Fault-Tolerant Facility Location

CHAITANYA SWAMY

University of Waterloo

AND

DAVID B. SHMOYS

Cornell University

Abstract. We consider a fault-tolerant generalization of the classical uncapacitated facility location problem, where each client \( j \) has a requirement that \( r_j \) distinct facilities serve it, instead of just one. We give a 2.076-approximation algorithm for this problem using LP rounding, which is currently the best-known performance guarantee. Our algorithm exploits primal and dual complementary slackness conditions and is based on clustered randomized rounding. A technical difficulty that we overcome is the presence of terms with negative coefficients in the dual objective function, which makes it difficult to bound the cost in terms of dual variables. For the case where all requirements are the same, we give a primal-dual 1.52-approximation algorithm.

We also consider a fault-tolerant version of the \( k \)-median problem. In the metric \( k \)-median problem, we are given \( n \) points in a metric space. We must select \( k \) of these to be centers, and then assign each input point \( j \) to the selected center that is closest to it. In the fault-tolerant version we want \( j \) to be assigned to \( r_j \) distinct centers. The goal is to select the \( k \) centers so as to minimize the sum of assignment costs. The primal-dual algorithm for fault-tolerant facility location with uniform requirements also yields a 4-approximation algorithm for the fault-tolerant \( k \)-median problem for this case. This the first constant-factor approximation algorithm for the uniform requirements case.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—Computations on discrete structures

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Approximation algorithms, facility location, \( k \)-median problem
1. Introduction

Facility location is a classical problem that has been widely studied in the field of operations research (see, e.g., the text of Mirchandani and Francis [1990]). In its simplest version, the uncapacitated facility location (UFL) problem, we are given a set of facilities $F$ and a set of clients $D$. Each facility $i$ has an opening cost $f_i$, and assigning client $j$ to facility $i$ incurs a cost equal to the distance $c_{ij}$ between $i$ and $j$. We want to open a subset of the facilities in $F$ and assign the clients to open facilities so as to minimize the sum of the facility opening costs and client assignment costs. We consider the case where the distances $c_{ij}$ form a metric, that is, they are symmetric and satisfy the triangle inequality.

In many settings it is essential to provide safeguards against failures by designing fault-tolerant solutions. For example, in a distributed network, we want to place caches and assign data requests to caches so as to be resistant against caches becoming unavailable due to node or link failures. A common solution is to replicate data items across caches and build some resilience in the network. This motivates the fault-tolerant facility location (FTFL) problem, wherein each client $j$ has a requirement $r_j$ and has to be assigned to $r_j$ distinct facilities instead of just one. Multiple facilities provide a backup against failures; if the facility closest to a client fails, the other facilities assigned to it could be used to serve it. To give a more concrete example, consider a setting where facilities (which could represent caches) fail independently with probability $p$, and each client $j$ must be guaranteed to be served by a (functional) facility with probability at least $q_j$. Then, this quality-of-service requirement translates precisely to the constraint that each client $j$ be assigned to $r_j = \lceil \log(1 - q_j)/\log p \rceil$ distinct facilities.

A more precise statement of the problem is as follows: We are given a set of facilities $F$ and a set of clients $D$, and each client $j$ has a requirement $r_j \geq 1$. Each facility $i$ has, as usual, an opening cost of $f_i$. In any feasible solution, we must assign every client $j$ to $r_j$ distinct open facilities. The assignment cost or service cost incurred for $j$ is the sum of the distances from $j$ to these $r_j$ facilities. The objective is to open a subset of the facilities, and assign each client $j$ to $r_j$ distinct open facilities so as to minimize the total facility opening and client assignment costs. This is a generalization of the uncapacitated facility location problem, which is the setting where $r_j = 1$ for each client $j \in D$.

Our main result is a 2.076-approximation algorithm for fault-tolerant facility location. This is currently the best-known guarantee, improving upon the guarantee of 2.408 due to Guha et al. [2003]. If all requirements are equal, we give a 1.52-approximation algorithm by building upon the algorithm of Jain et al. [2002] and Mahdian et al. [2006], which matches the current best guarantee for uncapacitated facility location (i.e., the unit requirement case). The previous best approximation guarantee for FTFL with uniform requirements was 1.861 [Mahdian et al. 2006]. We also consider the fault-tolerant version of the $k$-median problem where, in addition, a bound $k$ is specified on the number of facilities that may be opened. We consider
the case where all requirements are equal and give a 4-approximation algorithm for this case.

Related Work. The past several years have given rise to a variety of techniques for the design and analysis of approximation algorithms for the metric uncapacitated facility location problem. The first constant-factor approximation algorithm for this problem was due to Shmoys et al. [1997], who gave a 3.16-approximation algorithm using the filtering technique of Lin and Vitter [1992] to round the optimal solution of a linear program. After an improvement by Guha and Khuller [1999], Chudak and Shmoys [2003] gave an LP-rounding-based \((1 + \frac{3}{2})\)-approximation algorithm. They used information about the structure of optimal primal and dual solutions, and combined randomized rounding and the decomposition results of Shmoys et al. [1997] to get a variant that might be called clustered randomized rounding. Svirenko [2002] improved the ratio to 1.58. Jain and Vazirani [2001] gave a combinatorial primal-dual 3-approximation algorithm where the LP is used only in the analysis. Mettu and Plaxton [2003] gave a variant of this algorithm (which is not explicitly a primal-dual algorithm) that achieves the same approximation ratio, but runs in linear time. Local search algorithms were first analyzed by Korupolu et al. [2000] and later improved by Charikar and Guha [2005] and Arya et al. [2004]. Jain et al. [2003] gave a greedy algorithm and showed using a dual-fitting analysis that it has an approximation ratio of 1.61. This was improved by Mahdian et al. [2006] to 1.52, which is the best-known guarantee.

Charikar et al. [2002] gave the first constant-factor algorithm for the \(k\)-median problem based on LP rounding. This was improved in a series of papers [Jain and Vazirani 2001; Charikar and Guha 2005; Jain et al. 2003; Arya et al. 2004] to \((3 + \frac{1}{4})\) [Arya et al. 2004].

The fault-tolerant facility location (FTFL) problem was first studied by Jain and Vazirani [2000], who gave a primal-dual algorithm achieving a performance guarantee logarithmic in the largest requirement. Our algorithm is based on LP rounding. We consider the following LP and its dual.

\[
\begin{align*}
\text{(FTFL-P)} & \quad \min & \sum_i f_i y_i + c_{ij} x_{ij} & \quad \max & \sum_j r_j \alpha_j - z_i \\
\text{(FTFL-D)} & \quad \text{such that} & & \text{such that} & \\
& & x_{ij} \geq r_j & \alpha_j \leq \beta_{ij} + c_{ij} & \forall i, j \\
& & x_{ij} \leq y_i & \beta_{ij} \leq f_i + z_i & \forall i \\
& & y_i \leq 1 & \alpha_j, \beta_{ij}, z_i \geq 0 & \forall i, j \\
& & x_{ij}, y_i \geq 0 & & \forall i, j
\end{align*}
\]

Variable \(y_i\) indicates whether facility \(i\) is open, and \(x_{ij}\) indicates whether client \(j\) is assigned to facility \(i\). An integer solution to the LP corresponds exactly to a solution to our problem. Guha et al. [2003] round the aforesaid primal LP using filtering and the decomposition technique of Shmoys et al. [1997] to get a 3.16-approximation. They also show that a subsequent greedy local improvement postprocessing step reduces the approximation ratio to 2.408. They actually consider a more general version of FTFL where the service cost of a client \(j\) is a weighted sum of its distances to the \(r_j\) facilities to which it is assigned, where the weights are part of the
input. Unless otherwise stated, we use fault-tolerant facility location to denote the
unweighted (or unit-weighted) version of the problem.

In the case where all clients have the same requirement, namely, \( r_j = r \), better
results are known. Mahdian et al. [2001] showed that their 1.861-approximation
algorithm for UFL can be extended to give an algorithm for FTFL with a guarantee
of 1.861. Independent of our work, Jain et al. [2003] gave a 1.61-approximation
algorithm based on their 1.61-approximation algorithm for UFL.

Our Techniques. Our algorithm is also based on LP rounding, but does not use
filtering. Instead, it is based on the clustered randomized rounding technique of
Chudak and Shmoys [2003]. Our rounding algorithm exploits the optimality prop-
erties of the fractional solution by using complementary slackness conditions to
bound the cost of the solution, in terms of both the primal and dual optimal so-
lutions. One difficulty in using LP duality to prove an approximation ratio is the
presence of \( -j z_i \) in the dual objective function. As a result, bounding the cost
in terms of \( j r_j \alpha_j \) is not enough to prove an approximation guarantee. In gen-
eral, this is not an easy problem to tackle; for example, this problem also crops
up in designing approximation algorithms for the \( k \)-median problem, and conse-
sequently the only known LP rounding algorithm [Charikar et al. 2002] uses just
the optimal primal LP solution. For FTFL, however, complementary slackness al-

Our algorithm also clusters facilities around certain demand points, called cluster
centers, and opens at least one facility in each cluster. We do this clustering carefully
so as to ensure that each demand \( j \) has at least \( r_j \) open clusters “near” it; the facilities
opened from these clusters are used as backup facilities to serve demand \( j \). Each
facility \( i \) is opened with probability proportional to \( y_i \). The randomization step
allows us to reduce the service cost, since now for any demand \( j \) and any set \( S \) of
facilities that fractionally serve \( j \) such that the facility weight \( i \in S x_{ij} \) is “large”
(i.e., of at least some constant), there is a constant probability that a facility \( i \) from
\( S \) is opened.

Various difficulties arise in trying to extend the algorithm of Chudak and Shmoys
[2003] to the fault-tolerant setting. To ensure feasibility, we need to open different
facilities in different clusters. Also, we want a cluster to (ideally) have a fractional
facility weight of 1, so that the cost of opening a facility in this cluster can be
charged to the LP cost for opening a facility from this cluster. A small facility
weight could force us to incur huge cost (relative to the LP) in opening a facility
within the cluster, whereas if a cluster has a facility weight of more than 1 and
we open only one facility from the cluster, then we might end up opening less
facilities than necessary to satisfy the requirement of each client. However, unlike
UFL, once we require clusters to be disjoint, we cannot expect a cluster to have
a facility weight of exactly 1 because we generally will unable to partition those
facilities fractionally serving a client into disjoint sets with each set having a fa-
cility weight of 1. We tackle this problem by introducing another pruning phase
before clustering, where we open all facilities \( i \) with “large” \( y_i \). Hence, in this
clustering step we now only consider facilities that have “small” \( y_i \); this allows us
to pack a substantial facility weight within a cluster, without exceeding the limit
of 1.
To analyze the algorithm, we view a demand \( j \) with requirement \( r_j \) as being composed of \( r_j \) copies which have to be connected to distinct facilities. We allot to each copy a set of facilities from among those that fractionally serve \( j \), that is, a subset of \( \{ i : x_{ij} > 0 \} \), and a unique backup facility. A copy may only be assigned to a facility that is allotted to it, or to its backup facility. Again, to argue feasibility, we have to ensure that a facility is allotted to at most one copy, and we would like to allot to each copy a facility weight of 1. Although it may not be possible to simultaneously satisfy both of these requirements because of the pruning phase, we can allot a substantial facility weight to each copy. Thusly, we can upper-bound the probability of the event that no facility in the allotted set of the copy is open. This results in an approximation ratio of 2.25. To do better, we distribute facilities more evenly among the copies. We use the so-called pipage rounding technique of Ageev and Sviridenko [2004] to essentially derandomize a hypothetical randomized process in which each copy gets an equal allotment of facility weight. This yields a 2.076-approximation algorithm.

For the uniform requirement case, we improve the approximation guarantee of 1.861 [Mahdian et al. 2001] to 1.52 by building upon the algorithm of Jain et al. [2002]. The algorithm is analyzed using the dual fitting approach and we arrive at the same factor LP as Jain et al. [2002], thereby obtaining the same performance guarantees. Combined with a greedy improvement heuristic and the analysis in Mahdian et al. [2006], we get a 1.52-approximation algorithm. Using a Lagrangian relaxation technique introduced in Jain and Vazirani [2001] for the \( k \)-median version of UFL, we get a 4-approximation algorithm for the fault-tolerant \( k \)-median problem with uniform requirements.

### 2. A Simple 4-Approximation Algorithm

We first give a simple 4-approximation algorithm for the fault-tolerant facility location problem. The algorithm does not use filtering, but rather exploits the complementary slackness conditions to bound the cost of the solution in terms of both the primal and dual optimal solutions.

Let \((x, y)\) and \((\alpha, \beta, z)\) be the optimal primal and dual solutions, respectively, and \( OPT \) be the common optimal value. The primal slackness conditions are \( x_{ij} > 0 \iff \alpha_j = \beta_j + c_{ij}, y_i > 0 \iff j \beta_{ij} = f_i + z_i \). The dual slackness conditions are \( \alpha_j > 0 \iff i x_{ij} = r_j, \beta_{ij} > 0 \iff x_{ij} = y_i, \) and \( z_i > 0 \iff y_i = 1 \). We may assume without loss of generality for each client \( j \), \( \alpha_j > 0 \), so that \( i x_{ij} = r_j \). Furthermore, for every client \( j \) there is at most one facility \( i \) such that \( 0 < x_{ij} < y_i \), and we may assume this to be the farthest facility serving \( j \) because we can always “shift” the assignment of \( j \) to facilities nearer to \( j \) and ensure that this property holds.

Like the Chudak-Shmoys (CS) algorithm for UFL [Chudak and Shmoys 2003], our algorithm is based on the observation that the optimal solution is \( \alpha \)-close, that is, \( x_{ij} > 0 \iff c_{ij} \leq \alpha_j \). However, one additional difficulty encountered in using LP duality to prove an approximation ratio, which does not arise in the case of UFL, is the presence of the \(- i z_i\) term in the dual objective function. As a result, bounding the cost in terms of \( j r_j \alpha_j \) is not enough to prove an approximation guarantee. Nevertheless, the additional structure in the primal and dual solutions, resulting from complementary slackness, allows us to circumvent this difficulty;
since \( z_i > 0 \implies y_i = 1 \), we can open all such facilities \( i \) and charge the opening cost to the LP.

Throughout, we will view a client \( j \) with requirement \( r_j \) as consisting of \( r_j \) copies, each of which needs to be connected to a distinct facility. We use \( j^{(c)} \) to denote the \( c \)th copy of \( j \). We use the terms “client” and “demand” interchangeably, and also “assignment cost” and “service cost” interchangeably. The algorithm consists of two phases.

**Phase 1.** First we open all facilities with \( y_i = 1 \). Let \( L_1 \) be the set of facilities opened. For every client \( j \), if \( x_{ij} > 0 \) and \( y_i = 1 \), we connect a copy of \( j \) to \( i \). Notice that at most one copy of \( j \) is connected to any such facility. Let \( L_j = \{ i \in L_1 : x_{ij} > 0 \} \) and \( n_j = |L_j| \) be the number of copies of \( j \) connected in this phase. Note that \( n_j \leq r_j \). The following lemma bounds the cost for this phase.

**Lemma 2.1.** The cost of phase 1 is \( \sum_j n_j \alpha_j - \sum_i z_i \).

**Proof.** Each \( i \) with \( z_i > 0 \) is in \( L_1 \) and for any \( i \in L_1 \), all clients \( j \) with \( x_{ij} > 0 \) are connected to it. Since \( x_{ij} > 0 \implies \alpha_j = c_{ij} + \beta_{ij} \), for any \( i \in L_1 \), we have

\[
\alpha_j = \begin{cases} (c_{ij} + \beta_{ij}) & j : x_{ij} > 0 \\ c_{ij} + \beta_{ij} & j : x_{ij} > 0 \\ c_{ij} + f_i + z_i. & j : x_{ij} > 0 
\end{cases}
\]

The second equality follows, since \( \beta_{ij} > 0 \implies x_{ij} = y_i = 1 \). Since \( n_j = |\{ i \in L_1 : x_{ij} > 0 \}| \), the lemma follows by summing over all \( i \in L_1 \). \( \square \)

**Phase 2.** This is a simple clustering step. Let \( r_j = r_j - n_j \) be the residual requirement of \( j \). Let \( F_j = \{ i : y_i < 1, x_{ij} > 0 \} \) be the set of facilities not in \( L_1 \) that fractionally serve \( j \) in \((x, y)\). Let \( S = \{ j : r_j \geq 1 \} \). We will maintain the invariant that \( i \in F_j \implies y_i \geq r_j \) for all \( j \in S \). We iteratively do the following until \( S = \emptyset \).

**S1.** Choose \( j \in S \) with minimum \( \alpha_j \) as a cluster center.

**S2.** Order the facilities in \( F_j \) by increasing facility cost. We pick \( M \subseteq F_j \) starting from the first facility in \( F_j \) so that \( i \in M \implies y_i \geq r_j \). If \( i \in M \implies y_i \geq r_j \), we replace the last facility \( i \) in \( M \) (i.e., the furthest facility from \( j \)) by two “clones” of \( i \), called \( i_1 \) and \( i_2 \). Set \( y_{i_1} = r_j - \sum_{i \in M \setminus \{ i \}} y_i \), \( y_{i_2} = y_i - y_{i_1} \). For each client \( k \) (including \( j \)) with \( x_{ik} > 0 \), we set \( x_{i_1k}, x_{i_2k} \) arbitrarily maintaining that \( x_{i_1k} + x_{i_2k} = x_{ik}, x_{i_1k} \leq y_{i_1}, x_{i_2k} \leq y_{i_2} \). We include \( i_1 \) in \( M \), so now \( i \in M \implies y_i = r_j \).

**S3.** Open the \( r_k \) least expensive facilities in \( M \). For each client \( k \) (including \( j \)) with \( F_j \cap M = \emptyset \), we connect \( \min(r_k, r_j) \) copies of \( k \) to these open facilities, and set \( r_k = r_k - \min(r_k, r_j) \). \( F_k = F_k \setminus M \). Facilities in \( M \) and client \( j \) now effectively disappear from the input.

Figure 1 shows one iteration of steps S1 through S3. Step S2 is valid since we maintain the invariant that \( i \in F_k \implies y_i \geq r_k \) for all \( k \in S \). This is clearly true initially, and in any iteration, for any \( k \) with \( F_k \cap M = \emptyset \) and which lies in \( S \) after the iteration, we remove a facility weight of at most \( r_j \) from \( F_k \), and \( r_k \) decreases by exactly \( r_j \).
FIG. 1. One iteration of the clustering step in phase 2. Here, $j$ is the cluster center; 2 copies of $j$, $k$, and 1 copy of get connected in this iteration; $j$ and are removed from $S$ after this iteration.

We first argue that the algorithm returns a feasible solution. In phase 1, distinct copies get connected to distinct facilities, and no facility with $y_i = 1$ ever gets used in phase 2. In phase 2, we ensure that at most one clone of a facility $i$ is opened. This holds because whenever $i \in M$ is replaced by clones, its first clone is not opened in step S3: Since $i$ is included partially in $M$, it must be the most expensive facility in $M$ and because $i \in M \setminus \{i\}$ $y_i > r_j - y_i > r_j - 1$, there are at least $r_j$ facilities in $M \setminus \{i\}$ that are less expensive than $i$. Hence the clone of $i$ included in $M$ (the first clone) is not opened in step S3. Since only the second clone can be opened whenever a facility is split into clones, at most one clone of a facility is opened. It follows that a client $j$ uses a facility $i$ at most once. Thus, we get a feasible solution where each copy $j^{(c)}$ is connected to a distinct facility. We now bound the cost of the solution obtained.

**Lemma 2.2.** The facility opening cost in phase 2 is at most $\sum_i f_i y_i$.

**Proof.** Facilities are only opened in step S2 of phase 2, where we pick a set of facilities $M$ such that $i \in M$ and open the $r_j$ least expensive facilities in $M$. The cost of opening the least expensive facilities is at most $\sum_{i \in M} f_i y_i$. Also the facilities in $M$ are not reused. \hfill □

**Lemma 2.3.** Let $k^{(c)}$ be a copy of $k$ connected to facility $i$ in phase 2. Then $c_{ik} \leq 3\alpha_k$.

**Proof.** Let $M \subseteq F_j$ be the set of facilities picked in step S2 such that $i \in M$. Let $i$ be some facility in $F_k \cap M$ which is nonempty (see Figure 1). Then, $c_{ik} \leq c_{i,k} + c_{i,j} + c_{ij} \leq \alpha_k + 2\alpha_j$. The lemma now follows, since $\alpha_j \leq \alpha_k$ because $j$ was chosen as the cluster center and not $k$ (which is in $S$). \hfill □

**Theorem 2.4.** The preceding algorithm delivers a solution of cost at most $4 \cdot OPT$.

**Proof.** The facility cost in phase 2 is at most $\sum_i f_i y_i \leq OPT = \sum_j r_j \alpha_j + (\sum_j n_j \alpha_j - \sum_j z_j)$. The service cost of $j$ is the service cost for the $n_j$ copies
connected in phase 1 added to the service cost for the $r_j$ copies connected in phase 2. Each copy of $j$ connected in phase 2 incurs a service cost of at most $3\alpha_j$. So, the total cost is bounded by (cost of phase 1) + (facility cost in phase 2) + (service cost for $r_j$ copies in phase 2) \leq (\sum_i n_j \alpha_j - \sum_i z_i) + \sum_i f_i y_i + 3 \sum_j r_j \alpha_j \leq 2(\sum_i n_j \alpha_j - \sum_i z_i) + 4 \sum_j r_j \alpha_j \leq 4(\sum_i n_j \alpha_j - \sum_i z_i) = 4 \cdot OPT.$$

3. A Better “Randomized” Algorithm: An Overview

We now show that the performance guarantee can be improved substantially by using randomization along with clustering. At a high level, the algorithm proceeds as follows. First we run phase 1 as before, except that we connect a copy of $j$ to $i$ only if $x_{ij} = 1$. The main source of improvement is due to the fact that we open every other facility $i$ with probability proportional to $y_i$. This helps to reduce the service cost, since now for every client $j$ and copy $j^{(c)}$ there is a significant probability that a (distinct) facility with $x_{ij} > 0$ is open. The algorithm is thus in the spirit of the CS algorithm [Chudak and Shmoys 2003], which also uses randomization to reduce the service cost incurred. However, several obstacles have to be overcome to extend the approach to the fault-tolerant setting and thereby prove a good approximation guarantee.

We again cluster facilities around demand points, but now each cluster that we create contains a (fractional) facility weight close to 1, and we open at least one facility in the cluster by a randomized process. We will ensure that each demand $j$ has $r_j$ clusters “near” it, so that the facilities opened in these clusters, called backup facilities, can be used to serve $j$ without blowing up the service cost by much. This is done by introducing a notion of backup requirement, which is initialized to $r_j$. Whenever we create a cluster, we decrement the backup requirement of all demands $j$ that share a facility with the cluster created. The facility opened in this cluster (which is chosen randomly) serves as a backup facility for each such client $j$. As long as the backup requirement of $j$ is at least 1, it is a candidate for being chosen as a cluster center; thus at the end, $j$ will share facilities with $r_j$ clusters and these provide $r_j$ nearby backup facilities.

The randomization step, however, causes various technical difficulties. To argue feasibility, we need to open different facilities in different clusters. Also, ideally, we would like each cluster to contain a facility weight of exactly 1. If the facility weight is too small, then we incur a huge cost relative to the LP in opening a facility from the cluster; if the weight is more than 1, then we are using up a fractional facility weight of more than 1 while opening only a single facility, so we might not open enough facilities to satisfy the requirement of a client. In a deterministic setting like that in the 4-approximation algorithm described earlier, we know precisely which facility(ies) is (are) opened within a cluster, and can therefore afford to split facilities across clusters without sacrificing feasibility. Thusly, we can ensure that a cluster contains the “right” amount of facility weight. Guha et al. [2003] also deterministically decide which facility to open within a cluster, possibly splitting a facility across clusters, and thereby relatively easily extend the UFL algorithm of Shmoys et al. [1997] to the fault-tolerant setting.

With randomization, on the other hand, any of the facilities in a cluster might get opened. Therefore we cannot now split facilities across clusters, and require that the clusters be disjoint. However, unlike UFL, once we require clusters to be
disjoint, we cannot ensure that a cluster has a facility weight of exactly 1. For example, consider a client \( j \) with \( r_j = 2 \) served by three facilities \( i, i', \) and \( i'' \) with \( x_{ij} = x_{i'j} = x_{i''j} = y_i = y_{i'} = y_{i''} = \frac{2}{3} \); a cluster centered at \( j \) consisting of unsplit facilities cannot have a facility weight of exactly 1. We tackle this problem by introducing an intermediate phase 2 (before the clustering step), where we open all facilities \( i \) for which \( y_i \) is “large”; that is, \( y_i \) is at least some threshold \( \gamma \), and we connect a copy of \( j \) to \( i \) if \( x_{ij} \geq \gamma \). We now work with only the remaining set of facilities and the residual requirements, and perform the aforesaid clustering step. Clearly, we incur a loss of a factor of at most \( \gamma \) due to phase 2. But importantly, since each (remaining) \( y_i < \gamma \), we can now create disjoint clusters and ensure that a cluster contains a facility weight between \( 1 - \gamma \) and 1. Finally, we open every facility \( i \), be it a cluster facility or a noncluster facility, with probability proportional to \( y_i \).

To analyze the cost of the solution, we fix a particular way of assigning each copy (that is unassigned after phases 1 and 2) to an open facility, and separately bound the service cost for every copy. For each demand \( j \), we allot to each such copy \( j^{(c)} \) a set of preferred facilities \( P(j^{(c)}) \), which is a subset of those facilities that fractionally serve the unassigned copies, as well as a distinct backup facility \( b(j^{(c)}) \), which is a facility opened from a cluster near \( j \). We assign \( j^{(c)} \) to the nearest facility open in \( P(j^{(c)}) \) (if there is one) and otherwise to the backup facility \( b(j^{(c)}) \). Again, to argue feasibility we require that the preferred sets for the different copies are disjoint. Ideally, we would like to allot a disjoint set of facilities with facility weight \( 1 \) to each preferred set, but we face the same difficulty as in forming clusters: It might not be possible to divide the facilities among the different copies so that each copy gets a set with facility weight \( 1 \). However, phase 2 ensures that we can allot to each set \( P(j^{(c)}) \) a facility weight of at least \( 1 - \gamma \), which gives a reasonable upper bound on the probability that no facility in \( P(j^{(c)}) \) is open. Combining these various components, we thereby obtain an algorithm with a much better approximation ratio of about \( 2.2 \), but we can do even better.

3.1. Pipage Rounding. The final improvement comes by exploiting the pipage rounding technique of Ageev and Sviridenko [2004], which was applied in the context of uncapacitated facility location by Sviridenko [2002]. Suppose that we distribute those facilities serving the unassigned copies of \( j \) among the preferred sets and allot to each preferred set a facility weight of \( 1 \), perhaps by splitting facilities. A facility \( i \) could now lie in multiple preferred sets \( P(j^{(c)}) \); let \( z_{ij}(c) \) be the extent to which facility \( i \) is allotted to \( P(j^{(c)}) \), so \( \sum c z_{ij}(c) = x_{ij} \). Although the preferred sets are no longer disjoint, we can still use the aforesaid scheme of opening facilities and assigning copies to facilities as follows: We will make facility \( i \) available (for use) to exactly one copy \( c \) and with probability \( z_{ij}(c)/x_{ij} \). So for copy \( j^{(c)} \) to be assigned to facility \( i \), it must be that \( i \) is open, \( i \) is available to copy \( c \), and no facility in \( P(j^{(c)}) \) that is nearer to \( j \) is available to copy \( c \). So in expectation, each copy \( j^{(c)} \) has a facility weight of \( 1 \) available to it, and it seems plausible that we should be able to show the probability low of there being an available facility in the preferred set, thereby bounding the expected service cost of the copy. However, there is some dependence between the randomness involved in making a facility available to a copy, and in opening the facility, which makes it difficult to prove a good bound on the expected service cost of a copy.

Nevertheless, we will pretend that we have a hypothetical randomized process with various desired properties, writing an expression for the expected cost incurred.
under this randomized process and bounding this cost. More precisely, we will construct an expression \( \text{cost}(y_1, y_2, \ldots, y_n) \) that is a function of the \( y_i \) variables, where \( n = |F| \), satisfying the following properties.

**P1.** When the \( y_i \) values are set to those values given by the LP optimal solution, we have \( \text{cost}(y_1, \ldots, y_n) \leq c \cdot \text{OPT} \) for an appropriate constant \( c \) that we will specify later.

**P2.** For any integer solution satisfying certain properties, if we consider the corresponding \( \{0, 1\} \)-setting of the \( y_i \) values, \( \text{cost}(y_1, \ldots, y_n) \) gives an upper bound on the total cost of the integer solution.

**P3.** Furthermore, \( \text{cost}(\cdot) \) has some nice concavity properties (we will make these precise later).

Using property P3, we will argue that, given any initial fractional setting of the \( y_i \) values, we can obtain a \( \{0, 1\} \)-solution \( \tilde{y} \) satisfying certain properties, such that \( \text{cost}(\tilde{y}_1, \ldots, \tilde{y}_n) \leq \text{cost}(y_1, \ldots, y_n) \). So, if we set the \( y_i \) values to those values given by the LP optimal solution to begin with, then properties P1 and P2 show that we obtain an integer solution of cost at most \( c \cdot \text{OPT} \), thereby getting a \( c \)-approximation algorithm. Thus, we actually get a deterministic algorithm\(^1\) with an approximation guarantee of 2.076.

As mentioned earlier, we will obtain expression \( \text{cost}(\cdot) \) by imagining that we have a randomized process with certain desired properties, and writing out the expected cost incurred under this process. We emphasize that this is for intuitive purposes only; such a randomized process may not exist, and even if it does, we might not know how to implement such a randomized process.

4. Algorithm Details

The algorithm runs in three phases which are described in detail next (the entire algorithm is also summarized in Figure 3). Let \( \gamma < \frac{1}{2} \) be a parameter whose value we will fix later.

**Phase 1.** This is very similar to phase 1 of the simple 4-approximation algorithm. Let \( L_1 = \{ i : y_i = 1 \} \). We open all facilities in \( L_1 \). For every client \( j \), if \( x_{ij} = y_i = 1 \), we connect a copy of \( j \) to \( i \). Let \( n_j \) be the number of copies of \( j \) so connected, and let \( r_j = r_j - n_j \) be the residual requirement of \( j \).

**Phase 2.** Open each facility \( i \) (in \( F \setminus L_1 \)) with \( y_i \geq \gamma \). Let \( L_2 \) be the set of facilities opened. For a client \( j \), we now define \( L_j = \{ i : \gamma \leq x_{ij} < 1 \} \). Clearly \( L_j \subseteq L_1 \cup L_2 \). Connect \( \min(|L_j|, r_j) \) copies of \( j \) to distinct facilities in \( L_j \). This incurs the loss of a factor of \( \frac{1}{\gamma} \) compared to the LP. We ensure that for each \( j \), no facility \( i \) with \( x_{ij} \geq \gamma \) is used after this phase. Let \( r_j = \max(r_j - |L_j|, 0) \) be the residual requirement of \( j \). Note that \( i : x_{ij} < \gamma \), \( x_{ij} \geq r_j \), since if \( r_j > 0 \), then \( r_j = r_j - |L_j| \) and \( r_j = i : x_{ij} < 1 \).

**Phase 3.** We introduce some notation first. Let \( F_j \) denote the set of facilities \( \{ i : 0 < x_{ij} < \gamma \} \) sorted in order of increasing \( c_{ij} \). We define the facility weight of

---

\(^1\)This is why the word of randomized appears in quotes in the title of this section.
a set of facilities $S$ to be \( \text{facwt}(S, j) = \sum_{i \in S} x_{ij} \). We know that \( \text{facwt}(F_j, j) \geq r_j \); we assume without loss of generality that \( \text{facwt}(F_j, j) = r_j \) (If not, we may simply take a subset of \( F_j \) starting from the first facility and proceeding in order until \( \text{facwt}(F_j, j) = r_j \), where the last facility may only be partially included in \( F_j \)). We describe in the following how to open facilities; once we know the set of open facilities, we assign each client \( j \)'s remaining \( r_j \) copies to the \( r_j \) open facilities nearest to it to which it is not already assigned.

**Clustering.** Define the backup requirement \( \text{back}(j) = r_j \). For each client \( j \), let \( a_j \) initialized to 1 denote the current “active” copy of \( j \), and let \( N_j \), initialized to \( F_j \), be the current set of unclustered facilities in \( F_j \) ordered by increasing \( c_{ij} \) value. We will maintain the invariant \( \text{facwt}(N_j, j) \geq \text{back}(j) \geq 1 \) (see Lemma 5.1) for every \( j \). Let \( S = \{ j : \text{back}(j) \geq 1 \} \) be the set of candidate cluster centers.

While \( S = \emptyset \) we repeatedly do the following.

**C1.** For each \( j \in S \), let \( C_j(a_j) \) denote the average distance from \( j \) to the first \( l \) facilities (considered in sorted order) \( i_1, \ldots, i_l \) in \( N_j \) that gather a net \( x_{ij} \) weight of 1 (this makes sense because \( \text{facwt}(N_j, j) \geq \text{back}(j) \geq 1 \) where the last facility \( i_l \) may be included partially, that is, to an extent \( x \) such that \( p_{<l} x_{i_l, j} + x = 1, 0 < x \leq x_{i_l, j} \). So \( C_j(a_j) = \sum_{p < l} c_{ip, j} x_{ip, j} + c_{il, j} x \).

**C2.** Choose \( j \in S \) with minimum \( C_j(a_j) \) as a cluster center. Form a cluster \( M \subseteq N_j \) consisting of the first \( m \) facilities \( i_1, \ldots, i_m \) in \( N_j \) such that \( \text{facwt}(\{i_1, \ldots, i_m\}, j) \geq 1 - \gamma \). Note that here we do not split any facility (see Figure 2).

**C3.** For each \( k \) (including \( j \)) such that \( N_k \cap M = \emptyset \), decrease \( \text{back}(k) \) by 1. Initialize \( P(k^{(\alpha)}) = F_k \cap M = N_k \cap M \) (since facilities in \( M \) are previously unclustered) and set \( a_k = a_k + 1, N_k = N_k \setminus M \) (see Figure 2). For each such \( k \), we call \( M \) the backup cluster of copy \( k^{(a_k)} \).

**Pipage rounding.** For every client \( j \), we augment the preferred sets \( P(j^{(c)}) \), \( c = 1, \ldots, r_j \), so that each \( P(j^{(c)}) \) gets a facility weight of exactly 1. We do this by

**FIG. 2.** An iteration of the clustering step in phase 3.
Phase 1. Let $L_1 = \{i : y_i = 1\}$. Open all the facilities in $L_1$. For each client $j$ and each $i$ such that $x_{ij} = 1$, connect a copy of $j$ to $i$. Let $n_j = |\{i : x_{ij} = 1\}|$ and let $r'_j = r_j - n_j$. 

Phase 2. Let $L_2 = \{i : \gamma \leq y_i < 1\}$. Open all the facilities in $L_2$. For a client $j$, define $L_j = \{i : \gamma \leq x_{ij} < 1\}$. Connect $\min(\|L_j\|, r'_j)$ copies of $j$ to distinct facilities in $L_j$. Let $r'_j = \max(r'_j - |L_j|, 0)$ be the residual requirement of $j$. (Note that $\sum_{i \in U_j \cap \gamma} x_{ij} \geq \gamma$.)

Phase 3. For every $j$, let $F_j$ denote the facilities $\{i : 0 < x_{ij} < \gamma\}$ ordered by increasing $\gamma_j$ value. Define the facility weight of a set of facilities $\gamma$ by $\text{facwt}(\gamma) = \sum_{i \in \gamma} x_{ij}$. We may assume that $\text{facwt}(F_j, j)$ is exactly $r'_j$. 

Clustering. For each client $j$, initialize $\gamma_j \leftarrow r'_j$ and $a_j \leftarrow 1$; $\gamma_j$ denotes the current “active” copy of $j$. Let $N_j \leftarrow F_j$ be the current set of unclustered facilities in $F_j$ (ordered by increasing $c_{ij}$ value). Let $\mathcal{S} = \{j : \gamma_j \geq 1\}$ be the set of candidate cluster centers.

While $\mathcal{S} \neq \emptyset$:

C1. For each $j \in \mathcal{S}$, define $\gamma_j$ as follows. Let $i_1, \ldots, i_t$ be the first $t$ facilities in $N_j$ such that $\sum_{i \leq t} x_{ij} > 1 \leq \sum_{i > t} x_{ij}$. Define $\gamma_j = \gamma_j + c_{ij} \gamma_j + \gamma_j (1 - \sum_{i > t} x_{ij})$.

C2. Choose $j \in \mathcal{S}$ with minimum $\gamma_j$ as a cluster center. Form cluster $M \subseteq N_j$ consisting of the first $m$ facilities $i_1, \ldots, i_m$ in $N_j$ such that $\text{facwt}(\{i_1, \ldots, i_m\}, j) \geq 1 - \gamma$ (see Figure 2).

C3. For each $k$ (including $j$) such that $N_k \cap M \neq \emptyset$, initialize $P(k^{(\alpha_k)}) \leftarrow F_k \cap N \in N_k \cap M$. Update $\gamma_j \leftarrow \gamma_j - 1$, $a_j \leftarrow a_j + 1$, and $N_k \leftarrow N_k \setminus M$ (see Figure 2). We call $M$ the backup cluster of copy $k^{(\alpha_k)}$.

Pipage Rounding. For every client $j$, distribute the facilities remaining in $N_j$ arbitrarily among the preferred sets $P(j^{(c)})$, for $c = 1, \ldots, r'_j$ copies, splitting facilities across sets if necessary, so that each preferred set gets a facility weight of exactly 1. More precisely, compute an assignment $\{z_{ij}(c)\}_{i,c}$ (see Figure 4) such that

(i) $z_{ij}(c) \geq 0$ and $\sum_{c} z_{ij}(c) = 1$ for every copy $c$.

(ii) $\sum_{c} z_{ij}(c) = x_{ij}$ for every facility $i$, and

(iii) if $i \in F_j \setminus N_j$ was allotted to $P(j^{(c)})$ in the clustering step, then $z_{ij}(c) = x_{ij}$, and $P(j^{(c)}) = \{i : z_{ij}(c) > 0\} > 0$. Let $\{x_{ij}, y_{ij}, z_{ij}(c)\}$ denote respectively $\{x_{ij}, y_{ij}, z_{ij}(c)\} / (1 - \gamma)$.

Local Per-Client Rounding. For every client $j$, copy $c$, let $S^{(c)}_j$ be as defined by (2) if $j^{(c)}$ is a cluster center, and as defined by (3) otherwise. For every $i$ and copies $c, c'$ such that $0 < z_{ij}(c), z_{ij}(c') < x_{ij}$, perturb $z_{ij}(c)$ by $\epsilon$ and $z_{ij}(c')$ by $-\epsilon$, where $\epsilon$ is either $-z_{ij}(c)$ or $z_{ij}(c')$, so that $\sum_{c} S^{(c)}_j$ does not increase. Repeat this until $z_{ij}(c) \in \{0, x_{ij}\}$ for every $i, j, c$.

Global Rounding. Define $S^{(g)}_j = \sum_{c} S^{(c)}_j$ if $j^{(c)}$ is a cluster center, and by (4) otherwise. Define $T(y_1, \ldots, y_n) = \sum_{i \in \gamma} f_{ij}$, and $\text{cost}(y_1, \ldots, y_n) = T(y_1, \ldots, y_n) + \sum_{c} S^{(g)}_j$. Define $b_{i,c}(e') = \text{cost}(y_1, \ldots, y_i - e) - \text{cost}(y_1, \ldots, y_i) + e$.

While there exists a cluster $M$ with no fully open facility do the following: Pick indices $i', c'$ such that $y_{i'}, y_{i'} \in (0, 1)$. Let $e_1 = \min(y_{i'}, 1 - y_{i'})$ and $e_2 = \min(1 - y_{i'}, y_{i'})$. Compute $e' \in [-e_1, e_2]$ such that $b_{i',c'}(e') = \min_{e \in [e_1, e_2]} b_{i',c'}(e)$ and update $y_{i'} \leftarrow y_{i'} + e'$, $y_{i'} \leftarrow y_{i'} + e'$. For each remaining fractional $y_{i'}$, round $y_{i'}$ to 0 or 1, whichever decreases the value of $\text{cost}(\cdot)$.

For each (remaining) copy $j^{(c)}$ (where $c = 1, \ldots, r'_j$), if $j^{(c)}$ is a cluster center assign it to the nearest facility opened from that cluster; otherwise assign $j^{(c)}$ to the nearest open facility in $P(j^{(c)})$ if one is open, and to the nearest open facility from its backup cluster otherwise.

Fig. 3. Summary of the “randomized” algorithm for FTFL.
(a) Each facility $i \notin (L_1 \cup L_2)$ (so $y_i < \gamma$) is opened independently with probability $\hat{y}_i$. Note that $\hat{y}_i < 1$, since $y_i < \gamma \leq 1 - \gamma$.

(b) Within each cluster $M$, at least one facility is opened.

(c) For any client $j$, at most one of its copies gets to use a facility, and copy $c$ gets to use facility $i \in P(j^{(c)})$ with probability $\hat{z}_{ij}(c)$.

As mentioned earlier, this reference to an imaginary randomized process is for intuitive purposes only. In particular, notice that no randomized process can satisfy properties (a) and (b) simultaneously. The rounding is performed in two steps.

**Local per-client rounding.** We first ensure that for every client $j$, every facility $i$ is allotted to exactly one preferred set; so after this step, we will have $\hat{z}_{ij}(c)$ equal to either 0 or $\hat{x}_{ij}$ for every $i$. Fix a client $j$. Motivated by the aforementioned hypothetical randomized process, we write an expression $S_{loc}^{(c)}(j)$ for the service cost of each copy $j^{(c)}$. Suppose $j^{(c)}$ is the center of a cluster $M$, so $M \subseteq P(j^{(c)})$. Let $M = \{i_1, \ldots, i_m\}$ with $c_{i_{l-1}j} \leq \ldots \leq c_{i_mj}$, and let $d_l = c_{i_lj}$. One of the facilities in $M$ is guaranteed to be open, and we assign $j^{(c)}$ to the nearest such facility. In Lemma 5.1, we show that $y_i = x_{ij} = \hat{z}_{ij}(c)$ for each facility $i$ in $M$, so we can write

$$S_{loc}^{(c)}(j) = \hat{y}_i d_1 + (1 - \hat{y}_i) \hat{y}_i d_2 + \cdots + (1 - \hat{y}_i)(1 - \hat{y}_i)\cdots(1 - \hat{y}_{i_{m-1}}) d_m.$$  \hfill (2)

To avoid clutter, we have not explicitly indicated the functional dependence of $S_{loc}^{(c)}(j)$ on the variables $\hat{y}_i, \ldots, \hat{y}_{i_m}$.

If $j^{(c)}$ is not a cluster center, let $P(j^{(c)}) = \{i_1, \ldots, i_m\}$ ordered by increasing distance from $j$, and let $d_l = c_{i_lj}$. Let $M$ be the backup cluster of $j^{(c)}$, and let $M \setminus P(j^{(c)}) = \{b_1, \ldots, b_q\}$, again ordered by increasing $c_{b_j}$ value. Denote $c_{b_j}$ by $c_l$ for $l = 1, \ldots, q$. To keep notation simple, let $\hat{z}_{i_l}$ denote $\hat{z}_{i_lj}(c)$. We define

$$S_{loc}^{(c)}(j) = \hat{z}_{i_l} d_1 + (1 - \hat{z}_{i_l}) \hat{z}_{i_l} d_2 + \cdots + (1 - \hat{z}_{i_l})(1 - \hat{z}_{i_l})\cdots(1 - \hat{z}_{i_{m-1}}) \hat{z}_{i_{m-1}} d_m$$
$$+ (1 - \hat{z}_{i_l}) \cdots (1 - \hat{z}_{i_m}) (\hat{y}_{b_1} c_1 + \cdots + (1 - \hat{y}_{b_1}) \cdots (1 - \hat{y}_{b_{q-1}}) c_q).$$  \hfill (3)
The rationale behind the expression is the same as earlier: We assign \( j^{(c)} \) to the nearest open facility in \( P(j^{(c)}) \) and if no such facility is open (in which case, some facility in \( M \setminus P(j^{(c)}) \) must be open), then to the nearest facility opened from cluster \( M \).

Now for each \( j \), we round the \( \hat{z}_{ij}(c) \) values without increasing \( c \cdot S_j^{\text{loc}}(c) \), so that at the end we get an unsplittable allotment of facilities in \( F_j \) to copies; that is, for every \( i \in F_j \) there will be exactly one copy \( c \) with \( \hat{z}_{ij}(c) > 0 \) (and hence equal to \( \hat{x}_{ij} \)).

Observe that the expression \( S_j^{\text{loc}}(c) \) is linear in each variable \( \hat{z}_{ij}(c) \). (This is also true when \( j^{(c)} \) is a cluster center and \( i \) is not part of the cluster centered at \( j^{(c)} \).)

Suppose \( 0 < \hat{z}_{ij}(c) < \hat{x}_{ij} \) for some copy \( c \). There must be some other copy \( c \) such that \( 0 < \hat{z}_{ij}(c) < \hat{x}_{ij} \). So if we consider changing \( \hat{z}_{ij}(c) \) by \( + \) and \( \hat{z}_{ij}(c) \) by \( - \), where \( \hat{z}_{ij}(c) \) is either \( -\hat{z}_{ij}(c) \) or \( +\hat{z}_{ij}(c) \), then since \( S_j^{\text{loc}}(c) \) and \( S_j^{\text{loc}}(c) \) are linear in \( \hat{z}_{ij}(c) \) and \( \hat{z}_{ij}(c) \), respectively, we can always make one of these local moves without increasing \( c \cdot S_j^{\text{loc}}(c) \). In fact, notice that the values of \( S_j^{\text{loc}}(c) \) for copies \( c = c, c \), as well as that of \( S_k^{\text{loc}}(c) \) for any copy \( j^{(c)} \) where \( k = j \), remain unchanged. Thus we decrease the number of \( \hat{z}_{ij}(\gamma) \) values that lie in the interval \((0, x_{ij})\). Continuing in this way we get that at the end there is exactly one copy \( c \) with \( \hat{z}_{ij}(c) = \hat{x}_{ij} > 0 \); for every other copy \( c \) we have \( \hat{z}_{ij}(c) = 0 \). We repeat this for every facility \( i \in F_j \), and for every client \( j \), to get an unsplittable allotment for each client. Figure 4 shows a possible outcome of this process. Note that the \( \hat{y}_i \) values are not changed in this step.

**Global rounding.** Now we round the \( \hat{y}_i \) variables to 0-1 values. Each facility \( i \) with \( \hat{y}_i < \gamma \) is opened with probability \( \hat{y}_i \), so we can write the facility cost as

\[
T(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n) = \sum_{i \in [n]} \hat{y}_i, \quad \text{where } n = |\mathcal{F}|.
\]

The service cost of a copy \( j^{(c)} \) is given by the expression \( S_j^{\text{loc}}(c) \) at the end of the local rounding step. For a cluster center \( j^{(c)} \), we set \( S_j^{\text{loc}}(c) = S_j^{\text{loc}}(c) \). For a noncluster-center copy \( j^{(c)} \) we modify the expression for \( S_j^{\text{loc}}(c) \) slightly. Let \( P(j^{(c)}) \) consist of the facilities \( \{i_1, \ldots, i_m\} \) after the previous step, ordered by increasing distance from \( j \). Recall that \( P(j^{(c)}) \) only contains those facilities for which \( \hat{z}_{ij}(c) > 0 \) for any \( i \in P(j^{(c)}) \). We have \( \hat{z}_{ij}(c) = \hat{x}_{ij} \) and this is equal to \( \hat{y}_i \) for all but at most one facility. Therefore, the value of \( S_j^{\text{loc}}(c) \) depends only on the \( \hat{y}_i \) values and perhaps on one \( \hat{x}_{ij} \) value; we would like to get an expression for the service cost of \( j^{(c)} \) that depends only on the \( \hat{y}_i \) values. So, we modify the expression for \( S_j^{\text{loc}}(c) \) as follows: We substitute \( \hat{x}_{ij} \) with \( \hat{y}_i \), wherever it occurs. We use \( S_j^{\text{glob}}(c) \) to denote the new expression. More precisely, let \( M \) be the backup cluster of \( j^{(c)} \) with \( M \setminus P(j^{(c)}) = \{b_1, \ldots, b_q\} \) sorted by increasing \( c_{ij} \) value, let \( d_l \) denote \( c_{il} \) for \( l = 1, \ldots, m \), and \( c_l \) denote \( c_{lb} \) for \( l = 1, \ldots, q \). We define

\[
S_j^{\text{glob}}(c) = \hat{y}_{i_1}d_1 + (1 - \hat{y}_{i_1})\hat{y}_{i_2}d_2 + \cdots + (1 - \hat{y}_{i_1})(1 - \hat{y}_{i_2})\cdots(1 - \hat{y}_{i_{m-1}})\hat{y}_{im}d_m
+ (1 - \hat{y}_{i_1})\cdots(1 - \hat{y}_{i_{m-1}})\hat{y}_{ib_1}c_1 + \cdots + (1 - \hat{y}_{i_1})\cdots(1 - \hat{y}_{i_{m-1}})c_q. \quad (4)
\]

We show in Lemma 5.7 that this modification does not increase the cost, that is, \( S_j^{\text{glob}}(c) \leq S_j^{\text{loc}}(c) \). The total cost is therefore given by

\[
\text{cost}(\hat{y}_1, \ldots, \hat{y}_n) = T(\hat{y}_1, \ldots, \hat{y}_n) + \sum_{j,c} S_j^{\text{glob}}(c).
\]
We now convert the \( \hat{y}_i \) values to \( \{0, 1\}\)-values without increasing \( \text{cost}(\hat{y}_1, \ldots, \hat{y}_n) \). Observe that for a \( \{0, 1\}\)-setting of the \( \hat{y}_i \) values, \( T(\hat{y}_1, \ldots, \hat{y}_n) \) is precisely the facility cost of the solution. Moreover, as long as the \( \{0, 1\}\)-setting is such that each cluster contains an open facility, for each client \( j \) and for each copy \( c \), \( S^{gb}(c) \) is clearly an upper bound on the service cost of copy \( j^{(o)} \). Hence \( S^{glb}(c) \) is an upper bound on the service cost of \( j \). Therefore, \( \text{cost}(.) \) gives an upper bound on the total cost of the solution, satisfying property P2 of pipage rounding (Section 3.1).

For any two indices \( i < i' \), define
\[
h_{i,i'}(\cdot) = \text{cost}(\hat{y}_1, \ldots, \hat{y}_{i-1}, \hat{y}_i + \epsilon, \hat{y}_{i+1}, \ldots, \hat{y}_{i'-1}, \hat{y}_{i'}, \hat{y}_{i'+1}, \ldots, \hat{y}_n). \tag{5}
\]

In the analysis, we show that \( h_{i,i'}(\cdot) \) is concave in \( \epsilon \) in the range \([-\theta_1, \theta_2]\), where \( \theta_1 = \min(\hat{y}_i, 1-\hat{y}_i) \) and \( \theta_2 = \min(1-\hat{y}_i, \hat{y}_i) \). So, \( h_{i,i'}(\cdot) \) attains its minimum value (which is at most \( h_{i,i'}(0) = \text{cost}(\hat{y}_1, \ldots, \hat{y}_n) \)) at one of the endpoints \( \epsilon = -\theta_1 \) or \( \epsilon = \theta_2 \) and we can update \( \hat{y}_i \leftarrow \hat{y}_i + \epsilon \), \( \hat{y}_i \leftarrow \hat{y}_i - \epsilon \) without increasing \( \text{cost}(\cdot) \). This decreases the number of fractional \( \hat{y}_i \) values by one, and by continuing in this way we eventually get an integer solution.

This is the basic scheme we employ, but we choose the indices carefully so as to ensure that each cluster will contain at least one (fully) open facility. As long as there is some cluster which does not have a fully open facility, we do the following: Choose such a cluster \( M \), pick indices \( i \) and \( i' \) corresponding to any two fractionally open facilities in \( M \), and convert one of the \( \hat{y}_i, \hat{y}_i' \) values to an integer. Since \( \sum_{i \in M} y_i \geq 1 - \gamma \implies \sum_{i \in M} \hat{y}_i \geq 1 \) (this is true before we examine cluster \( M \)), and the sum \( \sum_{i \in M} \hat{y}_i \) does not change when we modify the \( \hat{y}_i \) values for facilities in \( M \), we will eventually open some facility in \( M \) to an extent of 1. Also note that if \( M \) contains no fully open facility, then there must be at least two fractionally open facilities in \( M \). After taking care of all clusters this way, we round the remaining \( \hat{y}_i \) values by picking any facility \( i \) such that \( 0 < \hat{y}_i < 1 \) and rounding it to either 1 or 0, whichever decreases the value of \( \text{cost}(\cdot) \). (Note that \( \text{cost}(\cdot) \) is linear in each variable \( \hat{y}_i \).)

**Remark 4.1** Once every cluster has a fully open facility, we can also do the following: Consider the randomized process that opens each facility \( i \) such that \( 0 < \hat{y}_i < 1 \) independently with probability \( \hat{y}_i \). The expected service cost of a copy \( j^{(o)} \) (under this randomized process) is bounded by \( S^{gb}(c) \), so this gives a randomized algorithm with the same performance guarantee.

5. Analysis

The analysis proceeds as follows. First, we prove some basic properties about the clustering step (Lemma 5.1). Next, in Lemma 5.3, we show that every facility in a backup cluster of a client is close to the client. Lemma 5.5 establishes some crucial properties about an expression of the form \( S^{glb}(c) \), including the concavity property that is exploited in the global rounding step. Using these properties along with Lemma 5.3, we bound the value of \( \sum_{c} S^{loc}(c) \) for client \( j \) at the beginning of the local rounding step in Lemma 5.6 and thus bound the total service cost for \( j \). We also argue that going from the “local” expression \( S^{loc}(c) \) to the “global” expression \( S^{gb}(c) \) does not increase the cost (Lemma 5.7). Finally, Theorem 5.8 puts the various pieces together and proves the bound on the approximation ratio.
LEMMA 5.1. The following invariants are maintained during the clustering step in phase 3.

(i) For any client $k$, $\text{back}(k) \leq \text{facwt}(N_k, k)$.
(ii) Each client $k$ has at least $r_k - \text{back}(k)$ clusters designated as backup clusters.
(iii) For every clustered facility $i$, if $i$ is part of a cluster centered at some client $k$, then $x_{ik} = y_i$.

PROOF. The proof is by induction on the number of iterations. At the beginning of phase 3, (i) holds, and (ii) and (iii) hold vacuously. Suppose the lemma holds for each copy.

Recall that clustering step, each copy.

LEMMA 5.1.

PROOF. If $i$ is in $\text{facwt}(N_k, k)$, then $\text{back}(k)$ decreases by exactly 1. For any other client $k$, $\text{back}(k)$, $\text{facwt}(N_k, k)$, and the number of backup clusters remain the same.

Since the clusters created are disjoint, Lemma 5.1 shows that at the end of the clustering step, each copy $k^{(c)}$ has an allocated distinct backup cluster allocated. Recall that $L_1$ is the set of facilities $\{i : y_i = 1\}$ and $L_2 = \{i : \gamma \leq y_i < 1\}$.

COROLLARY 5.2. (i) If $i \in L_1 \cup L_2$, then $i$ is not part of any cluster; and (ii) for any client $k$ and copy $c$, if $M$ is the backup cluster of $k^{(c)}$, then for every other copy $c'$, we always have $P(k^{(c')}) \cap M = \emptyset$.

PROOF. If $i$ lies in $L_1 \cup L_2$ and $i$ is part of some cluster centered around a client $j$, then $i \in F_j$; also $x_{ij} = y_i \geq \gamma$ by part (iii) of Lemma 5.1, which contradicts the definition of $F_j$.

$P(k^{(c')})$ is initialized to $F_k \cap M$ in step C3, so no other preferred set $P(k^{(c')})$ can contain a facility from $M$ after the clustering step, or after the distribution of facilities at the beginning of the rounding step. During the pipage rounding step, no "new" facility is ever added to $P(k^{(c')})$, where a new facility denotes a facility $i$ for which $z_{ik}(c) = 0$ after the initial allotment.

Corollary 5.2 shows that no facility used by client $j$ in phases 1 and 2 is reused in phase 3. In phase 3, copies of $j$ are connected either to facilities in $F_j$ or to cluster facilities, neither of which could have been used in phases 1 and 2.

LEMMA 5.3. Let $k^{(c)}$ be any copy $c$ of client $k$, and $M$ be the backup cluster of this copy. Then, for any facility $i \in M$, we have $c_{ik} \leq \alpha_k + 2C_k(c)/\gamma$.

PROOF. Let $j$ be the center of cluster $M$ and $a_j$ be the active copy of $j$ when $M$ was created. Since $M$ is the backup cluster of copy $k^{(c)}$, we know that $\alpha_k$ at this point. Let $i$ be a facility in $F_k \cap M$ (which is nonempty). Since $x_{ik} > 0$, we have $c_{ij} \leq \alpha_k$ by complementary slackness. We will show that $\max_{i \in M} c_{ij} \leq \alpha_k + 2C_k(c)/\gamma$. Thus, $c_{ik} \leq \alpha_k + 2C_k(c)/\gamma$.
Consider any two indices \( i \leq \). Let \( \theta_1 = \min(p_i, 1 - p_i) \), \( \theta_2 = \min(1 - p_i, p_i) \). Then \( (\cdot) \) is concave in \([-\theta_1, \theta_2]\).

**Proof.** Part (i) was proved by Sviridenko [2002] using the Chebyshev integral inequality [Hardy et al. 1952]. We include a proof for completeness. The Chebyshev integral inequality states the following: Let \( g_1, g_2 \) be functions from the interval \([a, b]\) to \( \mathbb{R}_+ \), where \( g_1 \) is monotonically nonincreasing and \( g_2 \) is monotonically nondecreasing.
Adding Eq. (6) over all copies

We will use this to bound the sum of the first $m$ terms of $E(.)$ by $(1-p)\prod_{i=1}^{m}(1-p_i)$, where $g_1(x)$ and $g_2(x)$ are functions defined on the interval $[0, P = \prod_{i=1}^{m} p_i]$, with $g_1(x) = \prod_{i=1}^{j} (1 - p_i)$ and $g_2(x) = d_i$ over the interval $[\prod_{i=1}^{j} p_i, 1]$. This gives

$$p_1d_1 + (1-p_1)p_2d_2 + \ldots + (1-p_1)(1-p_2)\ldots(1-p_m)d_m$$

$$= \prod_{i=1}^{m} p_i d_i = 1 - (1-p_1) .$$

To show part (ii), we write $E(.)$ as $A + (1-p_1)\ldots(1-p_{i-1})(p_i d_i + (1-p_i)D)$, where $A = p_1d_1 + \ldots + (1-p_1)\ldots (1-p_{i-2})p_{i-1}d_{i-1}$ and $D = p_{i+1}d_{i+1} + \ldots + (1-p_{i+1})\ldots(1-p_m)d_{m+1}$. Then, $D \geq d_i$, since $d_i \geq d_i$ for every $l \geq i + 1$ and $p_{i+1} + \ldots + (1-p_{i+1})(1-p_m) = 1$. Consequently, if we increase $p_i$, then $E(.)$ decreases.

We prove (iii) by writing $E(.) = A^2 + B + D$ and showing that $A \leq 0$. Clearly, $E(.)$ is quadratic in $\gamma$, since each term of $E(.)$ is a polynomial function of degree at most 2. Those terms that contribute to the coefficient $A$ are last $m+2-i$ terms from $(1-p_1)\ldots(1-p_i)\ldots(1-p_{i-1})p_i d_i$ to $(1-p_1)\ldots(1-p_m)d_{m+1}$.

So

$$A = (1-p_1)\ldots(1-p_{i-1})(1-p_{i+1})\ldots(1-p_{i-1})(d_i - D),$$

where $D = p_{i+1}d_{i+1} + \ldots + (1-p_{i+1})\ldots(1-p_m)d_{m+1}$. Again, since $d_i \geq d_i$ for every $l \geq i + 1$, we have $D \geq d_i$ and hence $A \leq 0$.

We can now bound $\gamma^{\text{loc}}_{j}(c)$ and thus bound the service cost incurred.

**Lemma 5.6.** Consider any client $j$. At any point in the local rounding step, the quantity $\gamma^{\text{loc}}_{j}(c)$ is bounded by $\tilde{C}_j(1 + e^{-1/(1-\gamma)}(\frac{2}{\gamma} - 1)) + e^{-1/(1-\gamma)} \cdot r_j \alpha_j$.

**Proof.** Since $\gamma^{\text{loc}}_{j}(c)$ does not increase in the local rounding step, as argued earlier, it suffices to bound this quantity at the beginning of the local rounding step. Define $D_j(c)$ as the $z_j(c)$-weighted average distance from $j$ to the facilities in $P(j^{(c)})$, namely, $D_j(c) = \sum_{i \in P(j^{(c)})} z_{ij}(c)c_{ij}$. Clearly $D_j(c) = \sum_{i \in F_j} c_{ij}x_{ij} = \tilde{C}_j$. We will show that for each copy $c$,

$$S^{\text{loc}}_{j}(c) \leq 1 - e^{-1/(1-\gamma)} D_j(c) + e^{-1/(1-\gamma)}(\alpha_j + 2C_j(c)/\gamma). \quad (6)$$

Adding Eq. (6) over all copies $c = 1, \ldots, r_j$ and using the fact that $\sum_{c=1}^{r_j} C_j(c) \leq \tilde{C}_j$ (Lemma 5.4), we prove the lemma.

So, fix copy $c$. Consider first the case when $j^{(c)}$ is not a cluster center. The expression for $S^{\text{loc}}_{j}(c)$ is given by Eq. (3). Recall that $P(j^{(c)}) = \{i_1, \ldots, i_m\}$, and $\tilde{z}_{ij}(c) = z_{ij}(c)/(1-\gamma)$ and $d_i = c_{ij}$. Then, $\sum_{i \leq m} \tilde{z}_{ij}(c) = D_j(c)$. 

ACM Transactions on Algorithms, Vol. 4, No. 4, Article 51, Publication date: August 2008.
Let \( p = \sum_{l \leq m} (1 - \hat{z}_l) \leq e^{-\sum_{l \leq m} \hat{z}_l} = e^{-1/(1-\gamma)} \), since \( l \leq m \hat{z}_l = 1 \). In the last bracketed term in Eq. (3), each distance \( c_l = c_{bj} \) for \( l = 1, \ldots, q \) is at most \( \alpha_j + 2C_j(c)/\gamma \) by Lemma 5.3, so since \( \hat{y}_{b_1} + \cdots + (1 - \hat{y}_{b_q}) \cdots (1 - \hat{y}_{e_{m-1}}) = 1 \), we can upper-bound the coefficient of \( (1 - \hat{z}_l) \cdots (1 - \hat{z}_{lm}) \) by \( \alpha_j + 2C_j(c)/\gamma \). Substituting this upper bound, we get an expression that has the form of \( E(.) \) in Lemma 5.5. So, by part (i) of the lemma, we can bound \( S_j^{\text{loc}}(c) \) by \( (1-p)D_j(c) + p \cdot (\alpha_j + 2C_j(c))/\gamma \). This is at most the bound in Eq. (6), since \( p \leq e^{-1/(1-\gamma)} \) and \( D_j(c) \leq \alpha_j \).

If \( j(c) \) is the center of some cluster \( M \), the expression for \( S_j^{\text{loc}}(c) \) is given by Eq. (2) where \( M = \{i_1, \ldots, i_m\} \subseteq P(j(c)) \). Let \( \{i_{m+1}, \ldots, i_q\} \) be the facilities in \( P(j(c)) \setminus M \) ordered by increasing \( c_{ij} \). Note that for every \( l \geq m + 1 \), since \( i_l \) was part of \( N_j \) when cluster \( M \) was created, it must be that \( d_l \geq d_m \). We compare \( S_j^{\text{loc}}(c) \) to the function

\[
\begin{align*}
    f(x) &= \hat{y}_i d_1 + \cdots + (1 - \hat{y}_i) \cdots (1 - \hat{y}_{m-1}) d_m + (1 - \hat{y}_i) \cdots (1 - \hat{y}_{m-1}) \\
    &\quad (1 - x) \hat{y}_{m+1} d_{m+1} + \cdots + (1 - \hat{y}_{m+1}) \cdots (1 - \hat{y}_{i_l}) \alpha_j.
\end{align*}
\]

Let \( p = \sum_{l \leq q} (1 - \hat{y}_i) \). We have \( S_j^{\text{loc}}(c) = f(1) \leq f(\hat{y}_{m+1}) \leq (1-p)D_j(c) + p \cdot \alpha_j \), which is at most the bound in Eq. (6). The first inequality is due to the nondecreasing property from part (ii) of Lemma 5.5, and the second inequality is from part (i) of the same lemma. \( \square \)

**LEMMA 5.7.** For any copy \( j(c) \), at the beginning of the global rounding step, we have \( S_j^{f\text{lb}}(c) \leq S_j^{f\text{loc}}(c) \).

**PROOF.** This is a simple corollary of part (ii) of Lemma 5.5. The only case in which \( S_j^{f\text{lb}}(c) \) differs from \( S_j^{f\text{loc}}(c) \) is when the expression for \( S_j^{f\text{loc}}(c) \) contains the term \( \hat{x}_{ij} \). In this case, we obtain \( S_j^{f\text{lb}}(c) \) from \( S_j^{f\text{loc}}(c) \) by substituting \( \hat{x}_{ij} \) with \( \hat{y}_i \geq \hat{x}_{ij} \). \( \square \)

Concavity of the function \( h_{1,i}(.) \) defined by Eq. (5) now follows, since: (1) Each \( S_j^{f\text{lb}}(c) \) is of the same form as \( E(.) \) in Lemma 5.5; (2) the contribution of \( S_j^{f\text{lb}}(c) \) to \( h_{1,i}(.) \) is \( S_j^{f\text{lb}}(c)(\hat{y}_i + \cdots, \hat{y}_i - \cdots) \) and this function of \( \hat{y}_i \) is either constant, if neither \( \hat{y}_i \) nor \( \hat{y}_i \) appear in the expression for \( S_j^{f\text{lb}}(c) \) (see Eq. (3)), or linear (if exactly one of them appears), or concave in the range of interest (if both appear as shown by part (iii) of Lemma 5.5); and (3) the remaining part of \( \text{cost}(.) \) is \( T(\hat{y}_1, \ldots, \hat{y}_n) \), which is a linear function and therefore the part \( h_{1,i}(.) \) corresponding to \( T(\hat{y}_1, \ldots, \hat{y}_n) \) is a linear function of \( \hat{y}_i \). We can finally prove the main theorem.

**THEOREM 5.8.** The aforesaid algorithm delivers a solution of expected cost of at most \( \max\{\frac{1}{\gamma}, \frac{1}{1-\gamma} + e^{-1/(1-\gamma)}, 1 + \frac{2}{\gamma} \cdot e^{-1/(1-\gamma)} \} \cdot \text{OPT} \). Taking \( \gamma = 0.4819 \), we get a solution of cost at most \( 2.0753 \cdot \text{OPT} \).

**PROOF.** First, observe that we return a feasible solution. Copies connected in phases 1 and 2 are connected to distinct facilities and, by Corollary 5.2, no such facility is reused in phase 3. In phase 3, each copy has a distinct backup cluster as well as a disjoint preferred set after the local rounding step, and each is connected only to a facility opened from one of these two sets. So, copies in phase 3 are also assigned to distinct facilities.
Let define \( \text{total cost is bounded by } \max(1) \) of this step. The first term is \( \sum_{i:y_i < y} f_i y_i / (1 − \gamma) \). The second term is bounded by \( \sum_{j} S_j^\text{opt}(c) \) (Lemma 5.7), which in turn is bounded by
\[
\hat{C}_j = 1 + e^{-1/(1−\gamma)} \cdot \frac{2}{\gamma} - 1 + e^{-1/(1−\gamma)} \cdot r_j \alpha_j.
\]
(by Lemma 5.6)

We know that \( i:x_j < y \geq r_j \) for every client \( j \), so using complementary slackness we get that \( j r_j \alpha_j \) is at most \( i:y_i < y \sum_{j} f_i y_i + \sum_{j} i:x_j < y c_j x_j \). So the total cost of phase 3 is at most
\[
\frac{1}{1−\gamma} \cdot \sum_{i:y_i < y} f_i y_i + 1 + e^{-1/(1−\gamma)} \cdot \frac{2}{\gamma} - 1 + \sum_{j} i:x_j < y e_j x_j + e^{-1/(1−\gamma)} \cdot \sum_{j} i:x_j < y r_j \alpha_j
\]
\[\leq \frac{1}{1−\gamma} + e^{-1/(1−\gamma)} \cdot \sum_{i:y_i < y} f_i y_i + 1 + \frac{2}{\gamma} \cdot e^{-1/(1−\gamma)} \cdot \sum_{j} i:x_j < y c_j x_j, \quad (7)\]
The total cost incurred in phases 1 and 2 is at most \( \frac{1}{\gamma} \cdot \left( \sum_{i:y_i < y} f_i y_i + \sum_{j} i:x_j < y c_j x_j \right) \), since each facility opened has \( y_i \geq \gamma \) and if a copy of \( j \) is connected to facility \( i \) then \( x_j \geq \gamma \). Combining this with Eq. (7), we see that the total cost is bounded by \( \max(\frac{1}{\gamma} + \frac{1}{1−\gamma} + e^{-1/(1−\gamma)}, 1 + \frac{2}{\gamma} \cdot e^{-1/(1−\gamma)}) \cdot OPT \).

6. A 1.52-Approximation Algorithm for Uniform Requirements

We now show that a modification of the algorithm given by Jain et al. [2002] gives an algorithm for the uniform requirement fault-tolerant case with the same approximation ratio. Combined with a greedy improvement step and the analysis of Mahdian et al. [2006], this gives a 1.52-approximation algorithm.

6.1. THE ALGORITHM. The algorithm is based on the primal-dual method and analyzed using the dual fitting approach (see, e.g., Vazirani [2001, Chapter 13]). We will simultaneously construct both a primal and a dual solution such that the cost of the primal is exactly paid for by the dual variables. However, the dual solution may be infeasible. We will bound the infeasibility by a factor \( c \), which becomes the approximation ratio of the algorithm.

Let \( r_j = r \) be the requirement of each demand point. There is a notion of time \( t \). We say that \( j \) is active at time \( t \) if not all of its copies have been connected, and inactive otherwise. If \( j \) is active we define the active copy of \( j \); non \( a_j \) initialized to 1, to be the first copy that is not yet connected. Each \( j \) has \( r \) dual variables associated with it: \( \alpha_j^{(1)}, \ldots, \alpha_j^{(r)} \). Initially, \( t = 0 \) and all dual variables are 0. All demands \( j \) are active, and all facilities are closed. As time increases, we raise the dual variable \( \alpha_j^{(c)} \) for the active copy \( c \) and open some facilities. Once copy \( c \) gets connected, we stop raising \( \alpha_j^{(c)} \), so if \( j \) is inactive none of its variables are raised.

Let \( i(j^{(c)}) \) denote the facility to which copy \( j^{(c)} \) is connected. If \( j \) is inactive, we define \( l_j \) to be \( \max \) (i.e., the distance between \( j \) and \( i(j^{(c)}) \)). At any time \( t \), the contribution of \( j \) to a closed facility \( i \) is \( \max(\alpha_j^{(a_i)} − c_{ij}, 0) \) if \( j \) is active, and \( \max(l_j − c_{ij}, 0) \) otherwise.
We raise the variables \( \alpha_j^{(a_j)} \) of all active demands at unit rate until one of the following events happens.

1. The total contribution from all demands \( j \) to some closed facility \( i \) becomes equal to \( f_i \): We open \( i \). For each \( j \) with a positive contribution to \( i \), we assign a copy of \( j \) to \( i \). After this, \( j \) cannot take back its contribution to \( i \). If \( j \) is active, connect \( j^{(a_j)} \) to \( i \); if \( a_j = r \), \( j \) becomes inactive, otherwise set \( \alpha_j^{(a_j+1)} = \alpha_j^{(a_j)} \) and \( a_j = a_j + 1 \). Note that the contribution of \( j \) to other closed facilities remains the same. If \( j \) is inactive, disconnect the copy \( c \) for which \( l_j = c \) and connect it to \( i \). Clearly \( l_j \) does not increase.

2. An active \( j \) reaches an open facility \( i \) (i.e., \( \alpha_j^{(a_j)} = c_j \)) and no copy of \( j \) is already connected to \( i \): Connect \( j^{(a_j)} \) to \( i \). If \( a_j = r \), \( j \) becomes inactive, otherwise set \( \alpha_j^{(a_j+1)} = \alpha_j^{(a_j)} \) and \( a_j = a_j + 1 \).

We now raise \( \alpha_j^{(a_j)} \) of only the active demands and continue until all demands become inactive.

6.2. ANALYSIS. It is clear that the cost of the primal solution is \( j \sum_{i \in \mathcal{F}} \alpha_j^{(i)} \). We first define a dual solution \( (\alpha, \beta, z) \) using the \( \alpha_j^{(i)} \) variables. Let \( \alpha_j = \alpha_j^{(r)} = \max_k \alpha_j^{(r)} \). Define \( \theta_j = \alpha_j - \alpha_j^{(r)} \) if \( j^{(r)} \) is connected to \( i \) and \( \alpha_j \) was active when it got connected to \( i \), and 0 otherwise. In the former case we say that \( j^{(r)} \) is primarily connected to \( i \). Let \( z_i = j \theta_j \). Note that \( j r \alpha_j - i z_i = j \th_j^{(i)} \), so the dual variables exactly pay for the cost of the primal. We will show that for each \( i \),

\[
\sum_{j \in \mathcal{F}} \alpha_j - z_i \leq \gamma (\sum_{j \in \mathcal{F}} c_{ij} + f_i) \tag{1}
\]

Then we can define \( \beta_j = \alpha_j - \gamma c_{ij} \) so that the solution \( (\alpha, \beta, z) \) would be infeasible by a factor of at most \( \gamma \), implying an approximation ratio of \( \gamma \). If we further show that \( j \in \mathcal{K} \alpha_j - z_i \leq \gamma \gamma c_{ij} + \gamma f_i \) for any \( i \) and any set of clients \( S \), then looking at each \( i \) opened (fractionally) in a solution and the set of clients \( \{ j : x_{ij} > 0 \} \), and we add the corresponding inequalities weighted by \( y_i \), we get that \( j r \alpha_j - i z_i \leq \gamma F_y + \gamma C_y \). Here \( F_y \) and \( C_y \) denote, respectively, the facility and connection cost of a fractional LP solution. (We need to be a little careful here, since we may have \( 0 < y_i < y_i \) for some \( i, j \). But we can decompose the star consisting of facility \( i \) and clients \( \{ j : x_{ij} > 0 \} \) into a collection of stars \( \{ T = (i, D_T) \} \), and weigh the inequality corresponding to star \( T \) by \( y_T \) in such a way that \( T y_T = y_i \) and for each client \( j \), \( T : j \in D_T, y_T = x_{ij} \).

Consider a star consisting of facility \( i \) and a set of \( k \) clients \( S = \{ j_1, \ldots, j_k \} \). It suffices to bound the ratio of \( j \in \mathcal{F} (\alpha_j - \theta_j) \) and \( (f_i + j \in \mathcal{F} c_{ij}) \). Let \( f = f_i, d_j = c_{ij}, \) and \( v_j = \alpha_j - \theta_j \). Note that \( v_j = \alpha_j^{(i)} \) if \( j^{(i)} \) is primarily connected to \( f \), and \( \alpha_j^{(r)} \) otherwise. We number the demands \( 1 \ldots k \) so that \( v_1 \leq \ldots \leq v_k \). Consider the time \( t = v_1 - \). At this time each demand \( j < i \) is either inactive or a copy \( j^{(r)} \) is primarily connected to \( f \). Let \( r_{j,i} = v_j \) if \( j^{(r)} \) is primarily connected to \( f \), and \( l_j \) (at the time \( v_i - \)) otherwise. Note that \( r_{j,i+1} \geq \ldots \geq r_{j,k} \).

We now write some inequalities involving the variables \( v_j, d_j, r_{j,i} \). Again consider time \( t = v_i - \). At this time the contribution of \( j \) to \( i \) is \( \max(r_{j,i} - d_j, 0) \) if \( j \) and \( \max(t - d_j, 0) \) otherwise. Since the total contribution to a facility never
exceeds its facility cost, we have
\[
\max_{j<i}(r_{j,i} - d_j, 0) + \max_{j\geq i}(v_i - d_j, 0) \leq f \quad \text{for all } i \text{'s.} \tag{8}
\]

Now we use the triangle inequality. If \( f \) is open at time \( t = v_i - d_i \), then \( v_i \leq d_i \).
Otherwise, every demand \( j < i \) is inactive at time \( t \) (since \( j^{(c)} \) is not primarily connected to \( f \), as \( f \) is not open), so \( j \) is already connected to \( r \) facilities. By the definition of \( v_i \) and \( t \), we have that \( i \) is still active at time \( t \). Thus, \( i \) is connected to less than \( r \) facilities at time \( t \), implying that there is some facility to which \( j \) is connected and but to which \( i \) is not yet connected. (This is where we use the fact that the requirements of all clients are equal.) The distance between this facility and \( i \) is upper bounded by \( r_{j,i} + d_i + d_j \) and this must be at least \( t \); otherwise, the algorithm would have connected \( i \) to this facility at a time earlier than \( t \). So, we have
\[
v_i \leq r_{j,i} + d_i + d_j \quad \text{for all } i, j < i. \tag{9}
\]

Using the preceding inequalities we can write a mathematical program to bound the ratio of
\[
\frac{\sum_{j \in S} (\alpha_j - \theta_{ij})}{\sum_{j \in S} c_{ij}}.
\]
The variables \( v_j, d_j, f, r_{j,i} \), obtained by running the algorithm, form a feasible solution to the optimization problem we give next, which can be written as a linear program (LP). Hence, the ratio is bounded by the LP optimum value.

\[
\gamma_k = \max \quad \frac{\sum_{j=k}^1 v_j}{\sum_{j=k}^1 d_j} \quad \text{(LP)}
\]

such that
\[
\begin{align*}
& v_1 \leq \cdots \leq v_k \\
& r_{j,j+1} \geq \cdots \geq r_{j,k} \quad \text{for all } j \\
& v_1 \leq r_{j,i} + d_i + d_j \quad \text{for all } i, j < i \\
& \max_{j<i}(r_{j,i} - d_j, 0) + \max_{j\geq i}(v_i - d_j, 0) \leq f \quad \text{for all } i \\
& v_j, d_j, f, r_{j,i} \geq 0
\end{align*}
\]

This is the same as the so-called factor-revealing LP in Jain et al. [2002], so all the results in their algorithm hold for this one, as well. In particular, we have the following lemma and theorem (Lemma 13 and Theorem 4 in Jain et al. [2002]).

**Lemma 6.1.** \( \gamma_k \leq 1.61 \) for all \( k \).

**Theorem 6.2.** The previous algorithm is a 1.61-approximation algorithm for fault-tolerant facility location with uniform requirements.

We say that an algorithm is a \((\gamma_f, \gamma_c)\)-approximation algorithm if it returns a solution of cost at most \( \gamma_f F^* + \gamma_c C^* \), where \( F^* \) and \( C^* \) denote, respectively, the facility and connection cost of any (fractional) solution. Note that there could be more than one \((\gamma_f, \gamma_c)\) pair for which the algorithm is a \((\gamma_f, \gamma_c)\)-approximation algorithm. Theorem 9 in Jain et al. [2002] and Lemma 2 in Mahdian et al. [2006] establish the next theorem.
THEOREM 6.3. Let $\gamma_f \geq 1$. Define $\gamma_f$ as $\max_{j \leq k} \frac{\gamma_j - \gamma_f}{d_j}$ subject to the same set of constraints as (LP), and $\gamma_c = \sup_k \{\gamma_k\}$. The preceding algorithm is a $(\gamma_f, \gamma_c)$-approximation algorithm. In particular, for $\gamma_f = 1$, $\gamma_c \leq 2$ and for $\gamma_f = 1.11$, $\gamma_c \leq 1.78$, so the aforesaid algorithm is a $(1,2)$- and a $(1.11,1.78)$-approximation algorithm, respectively.

6.3. SCALING AND GREEDY AUGMENTATION. It is possible to improve the performance of the previous algorithm by using scaling and greedy augmentation [Guha and Khuller 1999; Charikar and Guha 2005]. The combined algorithm is as follows.

**Algorithm. FTUFL($\delta$)**

1. Scale the facility costs by $\delta$, namely, set $f_i \leftarrow \delta f_i$.
2. Run the aforementioned primal-dual algorithm, called algorithm $A$, on the scaled instance.
3. Scale back the facility costs and perform greedy augmentation. Define the gain of a facility $i$, $\text{gain}_i$, to be the reduction in total cost obtained by adding facility $i$ to the current solution (if the total cost does not decrease, then $\text{gain}_i = 0$). While there exists a facility with positive gain, choose the facility $i$ for which $\text{gain}_i$ is maximized and add it to the current solution.

The next lemma was proved in Guha and Khuller [1999] and Charikar and Guha [2005], and in the context of fault-tolerant facility location in Guha et al. [2003].

**Lemma 6.4.** Let $F^*$ and $C^*$ be the facility and connection, respectively, of a (possibly fractional) solution to the fault-tolerant facility location problem. Greedy augmentation, when applied to a solution with initial facility cost $F$ and service cost $C$, produces a solution of cost at most $F + F^* \max\{0, \ln\left(\frac{C-C^*}{F^*}\right)\} + F^* + C^*$.

**Lemma 6.5 [Charikar and Guha 2005; Mahdian et al. 2006].** Let $A$ be a $(\gamma_f, \gamma_c)$-approximation algorithm. The aforesaid procedure with parameter $\delta \geq 1$, gives a $(\gamma_f + \ln \delta, 1 + \frac{\gamma_c-1}{\delta})$-approximation.

Taking $\delta = 1.504$ in algorithm FTUFL and plugging in $(\gamma_f, \gamma_c) = (1.11, 1.78)$, we get a 1.52-approximation.

**Theorem 6.6.** There is a 1.52-approximation algorithm for fault-tolerant facility location with uniform requirements.

7. The Fault-Tolerant k-Median Problem

We now consider the metric fault-tolerant $k$-median problem. In this variant, we have the additional constraint that we may open at most $k$ facilities. As in the fault-tolerant facility location version, the goal is to connect each demand $j$ to $r_j$ distinct open facilities and minimize the total cost of opening facilities and assigning clients to them. We can write an LP for this problem that is very similar to the LP for fault-tolerant UFL. The primal program (FTFL-P) has the additional constraint that $\sum_{i \leq k} y_i \leq k$. This modifies the objective function of the dual (FTFL-D) to
max \( j r_j\alpha_j - i z_i - k \), and constraint (1) changes to \( j \beta_j \leq f_i + z_i + 1 \). Let (KP) and (KD), respectively, denote the primal and dual programs for fault-tolerant \( k \)-median, and let \( OPT_k \) be the value of an optimal LP solution. In this section we will use the primal-dual algorithm of Section 6.1 to give a 4-approximation algorithm for the uniform requirement case \( r_j = r \). We assume \( r \leq k \); otherwise there is no feasible solution.

Given an instance of fault-tolerant UFL with facility costs \( f_i \), suppose we set the facility costs to \( 2f_i \) and run the primal-dual algorithm to get a primal solution of cost \( 2F, C \) with (unscaled) facility cost \( F \), connection cost \( C \), and a possibly infeasible setting of the dual variables \( \alpha_j, z_i \). We have that \( 2F + C = j \alpha_j - i z_i \), and by Theorem 6.3 we know that for \( \gamma_f = 1, \gamma_r \leq 2 \), so for any facility \( i \) and set of demands \( S \), \( j \alpha_j - 2f_i - z_i \leq 2 \sum_{j \in S} c_{ij} \). So if we set \((\alpha, z) = (\alpha/2, z/2)\) and \( \beta_j = \alpha_j - c_{ij} \), then \((\alpha, \beta, z)\) is a feasible dual solution and \( 2F + C \leq 2( j \alpha_j - i z_i ) \). In the sequel whenever we say that “we run the primal-dual algorithm,” we mean running this modified algorithm where we first scale the facility costs by a factor of 2 and then running the original primal-dual algorithm.

Also, when we state that the algorithm returns a primal solution of cost \((\hat{F}, \hat{C})\) and dual solution \((\hat{\alpha}, \hat{\beta}, \hat{z})\), \( \hat{F} \) is the original unscaled facility cost of the primal solution, and \((\hat{\alpha}, \hat{\beta}, \hat{z})\) is a feasible dual solution obtained as before so that \( 2\hat{F} + \hat{C} \leq 2( j \hat{\alpha} - i \hat{z} ) \).

Consider fixing and running the previous algorithm with the facility costs modified to \( f_i + 1 \) (i.e., we first scale \((f_i + 1)\) by 2 and then run the original primal-dual algorithm). Suppose the algorithm returns a primal solution \((\hat{F}, \hat{C})\) in which exactly \( k \) facilities are opened, as well as a dual solution \((\alpha, \beta, z)\). So the primal solution is a feasible solution to (KP) and \((\alpha, \beta, z, \Delta)\) is a feasible solution to (KD). Also, \( 2(F + k) + C \leq 2( j \alpha_j - i z_i ) \implies F + C \leq 2( j \alpha_j - i z_i - k ) \leq 2OPT_k \), so we have a solution of cost at most \( 2OPT_k \).

The basic idea now is to “guess” the right value of \( k \) so that when the facility costs are modified to \( f_i + 1 \), the algorithm ends up opening \( k \) facilities. This idea was first used by Jain and Vazirani [2001] for the (nonfault-tolerant, i.e., \( r_j = 1 \)) \( k \)-median problem. If at \( \gamma = 0 \) the algorithm opens at most \( k \) facilities, then, by the same reasoning as earlier, we have a feasible solution of cost at most \( 2OPT_k \).

So assume that we open more than \( k \) facilities at \( \gamma = 0 \). When is very large, say, \( nr \max_j c_{ij} \), the algorithm will open just \( r \) facilities and connect all demands to these facilities. We can show that there is a value \( \gamma = 0 \) such that, depending on how we break ties between events in the primal-dual algorithm, we get two primal solutions, one opening \( k_1 < k \) facilities and the other opening \( k_1 > k \) facilities, and a single dual solution. The two primal solutions can be found in polynomial time by performing a bisection search in the interval \([0, nr \max_j c_{ij}]\) and terminating the search when the length of search interval becomes less than \( 2^{L - \text{poly}(n + L)} \), where \( L = \log(\max_j c_{ij}) \). The proof is very similar to that in the conference version of Jain and Vazirani [2001].

Let \((\alpha, \beta, z, \Delta, 0)\) be the common dual solution for the two primal solutions. The dual solution \((\alpha, \beta, z, \Delta, 0)\) is used only in the analysis and not in the algorithm. Let \((x_1, y_1)\) and \((x_2, y_2)\) be the two solutions opening \( k_1 < k \) and \( k_2 > k \) facilities with costs \((F_1, C_1)\) and \((F_2, C_2)\), respectively. A convex combination of these two yields a fractional solution that is feasible in (KP) and opens exactly \( k \) facilities. Let \( a, b \)
be such that $ak_1 + bk_2 = k$, $a + b = 1$. Then,

$$2(aF_1 + bF_2) + (aC_1 + bC_2) \leq 2 \sum_{j} \alpha_j - \sum_{i} z_i - k \leq 2 \cdot \text{OPT}_k. \quad (10)$$

We will round this solution, losing a factor of 2. If $a \geq \frac{1}{2}$, we take the solution $(x_1, y_1)$, which is feasible and from Eq. (10) and get that $F_1 + C_1 \leq 4 \cdot \text{OPT}_k$. Otherwise, we open a subset of the facilities opened in $y_2$. We call a facility opened in $y_1$ a “small” facility and opened in $y_2$ a “large” facility; a facility opened in both is both small and large. We match each small facility with a large facility as follows: A small facility that is also large is matched with itself. We consider the other small facilities in arbitrary order, and pair each small facility with the unpaired large facility closest to it. Note that exactly $k_1$ large facilities are matched this way.

With probability $a$, we open all the small facilities, and with probability $b = 1 - a$ we open all the matched large facilities. We also select a random subset of $k - k_1$ unmatched large facilities and open all of these. Each client $j$ is simply connected to the $r_j = r$ open facilities closest to it. Each large facility is opened with probability $b = \frac{k - k_1}{k_1}$, therefore the total facility opening cost is at most $aF_1 + bF_2$.

**Lemma 7.1.** The total facility opening cost incurred in at most $aF_1 + bF_2$.

**Lemma 7.2.** The expected connection cost of a demand $j$ is at most $2 \sum_{i} c_{ij} (ax_{1,j} + bx_{2,j})$.

**Proof.** We will prove the claimed bound by considering a suboptimal way of assigning $j$ to $r$ open facilities (instead of connecting $j$ to the $r$ nearest open facilities), and bounding the connection cost of $j$ under this suboptimal assignment.

Let $S_j$ be the set of small facilities to which $j$ is connected, namely, $S_j = \{i : x_{1,j} = 1\}$. Similarly, let $L_j$ be the set of large facilities that serve $j$. Clearly $|S_j| = |L_j| = r$.

For each copy $j^{(c)}$, we define a set of facilities $T_c$, and $j^{(c)}$ will only be connected to a facility in $T_c$. First, we arbitrarily assign each facility $i \in S_j$, and the large facility $i$ to which it is matched (which could be the same as $i$), to a distinct set $T_c$. Observe the important fact that the sets $T_c$ are disjoint, since distinct facilities in $S_j$ are matched to distinct large facilities. Let $m(S_j)$ denote the set of large facilities that are matched to facilities in $S_j$. Then $|m(S_j)| = |S_j| = |L_j| \Rightarrow |m(S_j) \setminus L_j| = |L_j \setminus m(S_j)|$, so the number of sets $T_c$ not containing a facility from $L_j$ after the first step is equal to the number of unmatched facilities in $L_j$. We assign a distinct unmatched facility of $L_j$ to each set $T_c$ which does not already contain a facility from $L_j$. Note that the sets $T_c$ remain disjoint; so if we connect each copy $j^{(c)}$ to a facility in $T_c$, we will get a feasible solution.

For convenience, if facility $i \in S_j$ is matched with $i$, we will consider $i$ and $i$ as two different facilities even if $i = i$. Let $X$ be the service cost of $j$ and $X^{(c)}$ be the service cost of $j^{(c)}$. Fix copy $c$. The set $T_c$ contains at least one small facility $i_1 \in S_j$ and one large facility $i_2$ such that $i_1$ is matched to $i_2$. If these are the only two facilities then it must be that $i_2 \in L_j$. Exactly one of $i_1$ and $i_2$ is open; we assign $j^{(c)}$ to $i_1$ or $i_2$, whichever is open. So $E[X^{(c)}] = ac_{i_1,j} + bc_{i_2,j}$. Otherwise, $T_c$ contains a third facility $i_3 \in L_j$ such that $i_3$ is unmatched and $i_2 \notin L_j$. We assign $j^{(c)}$ to $i_3$ if it is open, otherwise to $i_1$ or $i_2$, whichever is open. So $E[X^{(c)}] = bc_{i_3,j} + a(ac_{i_1,j} + bc_{i_2,j})$. 

ACM Transactions on Algorithms, Vol. 4, No. 4, Article 51, Publication date: August 2008.
Since \( i_1 \) is matched with \( i_2 \) and \( i_3 \) is unmatched, it must be that \( i_2 \) is closer to \( i_1 \) than to \( i_3 \). So, \( c_{i_2j} \leq c_{i_1j} + c_{i_2i_3} \leq c_{i_1j} + c_{i_1i_3} \leq 2c_{i_1j} + c_{i_3j} \). Therefore

\[
E[X^{(c)}] \leq bc_{i_3j} + a(ac_{i_1j} + 2bc_{i_1j} + bc_{i_3j}) = a(1 + b)c_{i_1j} + b(1 + a)c_{i_3j} \\
\leq 2(ac_{i_1j} + bc_{i_3j}).
\]

Thus for every copy \( c \), if \( i, i \in T_c \), where \( i \in S_j \) and \( i \in L_j \), we have that \( E[X^{(c)}] \leq 2(ac_{ij} + bc_{ij}) = 2(ac_{ij}x_{1,ij} + bc_{ij}x_{2,ij}) \), since \( x_{1,ij} = x_{2,ij} = 1 \). So, summing over all copies \( c \), since the set of all facilities \( i \) is precisely \( S_j \) and the set of all facilities \( i \) is the set \( L_j \), we get \( E[X] \leq 2( \sum_{i \in S_j} ac_{ij}x_{1,ij} + \sum_{i \in L_j} bc_{ij}x_{2,ij}) = 2 \sum ac_{ij}(ax_{1,ij} + bx_{2,ij}), \) where the last equality follows since if \( i \notin S_j \), then \( x_{1,ij} = 0 \), and if \( i \notin L_j \), then \( x_{2,ij} = 0 \). \( \square \)

**Theorem 7.3.** The aforesaid algorithm returns a solution of expected cost at most four times the optimum for the fault-tolerant \( k \)-median problem with uniform requirements.

**Proof.** By Lemma 7.2, the expected service cost of client \( j \) is at most \( 2 \sum ac_{ij}(ax_{1,ij} + bx_{2,ij}) \). Also we have \( C_1 = \sum_{i \in S_j} c_{ij}x_{1,ij} \) and \( C_2 = \sum_{i \in L_j} c_{ij}x_{2,ij} \). So summing over all \( j \), we see that the expected total service cost is at most \( 2(aC_1 + bC_2) \). Combining this observation with Lemma 7.1, the expected total cost of the solution returned is at most \( (aF_1 + bF_2) + 2(aC_1 + bC_2) \leq 4 \cdot OPT_k \), from Eq. (10). \( \square \)

**Acknowledgments.** We thank the anonymous referees for their useful suggestions.

**References**


Fault-Tolerant Facility Location


RECEIVED AUGUST 2005; REVISED SEPTEMBER 2007; ACCEPTED OCTOBER 2007