fabián A. Chudak† David P. Williamson‡

Abstract

In a surprising result, Korupolu, Plaxton, and Rajaraman [13] showed that a simple local search heuristic for the capacitated facility location problem (CFLP) in which the service costs obey the triangle inequality produces a solution in polynomial time which is within a factor of $8+\epsilon$ of the value of an optimal solution. By simplifying their analysis, we are able to show that the same heuristic produces a solution which is within a factor of $6(1+\epsilon)$ of the value of an optimal solution. Our simplified analysis uses the supermodularity of the cost function of the problem and the integrality of the transshipment polyhedron.

Additionally, we consider the variant of the CFLP in which one may open multiple copies of any facility. Using ideas from the analysis of the local search heuristic, we show how to turn any α-approximation algorithm for this variant into a polynomial-time algorithm which, at an additional cost of twice the optimum of the standard CFLP, opens at most one additional copy of any facility. This allows us to transform a recent 2-approximation algorithm of Mahdian, Ye, and Zhang [17] that opens many additional copies of facilities into a polynomial-time algorithm which only opens one additional copy and has cost no more than four times the value of the standard CFLP.

1 Introduction

We consider the *capacitated facility location problem* (CFLP). In this problem, we are given a set of facilities F and a set of clients D. Each client $j \in D$ has a demand d_j that must be serviced by one or more open facilities. There is a cost f_i for opening facility $i \in F$, and it costs c_{ij} for facility i to service one unit of demand from client j. We call the first type of cost *facility cost* and the second *service cost*. Furthermore, no facility may service more than U units of demand. We wish to service all clients at minimum total cost. The capacitated facility location problem and variations of it have been well-studied in the literature (see, for example, the book of Mirchandani and Francis [18]) and arise in practice (see, for example, the paper of Barahona and Jensen [4] for an instance of a parts warehousing problem from IBM).

The CFLP is NP-hard even in the case that $U = \infty$, sometimes called the *uncapacitated facility location problem* (UFLP) [10]. Thus we turn our attention to *approximation algorithms*. We say we have an α-approximation algorithm for the CFLP if the algorithm runs in polynomial time and

*A preliminary version of this paper appeared in the *Proceedings of the 7th Conference on Integer Programming and Combinatorial Optimization* [9].

†Address: ETH Zurich, Institut für Operations Research, CLP D 7, Clausiusstrasse 45, 8092 Zürich, Switzerland. Email: fabian.chudak@ifor.math.ethz.ch. This research was performed while the author was a postdoctoral fellow at the IBM T.J. Watson Research Center.

‡Address: IBM Almaden Research Center, 650 Harry Rd. K53/B1, San Jose, CA, 95120, USA. Email: dpw@almaden.ibm.com. Web: www.almaden.ibm.com/cs/people/dpw.

returns a solution of value no more than α times the value of an optimal solution. The value α is sometimes called the *performance guarantee* of the algorithm.

It is possible to express any instance of the well-known set cover problem as an instance of the UFLP of the same cost, which implies that unless $P = NP$, there is no approximation algorithm for the UFLP with performance guarantee better than $c \ln |D|$, where c is some constant [15, 11, 21, 2]. Thus we turn to special cases of the CFLP. In particular, we assume that for any $k, l \in F \cup D$ a service cost c_{kl} is defined, and the service costs are symmetric and obey the triangle inequality. This is a natural assumption, since service costs are often associated with the distance between points in Euclidean space representing facilities and clients. From now on, when we refer to the CFLP or UFLP, we refer to this metric case.

Korupolu, Plaxton, and Rajaraman (KPR) gave the first approximation algorithm for the CFLP with constant performance guarantee [13]. Surprisingly, KPR show that a simple local search heuristic is guaranteed to run in polynomial time and to terminate with a solution of value no more than $(8 + \epsilon)$ times optimum, for any $\epsilon > 0$. The central contribution of our paper is to simplify and improve their analysis of the heuristic, showing that it is a $6(1 + \epsilon)$-approximation algorithm for the CFLP. Although our proof follows theirs closely at many points, we show that some case distinctions (e.g. "cheap" versus "expensive" facilities) are unnecessary and some proofs can be simplified and strengthened by using standard tools from mathematical programming. For example, using the supermodularity of the cost function of the CFLP reduces a five page proof to a half page, and using the notion of a transshipment problem and the integrality of its polyhedron allows us to get rid of the extraneous concept of a "refined β-allocation," which in turn leads to the improved performance guarantee.

We are also able to use a concept translated from KPR to get an improved approximation algorithm for a variant of the CFLP. The variant we consider is the one in which a solution may open up to k copies of facility i, each at cost f_i and having capacity U, and we denote this problem the k-CFLP (so that the ordinary CFLP is the same as the 1-CFLP). Shmoys, Tardos, and Aardal [22] give a polynomial-time algorithm for the $\frac{7}{2}$-CFLP which produces a solution of value no more than 7 times the optimal value of the 1-CFLP. Chudak and Shmoys [8], building on previous work [6, 7] for the UFLP, give a 3-approximation algorithm for the ∞-CFLP. Mahdian, Ye, and Zhang [16] give a 2.89-approximation algorithm, and the same authors [17] recently developed a 2-approximation algorithm. Here we show how to take any solution for the ∞-CFLP and produce from it a solution for the 2-CFLP adding cost no more than twice the optimal value of the 1-CFLP. Thus by using the algorithm of Mahdian et al., we are able to produce solutions in polynomial time for the 2-CFLP of cost no more than 4 times the optimal value of the 1-CFLP, improving the previous result of Shmoys et al. [22].

The recent work on approximation algorithms for facility location problems was started by the paper of Shmoys, Tardos, and Aardal [22], who gave a 3.16-approximation algorithm for the UFLP, the first approximation algorithm for this problem with a constant performance guarantee. There has been a substantial amount of research since then on the UFLP; the best currently known approximation algorithm for the problem has a performance guarantee of 1.52 [16]. An observation of Sviridenko [23], combined with a result of Guha and Khuller [12] implies that no approximation algorithm for the UFLP with performance guarantee 1.46 is possible, unless $P = NP$.

Following the appearance of an extended abstract of this paper [9], Charikar and Guha [5] have shown how to modify our algorithm slightly to obtain a performance guarantee of $3 + 2\sqrt{2} + \epsilon \approx 5.83 + \epsilon$. Pál, Tardos, and Wexler [20] give a 9-approximation algorithm for the case when the capacity of each facility is allowed to vary from facility to facility (that is, the capacity of facility i is U_i).

The rest of the paper is structured as follows. We begin in Section 2, where we introduce the

local search algorithm of KPR, define some notation, and prove some preliminary lemmas. We then, in Section 3, define the concept of a "swap graph", analogous to the concept of the β-allocation problem in KPR, and show how it leads to our algorithm for the 2-CFLP. Finally, we show how to obtain an improved analysis of the local search algorithm using the swap graph in Section 4.

2 The local search algorithm

2.1 Preliminaries

In this section, we define some notation and give some preliminary lemmas that will be needed in subsequent discussion. Given a set $S \subseteq F$ of facilities to open, it is easy to determine the minimum service costs for that set of facilities by solving the following transportation problem: for each facility $i \in F$ we have a supply node i with supply U, and for each client $j \in D$ we have a demand node j with demand d_j; the unit shipping cost from i to j is c_{ij}. When discussing the k-CFLP, we will let S be a multiset of facilities (here l copies of $i \in F$ in S corresponds to opening l facilities at location i). We let $x(S, i, j)$ denote the amount of demand of client j serviced by facility i in the solution given by S. We will denote the overall cost of the location problem given by opening the facilities in S by $c(S)$. Furthermore, we let $c_f(S)$ denote the facility costs of the solution S (i.e., $c_f(S) = \sum_{i \in S} f_i$) and $c_s(S)$ denote the service costs of the solution S (i.e., $c_s(S) = \sum_{i \in S, j \in D} c_{ij} x(S, i, j)$). Let S^* denote the set of facilities opened by some optimal solution; it will always be a solution to the 1-CFLP and hence not a multiset. Let $n = |F|$.

The local search algorithm given by KPR for the CFLP is the following: given a current solution S, perform any one of three types of operations that improve the value of the solution by at least $c(S)/p(n, \epsilon)$, where $p(n, \epsilon)$ is a suitably chosen polynomial in n and $1/\epsilon$, and continue doing so until none of these operations results in an improvement of at least that much (We will show later that $p(n, \epsilon) = 8n/\epsilon$ is a suitable choice). The operations are: *adding* a facility $i \in F - S$ to S (i.e., $S \leftarrow S + i$); *dropping* a facility $i \in S$ (i.e., $S \leftarrow S - i$); or *swapping* a facility $i \in S$ for a facility $i' \in F - S$ (i.e., $S \leftarrow S - i + i'$). We call any operation that improves the solution by at least $c(S)/p(n, \epsilon)$ an *admissible* operation; thus the algorithm runs until there are no more admissible operations. This heuristic runs in polynomial time, as KPR argued: start with some arbitrary feasible solution (for instance, setting $S = F$). Since in each step, the value of the solution improves by a factor of $(1 - \frac{1}{p(n,\epsilon)})$, after $p(n, \epsilon)$ operations the value of the solution will have improved by a constant factor. Since the value of the solution can't be smaller than $c(S^*)$, after $O(p(n, \epsilon) \log \frac{c(F)}{c(S^*)})$ operations the algorithm will terminate. Each local search step can be implemented in polynomial time, and $O(p(n, \epsilon) \log c(F))$ is a polynomial in the input size, so overall the algorithm takes polynomial time.

We now turn to proving some preliminary lemmas. These lemmas use the fact that the cost function c is supermodular; that is, if $A, B \subseteq F$, we have that

$$c(A) + c(B) \leq c(A \cap B) + c(A \cup B).$$

(See Babayev [3], Propositions 3.3 and 3.4 of Nemhauser, Wolsey, and Fisher [19].) In particular, c_s is supermodular, while c_f is modular (that is, $c_f(A) + c_f(B) = c_f(A \cap B) + c_f(A \cup B)$). We will use the fact that supermodularity holds even for multisets.

We will show the following three lemmas:

Lemma 2.1 If $c(S) \geq (1 + \epsilon)c(S^*)$ and $S \subseteq S^*$, then there is an admissible add operation.

Lemma 2.2 If $c(S) \geq (1 + \epsilon)c(S^*)$ and $S \supseteq S^*$, then there is an admissible drop operation.

Lemma 2.3 If for all $u \in V - S$, $c(S+u) - c(S) > -\beta c(S)$, then $c_s(S) \leq c(S^*) + n\beta$.

From the last lemma we derive the following corollary, previously shown in KPR.

Corollary 2.4 [KPR [13], Lemma 9.3] If there is no admissible add operation, then $c_s(S) \leq c(S^*) + \frac{n \cdot c(S)}{p(n,\epsilon)}$.

Proof. Use $\beta = 1/p(n,\epsilon)$. ■

In addition, in Section 4, we show the following theorem.

Theorem 2.5 If neither $S \subseteq S^*$ nor $S \supseteq S^*$ and there are no admissible drops or swaps, then

$$c_f(S - S^*) \leq 3c_f(S^* - S) + 2c_s(S) + 2c_s(S^*) + n \cdot c(S)/p(n,\epsilon).$$

2.2 The main theorem

Before proving the lemmas, we show how they lead to the $6(1+\epsilon)$-approximation algorithm for the CFLP.

Theorem 2.6 If there are no admissible operations, then

$$c(S) \leq 6(1+\epsilon)c(S^*).$$

Proof. If there are no admissible operations and if $S \subseteq S^*$ or $S \supseteq S^*$, then by Lemmas 2.1 and 2.2 we know that $c(S) \leq (1+\epsilon)c(S^*)$. If there are no admissible operations and neither $S \subseteq S^*$ nor $S \supseteq S^*$ then

$$c_f(S - S^*) \leq 3c_f(S^* - S) + 2c_s(S) + 2c_s(S^*) + n \cdot c(S)/p(n,\epsilon),$$

by Theorem 2.5. Adding $c_f(S \cap S^*) + c_s(S)$ to both sides, we obtain

$$\begin{aligned} c(S) &\leq 2c_f(S^* - S) + c_f(S^*) + 3c_s(S) + 2c_s(S^*) + n \cdot c(S)/p(n,\epsilon) \\ &\leq 3c(S^*) + 3c(S^*) + 4n \cdot c(S)/p(n,\epsilon), \end{aligned}$$

using Corollary 2.4. Then

$$c(S)\left(1 - \frac{4n}{p(n,\epsilon)}\right) \leq 6c(S^*),$$

or

$$c(S) \leq \frac{6}{1 - \frac{4n}{p(n,\epsilon)}} c(S^*).$$

This gives that $c(S) \leq 6(1+\epsilon)c(S^*)$ for $p(n,\epsilon) \geq \frac{8n}{\epsilon}$ and $\epsilon < 1$. ■

2.3 Proofs of preliminary lemmas

We start by proving somewhat more general forms of Lemmas 2.1 and 2.2, and deriving those Lemmas as corollaries.

Lemma 2.7 Let $f : V \to \Re$ be any supermodular function. If $S \subset S^* \subseteq V$, then there exists $u \in S^* - S$ such that

$$f(S+u) - f(S) \leq \frac{1}{|S^* - S|} \left(f(S^*) - f(S) \right).$$

Proof. Let $W = S^* - S = \{u_1, u_2, \ldots, u_k\}$. Let $W_i = \{u_1, \ldots, u_i\}$. The statement certainly holds if $|S^* - S| = 1$, so assume that $|S^* - S| \geq 2$. Then by the supermodularity of f we know that:

$$\begin{aligned} f(S + W_{k-1}) + f(S + u_k) &\leq f(S^*) + f(S) \\ f(S + W_{k-2}) + f(S + u_{k-1}) &\leq f(S + W_{k-1}) + f(S) \\ &\vdots \\ f(S + W_2) + f(S + u_3) &\leq f(S + W_3) + f(S) \\ f(S + u_1) + f(S + u_2) &\leq f(S + W_2) + f(S). \end{aligned}$$

Summing the inequalities and subtracting $\sum_{i=2}^{k-1} f(S + W_i)$ from both sides, we obtain

$$\sum_{i=1}^{k} f(S + u_i) \leq f(S^*) + (k-1) f(S),$$

so that there exists some i such that

$$f(S + u_i) - f(S) \leq \frac{1}{k} (f(S^*) - f(S)).$$

■

Proof of Lemma 2.1: It follows from Lemma 2.7 that if $c(S) \geq (1+\epsilon) c(S^*)$, then there exists an add operation that changes the cost by no more than $\frac{1}{n}(c(S^*) - c(S)) \leq \frac{1}{n}\left(\frac{1}{1+\epsilon} - 1\right) c(S) \leq -c(S)/p(n,\epsilon)$, for $p(n,\epsilon) \geq \frac{n(1+\epsilon)}{\epsilon}$. For our choice of $p(n,\epsilon) = \frac{8n}{\epsilon}$ this statement holds (assuming $\epsilon < 1$). So there is an admissible add operation. ■

Proof of Lemma 2.3: We prove the contrapositive. Suppose it is the case that $c_s(S) > c(S^*) + n\beta$. Then by adding $c_f(S)$ to the left-hand side of this inequality and $c_f(S - S^*)$ to the right-hand side, we have that

$$c(S) > c_s(S^*) + c_f(S \cup S^*) + n\beta.$$

Observing that $c_s(S \cup S^*) \leq c_s(S^*)$, we obtain

$$c(S) > c(S^* \cup S) + n\beta.$$

Applying Lemma 2.7 to the sets $S \subseteq S^* \cup S$ gives us that there is an add operation that changes the cost by no more than $\frac{1}{n}(c(S^* \cup S) - c(S)) \leq -\beta c(S)$. ■

Lemma 2.8 Let $f : V \to \Re$ be any supermodular function. If $S \supset S^*$ then there exists $u \in S - S^*$ such that

$$f(S-u) - f(S) \leq \frac{1}{|S - S^*|} \left(f(S^*) - f(S)\right).$$

Proof. The following proof was given by an anonymous referee. Define $g(S) = f(V - S)$. Then by assumption, g is supermodular and $V - S \subseteq V - S^* \subseteq V$ satisfy the assumptions of Lemma 2.7. By Lemma 2.7, there exists $u \in (V - S^*) - (V - S) = S - S^*$ such that

$$g(V - S + u) - g(V - S) \leq \frac{1}{|(V - S^*) - (V - S)|} \left(g(V - S^*) - g(V - S)\right).$$

This implies the lemma statement. ■

Proof of Lemma 2.2: It follows from Lemma 2.8 that if $c(S) \geq (1+\epsilon)c(S^*)$, then there exists a drop operation that changes the cost by no more than $\frac{1}{n}(c(S^*) - c(S)) \leq \frac{1}{n}\left(\frac{1}{1+\epsilon} - 1\right)c(S) \leq -c(S)/p(n,\epsilon)$, for $p(n,\epsilon) \geq \frac{n(1+\epsilon)}{\epsilon}$, which holds for our choice of $p(n,\epsilon) = \frac{8n}{\epsilon}$ (assuming $\epsilon < 1$). So there is an admissible drop operation. ■

3 Path decompositions and the swap graph

3.1 A path decomposition

In this section, we define a path decomposition and a concept called the swap graph which will be useful in both of our results. The path decomposition is more or less equivalent to the "difference graph" of KPR [13], while the swap graph roughly corresponds to their "β-allocation problem". The path decomposition is useful in comparing the value of our current solution with the optimal solution. The swap graph will be used in the analysis of the local search algorithm (in the proof of Theorem 2.5) and will be used in the algorithm and analysis of our result for the 2-CFLP.

To obtain the path decomposition, we start with some current solution S and the optimal solution S^*. We construct the following directed graph: we include a node j for each client $j \in D$, and a node i for each facility $i \in S \cup S^*$. We include an arc (j,i) of weight $w(j,i) = x(S^*,i,j)$ for all $i \in S^*, j \in D$ when $x(S^*,i,j) > 0$, and an arc (i,j) of weight $w(i,j) = x(S,i,j)$ for all $i \in S, j \in D$ when $x(S,i,j) > 0$. See Figure 1 for an example. Observe that by the properties of x, the total weight of all arcs incoming to a node j for $j \in D$ is d_j, as is the total weight of all outgoing arcs. The total weight of arcs incoming to any node i for $i \in S^*$ is at most U, and the total weight of arcs going out of any node i for $i \in S$ is also at most U. Furthermore, notice that $\sum c_{ij} w(i,j) = c_s(S^*) + c_s(S)$.

By standard path-stripping arguments (see, for example, Section 3.5 of Ahuja, Magnanti, and Orlin [1]), we can decompose this graph into a set of weighted paths and cycles. The paths start at nodes in S and end at nodes in S^*. The cycles must be on nodes in $S \cap S^*$; we will ignore them, since they play no role in our results. We now introduce some notation that we will use to discuss the paths.

Notation 3.1

- Let $\mathcal{P}$ denote the set of weighted paths from nodes in S to nodes in S^* obtained by applying path-stripping to the graph defined above.

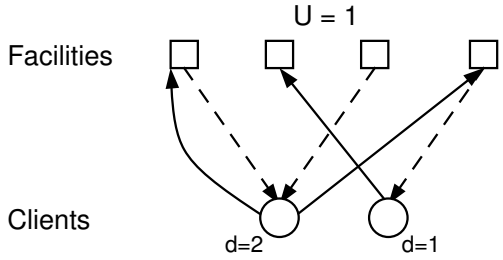

Figure 1: A sample of the path decomposition. The dashed lines are the assignment of the current solution S, while the solid lines are the assignment of the optimal solution S^*. Each line represents one unit of demand.

- Let $w(P)$ denote the weight of path $P \in \mathcal{P}$.
- Let $c(P)$ denote the cost of path $P \in \mathcal{P}$; that is, $c(P) = \sum_{(i,j)\in P} c_{ij}$.
- For any subset of paths $\mathcal{P}' \subseteq \mathcal{P}$:
  - let $\mathcal{P}'(A, \cdot)$ denote the set of paths in $\mathcal{P}'$ starting at nodes $i \in A$ for $A \subseteq S$;
  - let $\mathcal{P}'(\cdot, B)$ denote the set of paths in $\mathcal{P}'$ ending at nodes $i' \in B$ for $B \subseteq S^*$;
  - let $\mathcal{P}'(A, B)$ denotes the set of paths in $\mathcal{P}'$ from $i \in A \subseteq S$ to $i' \in B \subseteq S^*$;
  - let $w(\mathcal{P}') = \sum_{P\in\mathcal{P}'} w(P)$;
  - let $val(\mathcal{P}') = \sum_{P\in\mathcal{P}'} w(P)c(P)$.

Observe that $\sum_{P\in\mathcal{P}} c(P)w(P) \leq c_s(S) + c_s(S^*)$. Thus $val(\mathcal{P}) \leq c_s(S) + c_s(S^*)$.

3.2 The swap graph

The swap graph simply corresponds to a transshipment problem from a specified subset S' of nodes of a current solution S (possibly a multiset) to a subset of facilities $F' \subset F$. We place demands of 1 on the nodes of S' and integer supplies on the facilities F', and set the cost of an edge $\hat{c}_{kl}$ from $k \in S'$ to $l \in F - S$ to be $Uc_{kl} + f_l - f_k$. When using a swap graph, we use the path decomposition to prove that a fractional solution of some value β exists to the transshipment problem. Then by the integrality of the transshipment polyhedra, we know that there exists an integral solution to the transshipment problem of cost no more than β such that one unit of flow is shipped from each node in S' to exactly one node in F'.

We then observe that in the integral solution to the transshipment problem, each unit of flow from $k \in S'$ to $l \in F'$ corresponds to a swap operation in our current solution that can be performed while increasing the cost of the current solution by no more than $\hat{c}_{kl}$: each unit of demand assigned from client j to $k \in S'$ in the current solution can be assigned to $l \in F'$ at a change in cost of

$$c_{lj} - c_{kj} \leq c_{lk} + c_{kj} - c_{kj} \leq c_{kl}.$$

There are at most U units of demand assigned to $k \in S'$, so the total change in cost of transferring the demand assigned to k to l is at most Uc_{kl}, and the change in cost of closing facility k and opening facility l is $f_l - f_k$. Thus the overall cost of performing the swap is at most $Uc_{kl} + f_l - f_k = \hat{c}_{kl}$.

3.3 An algorithm for the 2-CFLP

To illustrate the use of the swap graph, we give an algorithm such that given any solution to the k-CFLP, for $k > 2$, the algorithm returns a solution to the 2-CFLP at additional cost no more than twice the cost of an optimal solution to the 1-CFLP. Given a solution S to the k-CFLP (a multiset), the algorithm works as follows. As long as there exists an add operation that reduces the cost, we add facilities from $F - S$ to S one at a time. Let $\tilde{S}$ be the solution when there are no longer add operations that improve the cost of the solution. We then solve a transshipment problem via the swap graph between nodes in $S' \subseteq \tilde{S}$ and F, where S' consists of all the copies of facilities that are used at full capacity; let $\tilde{S}_1 = \tilde{S} - S'$ be the remainder copies of facilities in S. We put demands of 1 on the nodes in S', and supplies of 1 on the nodes in F, so that we have the following problem:

$$\text{Min} \quad \sum_{k \in S', l \in F} \hat{c}_{kl} x_{kl}$$

subject to:

$$\sum_{l \in F} x_{kl} = 1 \qquad \forall k \in S'$$

$$\sum_{k \in S'} x_{kl} \leq 1 \qquad \forall l \in F$$

$$x_{kl} \geq 0 \qquad \forall k \in S', l \in F.$$

Given the integral solution x_{kl} to the transshipment problem, whenever $x_{kl} = 1$, we obtain a new solution $\hat{S}$ by swapping $k \in S'$ for $l \in F$ at change in cost $\hat{c}_{kl}$. It is easy to verify that in solution $\hat{S}$ we open at most 2 facilities for each $i \in F$, one possibly from the assignment problem, and one from $\tilde{S}_1$, so that we have a solution to the 2-CFLP.

Certainly the algorithm runs in polynomial time. We can now prove that this algorithm does not increase the cost of the original solution by much.

Theorem 3.2 Suppose there exists a feasible solution to the 1-CFLP. The algorithm above, given a solution S to the k-CFLP, produces a solution $\hat{S}$ to the 2-CFLP at additional cost at most twice the optimal value of a solution to the 1-CFLP.

Proof. We start with the solution S and apply add operations, each of which does not increase the cost of the solution. Given the solution $\tilde{S}$ (after we have applied all add operations to S that improve the cost of the solution), let $\mathcal{P}$ be the path decomposition giving paths from facilities in $\tilde{S}$ to an optimal 1-CFLP solution S^*. We use the path decomposition to give a fractional solution to the transshipment problem of cost no more than

$$c_s(\tilde{S}) + c_s(S^*) - c_f(S') + c_f(S^*) \leq c_s(\tilde{S}) + c(S^*).$$

By applying Lemma 2.3 with $\beta = 0$, we know that we must have $c_s(\tilde{S}) \leq c(S^*)$. Since the cost of the solution $\hat{S}$ obtained after swapping is at most the cost of $c(\tilde{S})$ plus the cost of the solution to the transshipment problem, we know that

$$c(\hat{S}) \leq c(\tilde{S}) + 2c(S^*).$$

To obtain a feasible fractional solution $\widetilde{x}$ to the transshipment problem, we set $\widetilde{x}_{kl}$ to be $1/U$ times the total weight of paths from $k \in S'$ to $l \in S^*$ (that is, $\widetilde{x}_{kl} = w(\mathcal{P}(k,l))/U$). Clearly $\widetilde{x}$ is a

feasible solution for the transshipment problem, since the total weight of paths leaving any $k \in S'$ is U, and the total weight of paths entering any $l \in S^*$ is at most U. The cost of the solution $\widetilde{x}$ is

$$\begin{aligned}
\sum_{k \in S', l \in S^*} \hat{c}_{kl} \widetilde{x}_{kl} &= \sum_{k \in S', l \in S^*} (U c_{kl} - f_k + f_l)(1/U) \sum_{P \in \mathcal{P}(k,l)} w(P) \\
&\leq \sum_{k \in S', l \in S^*} \sum_{P \in \mathcal{P}(k,l)} w(P) c_{kl} - c_f(S') + c_f(S^*) \\
&\leq \sum_{P \in \mathcal{P}} c(P) w(P) - c_f(S') + c_f(S^*) \\
&\leq c_s(\tilde{S}) + c_s(S^*) - c_f(S') + c_f(S^*),
\end{aligned}$$

where the inequality $c_{kl} \leq c(P)$ follows from the triangle inequality. ■

Corollary 3.3 There is a polynomial-time algorithm that finds a solution to the 2-CFLP of cost at most 4 times the optimal value of a 1-CFLP solution.

Proof. We apply the 2-approximation algorithm of Mahdian, Ye, and Zhang [17] for the ∞-CFLP to obtain our initial solution S. Since the cost of the optimal solution for ∞-CFLP is at most the cost of the optimal solution for the 1-CFLP, the corollary follows. ■

4 Analysis of the local search algorithm

We now use the path decomposition and swap graph tools from the previous section to complete our analysis of the local search algorithm, and prove Theorem 2.5. The lemmas we derive below are roughly similar to those of KPR [13]: Lemma 4.5 corresponds to their Claim 9.8, and Lemma 4.6 to their Claims 9.9 and 9.10. However, we do not need an analogue of their "refined β-allocation", which gives us an improvement in the analysis in Lemma 4.5.

The basic strategy of the proof is as follows. We need to bound the cost of the facilities in $S - S^*$ in order to prove Theorem 2.5. To do this, we will demonstrate a set of swap and drop moves that could be performed on the current solution, such that: (1) each facility in $S - S^*$ occurs in exactly one of these moves; (2) the total cost of these moves is bounded by $3c_f(S^* - S) + 2c_s(S) + 2c_s(S^*) - c_f(S - S^*)$. By hypothesis, none of these moves is admissible, so we also know that each move costs at least $-c(S)/p(n, \epsilon)$, so that the total cost is at least $-|S - S^*|c(S)/p(n, \epsilon)$. Thus,

$$3c_f(S^* - S) + 2c_s(S) + 2c_s(S^*) - c_f(S - S^*) \geq -|S - S^*|c(S)/p(n, \epsilon),$$

and rearranging terms gives the desired inequality. We emphasize that the set of swap and drop moves we construct are solely for the purpose of analysis and constructing a bound on the cost of the facilities in $S - S^*$; the moves are not used in the algorithm. In particular, the set of moves could contain multiple swaps for the same facility in $S^* - S$; since this is for the purpose of the analysis, it is not problematic.

We begin by defining some notation we will need. Let S be a solution meeting the conditions of Theorem 2.5; namely, neither $S \subseteq S^*$ nor $S \supseteq S^*$ and there are no admissible drops or swaps. Let $\mathcal{P}$ be the path decomposition for S and an optimal solution S^* as defined in Notation 3.1. We will be particularly interested in three subsets of paths from $\mathcal{P}$, and we define them as follows.

Definition 4.1 Define $\mathcal{T} = \mathcal{P}(S - S^*, S \cap S^*)$; that is, the set of all paths from nodes in $S - S^*$ to $S \cap S^*$ (if any). We call these the *transfer paths*.

The basic idea of the transfer paths in the proof is that for any path $P \in \mathcal{T}$, we claim we can transfer $w(P)$ of the demand assigned to the start node of the path to the end node of the path at a cost of $c(P)w(P)$ without violating the capacity constraints. We establish this claim later.

Definition 4.2 Define $\mathcal{S} = \mathcal{P}(S - S^*, S^* - S)$; that is, the set of all paths from $S - S^*$ to $S^* - S$. We call these the *swap paths*.

We use the swap paths to get a fractional feasible solution for a transshipment problem from $S - S^*$ to $S^* - S$ in the swap graph, and get an integral solution of swap moves whose cost is a simple expression in terms of $c_s(S), c_s(S^*), c_f(S - S^*)$, and $c_f(S^* - S)$. Thus, as argued above, if no swap is admissible, this implies a bound on $c_f(S - S^*)$.

This idea does not quite work as stated because we might not get a good bound on the total cost of the swap/drop moves in this way. Thus, as in KPR [13], we split the nodes of $S - S^*$ into two types.

Definition 4.3 We define *heavy nodes* $H \subseteq S - S^*$ such that the weight of the swap paths from any $i \in H$ to $S^* - S$ is at least $U/2$ (i.e., $H = \{i \in S - S^* : w(\mathcal{S}(i, \cdot)) \geq U/2\}$). We define *light nodes* to be all other nodes in $S - S^*$: $L = S - S^* - H$.

We will be able to set up a transshipment problem for the nodes in H, which will give us a set of swap moves for H and thus a bound on $c_f(H)$. To get a bound on $c_f(L)$, we will have to set up a transshipment problem in a different manner and use the observation that we can transfer the demand assigned from one light node to another light node without violating capacity constraints.

To build towards our proof of Theorem 2.5, we now formalize the statements above in a series of lemmas.

Lemma 4.4 Given the current assignment, a new assignment can be created in which weight $w(\mathcal{T}(i, \cdot))$ of the demand assigned to facility $i \in S - S^*$ is transferred to nodes in $S \cap S^*$. The new assignment has cost at most that of the current assignment plus $val(\mathcal{T}(i, \cdot))$.

Proof. To prove the lemma, consider a path $P \in \mathcal{T}(i, \cdot)$, with start node i and end node i'. We observe that the first edge (i, j) in path P corresponds to a demand $w(P)$ assigned to i by client j in the current assignment. We reassign this demand to $i' \in S \cap S^*$; the increase in cost is at most $(c_{i',j} - c_{i,j})w(P) \leq c(P)w(P)$ by the triangle inequality. We now must show that such a reassignment does not violate the capacity constraints at i'. To see this, observe that by the properties of path-stripping, the total weight of paths in $\mathcal{T}$ ending at any node $i' \in S^* \cap S$ is the difference between the total weight of arcs coming into node i' and the total weight of arcs going out of node i'. Since the total weight of arcs coming into node i' corresponds to the total amount of demand assigned to i' by the optimal solution, and the total weight of arcs going out of node i' corresponds to the total amount of demand assigned to i' by the current solution, and the optimal solution must be feasible, we can increase the demand serviced by i' by this difference and still remain feasible. ■

Lemma 4.5 If there is not an admissible swap operation, then

$$c_f(H) \leq 2c_f(S^* - S) + 2val(\mathcal{S}(H, \cdot)) + val(\mathcal{T}(H, \cdot)) + |H|c(S)/p(n, \epsilon).$$

Proof. As suggested in the exposition, we set up a transshipment problem from H to $S^* - S$, as follows:

$$\text{Min} \quad \sum_{k \in H, l \in S^* - S} \hat{c}_{kl} x_{kl}$$

subject to:

$$\begin{aligned}
\sum_{l \in S^* - S} x_{kl} &= 1 && \forall k \in H \\
\sum_{k \in H} x_{kl} &\leq 2 && \forall l \in S^* - S \\
x_{kl} &\geq 0 && \forall k \in H, l \in S^* - S.
\end{aligned}$$

As stated in the exposition, we will give a fractional solution to this LP of relatively low cost and derive from it an integral solution of no greater cost corresponding to swaps for the nodes in H. The first set of constraints of the LP indicates that each node in H will be involved in exactly one swap. The second set of constraints indicates that each node in $S^* - S$ will be in no more than two swaps. The number of swaps in which nodes in $S^* - S$ are involved determines the factor in front of the term $c_f(S^* - S)$ in the lemma statement; we would like it to be as small as possible, and we show below that two is feasible.

We claim that we can give a fractional solution to the transshipment problem of cost no more than $2val(\mathcal{S}(H, \cdot)) + 2c_f(S^* - S) - c_f(H)$. Thus there exists an integral solution of no greater cost. Given an integral solution x, when $x_{kl} = 1$, we can swap facility $k \in H$ for $l \in S^* - S$ and transfer the demand $w(\mathcal{S}(k, \cdot))$ assigned to k at change in cost at most $\hat{c}_{kl}$. By Lemma 4.4, we can transfer the remaining demand $w(\mathcal{T}(k, \cdot))$ assigned to k to nodes in $S \cap S^*$ at change in cost at most $val(\mathcal{T}(k, \cdot))$. By the hypothesis of the lemma, we know that any swap for a facility results in a change in cost of at least $-c(S)/p(n, \epsilon)$. Summing over all swaps for $k \in H$ given by the solution to the transshipment problem, we have that

$$2val(\mathcal{S}(H, \cdot)) + 2c_f(S^* - S) - c_f(H) + val(\mathcal{T}(H, \cdot)) \geq -\frac{|H| c(S)}{p(n, \epsilon)}.$$

Rearranging terms gives us the lemma.

To complete the proof, we give a fractional solution $\widetilde{x}$ for this transshipment problem by setting

$$\widetilde{x}_{kl} = \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))}.$$

Certainly the constraints $\sum_{l \in S^* - S} \widetilde{x}_{kl} = 1$ are obeyed for all $k \in H$. The constraints $\sum_{k \in H} \widetilde{x}_{kl} \leq 2$ are also obeyed since

$$\sum_{k \in H} \widetilde{x}_{kl} = \sum_{k \in H} \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} \leq \sum_{k \in H} \frac{w(\mathcal{S}(k, l))}{U/2} \leq 2,$$

where the first inequality follows by the definition of H and the second since the total weight of paths adjacent to any node is at most U. The cost of this fractional solution is

$$\begin{aligned}
\sum_{k \in H, l \in S^* - S} \hat{c}_{kl} \widetilde{x}_{kl} &= \sum_{k \in H, l \in S^* - S} (U c_{kl} + f_l - f_k) \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} \\
&\leq \sum_{k \in H, l \in S^* - S} \left[(U c_{kl} + f_l) \frac{w(\mathcal{S}(k, l))}{U/2} - f_k \frac{w(\mathcal{S}(k, l))}{w(\mathcal{S}(k, \cdot))} \right] \\
&\leq \sum_{k \in H, l \in S^* - S} 2 c_{kl} w(\mathcal{S}(k, l)) + 2c_f(S^* - S) - c_f(H) \\
&\leq 2val(\mathcal{S}(H, \cdot)) + 2c_f(S^* - S) - c_f(H).
\end{aligned}$$

■

Lemma 4.6 [KPR [13], Claims 9.9 and 9.10] If there are no admissible drop and swap operations, then

$$c_f(L) \leq c_f(S^* - S) + 2val(\mathcal{T}(L,\cdot)) + 2val(\mathcal{S}(L,\cdot)) + |L|c(S)/p(n,\epsilon).$$

Proof. The proof of this lemma is similar to the proof of the previous lemma, although here we will have to set up a transshipment problem to capture both swap and drop operations. One difficulty with translating the previous proof to this case is ensuring that one can find a feasible fractional solution such that each facility in $S^* - S$ is in no more than a small constant number swap/drop operations. As in the previous lemma, the number of times each facility in $S^* - S$ is involved in a swap/drop operation implies the factor in front of the term $c_f(S^* - S)$ and thus we would like it to be as small as possible. We keep the number small by choosing exactly one "primary" facility k in L that can be swapped for a given facility l in $S^* - S$; i.e. $x_{kl} > 0$ for exactly one $k \in L$. We make a careful choice of this facility k so that any other facility i to which we might otherwise normally make a fractional assignment $x_{il} > 0$, we can drop i and reassign its demand to k, the primary facility of l, at not much more cost.

In order to set up the needed transshipment problem, we first need to define a cost function $\theta(k)$ on nodes $k \in L$. In words, the cost $\theta(k)$ is the cost per unit capacity for making $U/2$ units of capacity available at node k, either through the unused capacity at k (which incurs zero cost per unit), or through transferring demand via the transfer paths $\mathcal{T}(k,\cdot)$ (which incurs cost $val(\mathcal{T}(k,\cdot))/w(\mathcal{T}(k,\cdot))$ per unit).[1] Note that since $k \in L$, $w(\mathcal{S}(k,\cdot)) \leq U/2$, and thus the unused capacity at node k plus $w(\mathcal{T}(k,\cdot))$ is at least $U/2$. Let N_k denote the amount of unused capacity at k. If $N_k \geq U/2$ then $\theta(k) = 0$, otherwise

$$\theta(k) = \left(0 \cdot N_k + \frac{val(\mathcal{T}(k,\cdot))}{w(\mathcal{T}(k,\cdot))}(U/2 - N_k)\right)/(U/2).$$

Note that $\frac{U}{2}\theta(k) \leq val(\mathcal{T}(k,\cdot))$.

We can now define the needed transshipment problem from L to $(F - S) \cup L$ by setting cost $\hat{c}_{kl} = w(\mathcal{S}(k,\cdot))c_{kl} + f_l - f_k$ for $l \in F - S$, $\hat{c}_{kl} = w(\mathcal{S}(k,\cdot))(c_{kl} + \theta(l)) - f_k$ for $l \in L, l \neq k$, and $\hat{c}_{kk} = \infty$. The transshipment problem is then:

$$\text{Min} \sum_{k \in L, l \in (F-S)\cup L} \hat{c}_{kl} x_{kl}$$

subject to:

$$\sum_{l \in (F-S)\cup L} x_{kl} = 1 \qquad \forall k \in L$$

$$\sum_{k \in L} x_{kl} \leq 1 \qquad \forall l \in F - S$$

$$x_{kl} \geq 0 \qquad \forall k \in L, l \in (F - S) \cup L.$$

The first set of LP constraints ensures that each $k \in L$ is in exactly one drop or swap operation; the second set of constraints ensures that the facilities in $F - S$ are in no more than one drop/swap operation. We claim that we can give a fractional solution to the transshipment problem of cost no greater than $2val(\mathcal{S}(L,\cdot)) + val(\mathcal{T}(L,\cdot)) - c_f(L) + c_f(S^* - S)$. Thus there exists an integral

[1] The same cost function, including the definition of θ, was used by KPR [13].

solution of no greater cost. Given an integral solution x, when $x_{kl} = 1$ for $k \in L, l \in F - S$, we can swap facility $k \in L$ for $l \in F - S$ and transfer the demand $w(\mathcal{S}(k,\cdot))$ assigned to k at change in cost at most $\hat{c}_{kl}$. By Lemma 4.4, we can transfer the remaining demand $w(\mathcal{T}(k,\cdot))$ assigned to k to nodes in $S \cap S^*$ at change in cost at most $val(\mathcal{T}(k,\cdot))$. When $x_{ki} = 1$ for $k \in L, i \in L, k \neq i$, we can drop facility k from S and transfer the demand $w(\mathcal{S}(k,\cdot))$ assigned to k to i at change in cost $\hat{c}_{ki} = w(\mathcal{S}(k,\cdot))(c_{ki} + \theta(i))$, as this cost covers transferring these units of demand to i and transferring the same amount of demand from i to nodes in $S \cap S^*$. By Lemma 4.4, we can transfer the remaining demand $w(\mathcal{T}(k,\cdot))$ assigned to k to nodes in $S \cap S^*$ at change in cost at most $val(\mathcal{T}(k,\cdot))$. By the hypothesis of the lemma, we know that any swap or drop of a facility results in a change in cost of at least $-c(S)/p(n,\epsilon)$. Summing over all swaps and drops for $k \in L$ given by the solution to the transshipment problem, we have that

$$2val(\mathcal{S}(L,\cdot)) + 2val(\mathcal{T}(L,\cdot)) - c_f(L) + c_f(S^* - S) \geq -\frac{|L|c(S)}{p(n,\epsilon)}.$$

Rearranging terms gives us the lemma.

To complete the proof, we give a fractional solution $\widetilde{x}$ for this transshipment problem. For each $l \in S^* - S$ we find $k \in L$ that minimizes $c_{kl} + \theta(k)$ and designate k as the primary node $\pi(l)$ for l. We then set $\widetilde{x}_{kl}$ as follows. For each $l \in S^* - S$, if k is the primary node for l, we set $\widetilde{x}_{kl} = w(\mathcal{S}(k,l))/w(\mathcal{S}(k,\cdot))$, otherwise $\widetilde{x}_{kl} = 0$. For each $i \in L$, we set

$$\widetilde{x}_{ki} = \sum_{l \in S^* - S : i = \pi(l), k \neq \pi(l)} \frac{w(\mathcal{S}(k,l))}{w(\mathcal{S}(k,\cdot))}.$$

In other words, the fractional solution is constructed by rerouting the swap paths $\mathcal{S}(k,l)$ to l's primary node if k is not l's primary; $\widetilde{x}_{ki}$ gets the corresponding fraction of the paths $\mathcal{S}(k,\cdot)$ rerouted to i. This solution is feasible since certainly $\sum_{l \in F} x_{kl} = 1$ for all $k \in L$. Also, since for at most one $k \in L$ is $\widetilde{x}_{kl} > 0$ for $l \in F - S$, $\sum_{k \in L} x_{kl} \leq 1$. Observe that when $\widetilde{x}_{ki} > 0$ for $k \in L, i = \pi(l), l \in S^* - S$, then

$$\begin{aligned}
\hat{c}_{ki} &= w(\mathcal{S}(k,\cdot))(c_{ki} + \theta(i)) - f_k \\
&\leq w(\mathcal{S}(k,\cdot))(c_{kl} + c_{il} + \theta(i)) - f_k \\
&\leq w(\mathcal{S}(k,\cdot))(2c_{kl} + \theta(k)) - f_k, \qquad (1)
\end{aligned}$$

since $c_{il} + \theta(i) \leq c_{kl} + \theta(k)$ by the definition of primary nodes. Then the cost of this fractional solution is

$$\begin{aligned}
\sum_{k \in L, l \in F} \hat{c}_{kl}\widetilde{x}_{kl} &= \sum_{k \in L, l \in S^* - S, k = \pi(l)} \hat{c}_{kl} \frac{w(\mathcal{S}(k,l))}{w(\mathcal{S}(k,\cdot))} + \sum_{k \in L, i \in L} \hat{c}_{ki} \sum_{l \in S^* - S : i = \pi(l), k \neq \pi(l)} \frac{w(\mathcal{S}(k,l))}{w(\mathcal{S}(k,\cdot))} \qquad (2) \\
&\leq \sum_{k \in L, l \in S^* - S, k = \pi(l)} \hat{c}_{kl} \frac{w(\mathcal{S}(k,l))}{w(\mathcal{S}(k,\cdot))} + \sum_{k \in L, l \in S^* - S, i = \pi(l), k \neq i} \hat{c}_{ki} \frac{w(\mathcal{S}(k,l))}{w(\mathcal{S}(k,\cdot))} \qquad (3) \\
&\leq \sum_{k \in L, l \in S^* - S, k = \pi(l)} [w(\mathcal{S}(k,\cdot))c_{kl} + f_l - f_k] \frac{w(\mathcal{S}(k,l))}{w(\mathcal{S}(k,\cdot))} \\
&\quad + \sum_{k \in L, l \in S^* - S, k \neq \pi(l)} [w(\mathcal{S}(k,\cdot))(2c_{kl} + \theta(k)) - f_k] \frac{w(\mathcal{S}(k,l))}{w(\mathcal{S}(k,\cdot))} \qquad (4) \\
&\leq \sum_{k \in L, l \in S^* - S} [w(\mathcal{S}(k,\cdot))(2c_{kl} + \theta(k)) - f_k] \frac{w(\mathcal{S}(k,l))}{w(\mathcal{S}(k,\cdot))} + c_f(S^* - S) \qquad (5)
\end{aligned}$$

$$\leq \sum_{k\in L, l\in S^*-S} 2c_{kl}w(\mathcal{S}(k,l)) + \sum_{k\in L} val(\mathcal{T}(k,\cdot)) - c_f(L) + c_f(S^*-S) \quad (6)$$
$$\leq 2val(\mathcal{S}(L,\cdot)) + val(\mathcal{T}(L,\cdot)) - c_f(L) + c_f(S^*-S).$$

Equation (2) follows from separating the sum into two types of non-zero terms according to the definition of $\widetilde{x}_{kl}$. Inequality (3) follows from rewriting the double sum. Inequality (4) follows from expanding the definition of $\hat{c}_{kl}$ and from inequality (1). Inequality (5) follows by collecting common terms of the two sums in (4) and bounding the summation over f_l in the first sum by $c_f(S^*-S)$. Inequality (6) follows since the definition of L and θ implies that $\sum_{l\in S^*-S} w(\mathcal{S}(k,l))\theta(k) \leq \frac{U}{2}\theta(k) \leq val(\mathcal{T}(k,\cdot))$. ■

We can now complete the proof of Theorem 2.5.

Proof of Theorem 2.5: Lemma 4.5 implies that

$$c_f(H) \leq 2c_f(S^*-S) + 2val(\mathcal{S}(H,\cdot)) + val(\mathcal{T}(H,\cdot)) + |H|c(S)/p(n,\epsilon).$$

Lemma 4.6 implies that

$$c_f(L) \leq c_f(S^*-S) + 2val(\mathcal{T}(L,\cdot)) + 2val(\mathcal{S}(L,\cdot)) + |L|c(S)/p(n,\epsilon).$$

Summing the two together yields

$$\begin{aligned} c_f(S-S^*) &= c_f(H) + c_f(L) \\ &\leq 3c_f(S^*-S) + 2val(\mathcal{S}(S-S^*,\cdot)) + 2val(\mathcal{T}(S-S^*,\cdot)) + |S-S^*|c(S)/p(n,\epsilon) \\ &\leq 3c_f(S^*-S) + 2(c_s(S) + c_s(S^*)) + |S-S^*|c(S)/p(n,\epsilon), \end{aligned}$$

which gives Theorem 2.5. ■

5 Conclusion

Most approximation algorithms compare the solution obtained against a polynomial-time computable bound on the value of the optimal solution. This bound is sometimes implicit, but it usually not too hard to discover. One surprising facet of the KPR result is that it is not at all clear what the bound is. A typical bound is a linear programming relaxation of the problem. However, this bound can be quite weak for CFLP (see Section 3 of [22] for the relaxation and a bad example). Thus an interesting open question is to determine a stronger lower bound for the CFLP. The fact that a $6(1+\epsilon)$-approximation algorithm exists for the problem seems to imply that such a bound must exist.

Acknowledgments

We thank the two anonymous referees for many useful comments on the presentation of the paper.

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[2] S. Arora and M. Sudan. Improved low-degree testing and its applications. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 485–495, 1997.

[3] Dj. A. Babayev. Comments on the note of Frieze. *Mathematical Programming* 7:249–252, 1974.

[4] F. Barahona and D. Jensen. Plant location with minimum inventory. *Mathematical Programming* 83:101–111, 1998.

[5] M. Charikar and S. Guha. Improved combinatorial algorithms for facility location and k-median problems. In *Proceedings of the 40th IEEE Symposium on the Foundations of Computer Science*, pages 378–388, 1999.

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

[7] F. Chudak and D.B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. To appear in the *SIAM Journal on Computing*.

[8] F. Chudak and D.B. Shmoys. Improved approximation algorithms for a capacitated facility location problem. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 875–876, 1999.

[9] F. Chudak and D.P. Williamson. Improved approximation algorithms for capacitated facility location problems. In *Proceedings of the 7th IPCO Conference*, pages 99–113, 1999.

[10] G. Cornuéjols, G. Nemhauser, and L. 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.

[11] U. Feige. A threshold of $\ln n$ for approximating set-cover. *Journal of the ACM* 45:634–652, 1998.

[12] S. Guha and S. Khuller. Greedy strikes back: improved facility location algorithms. *Journal of Algorithms* 31:228–248, 1999.

[13] M. Korupolu, C. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms* 37:146–188, 2000.

[14] E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston, New York, 1976.

[15] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM* 41:960–981, 1994.

[16] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 229–242, 2002.

[17] M. Mahdian, Y. Ye, and J. Zhang. A 2-approximation algorithm for the soft-capacitated facility location problem. In *Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 129-140, 2003.

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

[19] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical Programming* 14:265–294, 1978.

[20] M. Pál, É. Tardos, and T. Wexler. Facility location with nonuniform hard capacities. In *Proceedings of the 41st IEEE Symposium on the Foundations of Computing*, pages 329–338, 2001.

[21] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 475–484, 1997.

[22] D. 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.

[23] M. Sviridenko, July, 1998. Personal communication.