Fault-Tolerant Facility Location*

Chaitanya Swamy† David B. Shmoys‡

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 the 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 the 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.

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 [18]). In its simplest version, the uncapacitated facility location (UFL) problem, we are given a set of facilities $\mathcal{F}$ and a set of clients $\mathcal{D}$. 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 $\mathcal{F}$ and assign the clients to open facilities so as to minimize the sum of the facility opening costs and the 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, and 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 demonstrating this, consider a setting where facilities (which could represent caches) fail

*A preliminary version [21] appeared in the Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003.

†cswamy@math.uwaterloo.ca. Dept. of Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1. Work done while the author was a student at the Department of Computer Science, Cornell University, Ithaca, NY 14853. Research partially supported by NSF grant CCR-9912422.

‡shmoys@cs.cornell.edu. Dept. of Computer Science, Cornell University, Ithaca, NY 14853. Research partially supported by NSF grants CCR-9912422, CCF-0430682 and DMI-0500263.

independently with probability p, and each client j must be guaranteed that it will 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 $\mathcal{F}$ and a set of clients $\mathcal{D}$, 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 problem is a generalization of the uncapacitated facility location problem, which is the setting where $r_j = 1$ for each client $j \in \mathcal{D}$.

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. [7]. If all requirements are equal, we give a 1.52-approximation algorithm by building upon the algorithm of [10, 16], 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 [15]. 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, Tardos and Aardal [19] who gave a 3.16-approximation algorithm, using the *filtering* technique of Lin and Vitter [14] to round the optimal solution of a linear program. After an improvement by Guha and Khuller [6], Chudak and Shmoys [5] gave an LP rounding based $\left(1+\frac{2}{e}\right)$-approximation algorithm. They used information about the structure of optimal primal and dual solutions, and combined randomized rounding and the decomposition results of [19] to get a variant that might be called *clustered randomized rounding*. Sviridenko [20] improved the ratio to 1.58. Jain and Vazirani [12] gave a combinatorial *primal-dual* 3-approximation algorithm where the LP is used only in the analysis. Mettu and Plaxton [17] 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, Plaxton and Rajaraman [13] and later improved by [3, 2]. Jain, Mahdian, Markakis, Saberi and Vazirani [9] 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, Ye and Zhang [16] to 1.52, which is the best known guarantee.

Charikar, Guha, Tardos and Shmoys [4] gave the first constant-factor algorithm for the k-median problem based on LP rounding. This was improved in a series of papers [12, 3, 9, 2] to $(3+\epsilon)$ [2].

The fault-tolerant facility location (FTFL) problem was first studied by Jain and Vazirani [11] who gave a primal-dual algorithm achieving a performance guarantee that is logarithmic in the largest requirement. Our algorithm is based on LP rounding. We consider the following LP and its dual.

$$\begin{array}{lll} \min & \sum_i f_i y_i + \sum_j \sum_i c_{ij} x_{ij} & \text{(FTFL-P)} \\ \text{s.t.} & \sum_i x_{ij} \geq r_j & \forall j \\ & x_{ij} \leq y_i & \forall i,j \\ & y_i \leq 1 & \forall i \\ & x_{ij}, y_i \geq 0 & \forall i,j. \end{array} \qquad \begin{array}{lll} \max & \sum_j r_j \alpha_j - \sum_i z_i & \text{(FTFL-D)} \\ \text{s.t.} & \alpha_j \leq \beta_{ij} + c_{ij} & \forall i,j \\ & \sum_j \beta_{ij} \leq f_i + z_i & \forall i \\ & \alpha_j, \beta_{ij}, z_i \geq 0 & \forall i,j. \end{array} \quad (1)$$

Variable y_i indicates if facility i is open, and x_{ij} indicates if client j is assigned to facility i. An integer solution to the LP corresponds exactly to a solution to our problem. Guha, Meyerson and Munagala [7] round the primal LP above using filtering and the decomposition technique of [19] to get a 3.16-approximation. They also show that a subsequent greedy local improvement post-processing 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, i.e., $r_j = r$, better results are known. Mahdian et al. [15] 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. [9] 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 [5]. Our rounding algorithm exploits the optimality properties of the fractional solution by using the *complementary slackness* conditions to bound the cost of the solution in terms of both the primal and dual optimal solutions. One difficulty in using LP duality to prove an approximation ratio, is the presence of $-\sum_i z_i$ in the dual objective function. As a result, bounding the cost in terms of $\sum_j r_j \alpha_j$ is not enough to prove an approximation guarantee. In general, 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 consequently the only known LP rounding algorithm [4] uses just the optimal primal LP solution. However for FTFL, complementary slackness allows us to, in effect, get rid of the negative z_is by a single pruning phase; since $z_i > 0 \implies y_i = 1$, we can open all such i and charge the opening cost to the LP.

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, in a way that ensures 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 client j and any set S of facilities that fractionally serve j such that the facility weight $\sum_{i \in S} x_{ij}$ is "large" (i.e., 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 [5] 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 a 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 will not in general be able to partition the facilities fractionally serving a client into disjoint sets with each set having a facility weight of 1. We tackle this problem by introducing another pruning phase before clustering where we open all facilities i with "large" y_i, so that in the clustering step we now only consider facilities that have "small" y_i. In the clustering step, 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 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 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 each copy a facility weight of one, but it may not be possible to simultaneously satisfy both requirements. However because of the pruning phase one can allot a substantial facility weight to each copy, due to which one 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 [1] 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 [15] to 1.52 by building upon the algorithm of Jain, Mahdian and Saberi [10]. The algorithm is analyzed using the dual fitting approach and we arrive at the same factor LP as in [10]; thus we obtain the same performance guarantees. Combined with a greedy improvement heuristic and the analysis in [16], we get a 1.52-approximation algorithm. Using a Lagrangian relaxation technique introduced in [12] 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 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 (α, β, z) be the optimal primal and dual solutions respectively and OPT be the common optimal value. The primal slackness conditions are: $x_{ij} > 0 \implies \alpha_j = \beta_{ij} + c_{ij}$, $y_i > 0 \implies \sum_j \beta_{ij} = f_i + z_i$. The dual slackness conditions are: $\alpha_j > 0 \implies \sum_i x_{ij} = r_j$, $\beta_{ij} > 0 \implies x_{ij} = y_i$, and $z_i > 0 \implies y_i = 1$. We may assume without loss of generality for each client j, $\alpha_j > 0$ so that $\sum_i x_{ij} = r_j$. Furthermore, for every client j there is at most one facility i such that $0 < x_{ij} < y_i$, and that this is the farthest facility serving j, because one 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 [5], the algorithm is based on the observation that the optimal solution is α-close, that is, $x_{ij} > 0 \implies c_{ij} \leq \alpha_j$. 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 $-\sum_i z_i$ term in the dual objective function. As a result, bounding the cost in terms of $\sum_j r_j \alpha_j$ is not enough to prove an approximation guarantee. However 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* $\sum_j n_j \alpha_j - \sum_i z_i$.

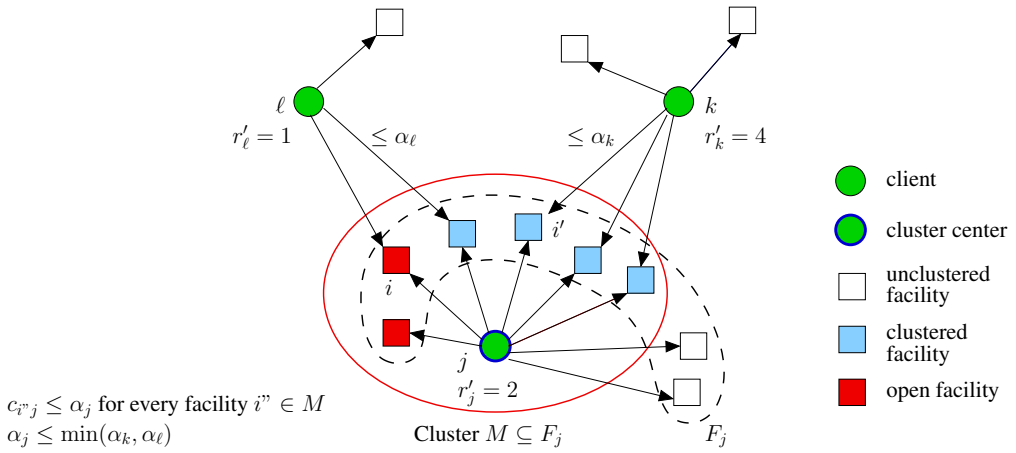


Figure 2.1: One iteration of the clustering step in Phase 2. j is the cluster center; 2 copies of j, k, and 1 copy of ℓ get connected in this iteration; j and ℓ are removed from $\mathcal{S}$ after this iteration.

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,

$$\sum_{j:x_{ij}>0} \alpha_j = \sum_{j:x_{ij}>0} (c_{ij} + \beta_{ij}) = \sum_{j:x_{ij}>0} c_{ij} + \sum_j \beta_{ij} = \sum_{j:x_{ij}>0} c_{ij} + f_i + z_i.$$

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

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 $\mathcal{S} = \{j : r'_j \geq 1\}$. We will maintain the invariant $\sum_{i \in F_j} y_i \geq r'_j$ for all $j \in \mathcal{S}$. We iteratively do the following until $\mathcal{S} = \emptyset$.

S1. Choose $j \in \mathcal{S}$ with minimum α_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 $\sum_{i' \in M} y_{i'} \geq r'_j$. If $\sum_{i' \in M} y_{i'} > r'_j$, we replace the last facility i in M (that is, i is the facility furthest 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 $x_{i_1 k}, x_{i_2 k}$ arbitrarily maintaining that $x_{i_1 k} + x_{i_2 k} = x_{ik}$, $x_{i_1 k} \leq y_{i_1}$, $x_{i_2 k} \leq y_{i_2}$. We include i_1 in M, so now $\sum_{i' \in M} y_{i'} = r'_j$.

S3. Open the r'_j cheapest facilities in M. For each client k (including j) with $F_k \cap M \neq \emptyset$, 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 2.1 shows one iteration of steps S1–S3. Step S2 is valid since we maintain the invariant $\sum_{i \in F_k} y_i \geq r'_k$ for all $k \in \mathcal{S}$. This is clearly true initially, and in any iteration, for any k with $F_k \cap M \neq \emptyset$ and which lies in $\mathcal{S}$ after the iteration, we remove a facility weight of *at most* r'_j from F_k and r'_k decreases by *exactly* r'_j.

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 $\sum_{i' \in M \setminus \{i\}} y_{i'} > r'_j - y_i > r'_j - 1$ there are at least r'_j facilities in $M \setminus \{i\}$ that are cheaper 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 $\sum_{i \in M} y_i = r'_j$ and open the r'_j cheapest facilities in M. The cost of opening the cheapest facilities is at most $r'_j \cdot$(average cost) = $\sum_{i \in M} f_i y_i$. Also the facilities in M are not reused. ■

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 non-empty (see Fig. 2.1). 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 $\mathcal{S}$). ■

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

Proof : The facility cost in Phase 2 is at most $\sum_i f_i y_i \leq OPT = \sum_j r'_j \alpha_j + \left(\sum_j n_j \alpha_j - \sum_i z_i\right)$. 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) $\leq \left(\sum_j n_j \alpha_j - \sum_i z_i\right) + \sum_i f_i y_i + 3\sum_j r'_j \alpha_j \leq 2\left(\sum_j n_j \alpha_j - \sum_i z_i\right) + 4\sum_j r'_j \alpha_j \leq 4\left(\sum_j r_j \alpha_j - \sum_i z_i\right) = 4 \cdot OPT$. ■

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 above, 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 [5], 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 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 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 in the 4-approximation algorithm above, we know precisely which facility(ies) is(are) opened within a cluster, and can therefore afford to split facilities across clusters, without sacrificing feasibility, to ensure that a cluster contains the "right" amount of facility weight. Guha et al. [7] also deterministically decide which facility to open within a cluster, possibly splitting a facility across clusters, and thereby extend the UFL algorithm of Shmoys et al. [19] relatively easily to the fault-tolerant setting.

With randomization however, 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. But 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 γ, 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 clustering step above. Clearly, we incur a loss of a factor of at most γ 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 non-cluster 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 bound the service cost for every copy separately. For each demand j, we allot each such copy $j^{(c)}$ a set of *preferred facilities* $P\big(j^{(c)}\big)$, which is a subset of the facilities that fractionally serve the unassigned copies, and 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 $P\big(j^{(c)}\big)$ (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 up the facilities among the different copies so that each copy gets a set with facility weight 1. However, Phase 2 ensures that one can allot each set $P\big(j^{(c)}\big)$ a facility weight of at least $1 - \gamma$, which gives a reasonable upper bound on the probability that no facility in $P\big(j^{(c)}\big)$ is open. Combining these various components, we already obtain an algorithm with a much better approximation ratio, 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 [1], which was applied in the context of uncapacitated facility location by Sviridenko [20]. Suppose that we distribute the facilities serving the unassigned copies of j among the preferred sets and allot each preferred set a facility weight of 1, perhaps by splitting facilities. A facility i could now lie in multiple preferred sets $P\big(j^{(c)}\big)$; let $z_{ij}(c)$ be the extent to which facility i is allotted to $P\big(j^{(c)}\big)$, so $\sum_c z_{ij}(c) = x_{ij}$. Although the preferred sets are no longer disjoint, we can still use the above 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 $P\big(j^{(c)}\big)$ 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 one should be able to show that the probability that no facility in the preferred set is available is small, and thereby bound 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, write an expression for the expected cost incurred under this randomized process, and bound this cost. More precisely, we will construct an expression $\mathsf{cost}(y_1, y_2, \ldots, y_n)$ that is a function of the y_i variables, where $n = |\mathcal{F}|$, satisfying the following properties:

P1. When the y_i values are set to the values given by the LP optimal solution, we have $\mathsf{cost}(y_1, \ldots, y_n) \leq c \cdot OPT$ 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, $\mathsf{cost}(y_1, \ldots, y_n)$ gives an upper bound on the total cost of the integer solution.

P3. $\mathsf{cost}(.)$ has some nice concavity properties (we will make this 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 $\mathsf{cost}(\tilde{y}_1, \ldots, \tilde{y}_n) \leq \mathsf{cost}(y_1, \ldots, y_n)$. So if we set the y_i values to the 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 $c \cdot OPT$, 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 $\mathsf{cost}(.)$ 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 exists, we might not know how to implement such a randomized process.*

4 Algorithm details

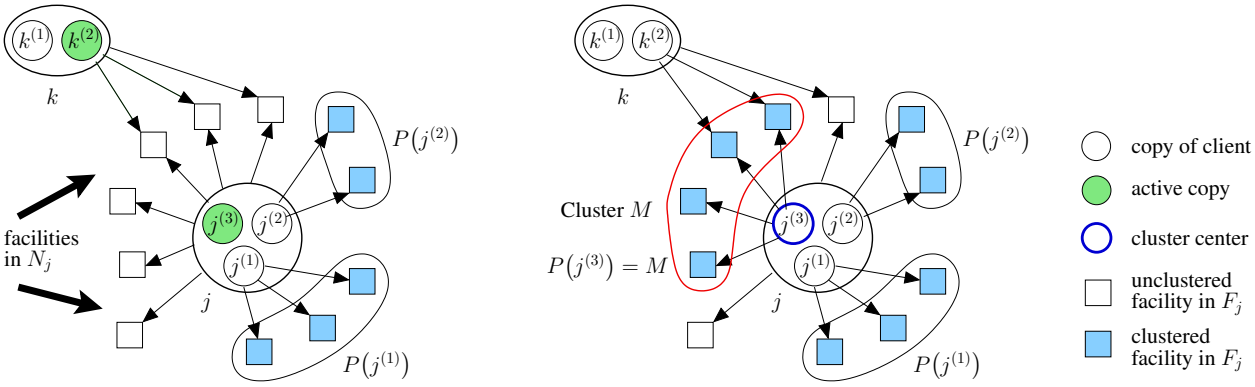
The algorithm runs in 3 phases, which are described in detail below. The entire algorithm is also summarized in Figure 4.3. Let $\gamma < \frac{1}{2}$ be a parameter whose value we will fix later.

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 $\mathcal{F} \setminus L_1$) 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 $\sum_{i:x_{ij}<\gamma} x_{ij} \geq r''_j$ since if $r''_j > 0$, then $r''_j = r'_j - |L_j|$ and $r'_j = \sum_{i:x_{ij}<1} x_{ij}$.

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 a set of facilities S to be $\mathsf{facwt}\big(S, j\big) = \sum_{i \in S} x_{ij}$. We know that $\mathsf{facwt}\big(F_j, j\big) \geq r''_j$; we assume without loss of generality that $\mathsf{facwt}\big(F_j, j\big)$ is *exactly* r''_j. (If not, we may simply take a subset of F_j starting from the first facility and proceeding in order until $\mathsf{facwt}\big(F_j, j\big) = r''_j$, where the last facility may only be partially included in F_j.)

[1] This is why randomized appears in quotes in the title of this section.



a) At step C1 when the iteration starts:
clients j, k are in $\mathcal{S}$
$r''_j = 3$, $\mathsf{back}(j) = 1$, $a_j = 3$
$r''_k = 2$, $\mathsf{back}(k) = 1$, $a_k = 2$
sets $P\big(j^{(1)}\big), P\big(j^{(2)}\big)$ (and $P\big(k^{(1)}\big)$) have been initialized by previous iterations

b) After the iteration ends. During the iteration:
$j^{(3)}$ is chosen as cluster center, cluster M is created
$\mathsf{back}(j)$ and $\mathsf{back}(k)$ are decremented, so j, k are are no longer in $\mathcal{S}$
$P\big(j^{(3)}\big)$, $P\big(k^{(2)}\big)$ are initialized

Figure 4.2: An iteration of the clustering step in Phase 3.

Clustering. Define the backup requirement $\mathsf{back}(j) = r''_j$. 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 $\mathsf{facwt}\big(N_j, j\big) \geq \mathsf{back}(j)$ (see Lemma 5.1) for every j. Let $\mathcal{S} = \{j : \mathsf{back}(j) \geq 1\}$ be the set of candidate cluster centers. While $\mathcal{S} \neq \emptyset$ we repeatedly do the following.

C1. For each $j \in \mathcal{S}$, 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 $\mathsf{facwt}\big(N_j, j\big) \geq \mathsf{back}(j) \geq 1$) where the last facility i_l may be included partially, that is, to an extent x such that $\sum_{p<l} x_{i_p j} + x = 1$, $0 < x \leq x_{i_l j}$. So $C_j(a_j) = \sum_{p<l} c_{i_p j} x_{i_p j} + c_{i_l j} x$.

C2. Choose $j \in \mathcal{S}$ 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 $\mathsf{facwt}\big(\{i_1, \ldots, i_m\}, j\big) \geq 1 - \gamma$. Note that here *we do not split any facility* (see Fig. 4.2).

C3. For each k (including j) such that $N_k \cap M \neq \emptyset$, decrease $\mathsf{back}(k)$ by 1. Initialize $P\big(k^{(a_k)}\big) \leftarrow F_k \cap M = N_k \cap M$ (since facilities in M are previously unclustered) and set $a_k \leftarrow a_k + 1$, $N_k \leftarrow N_k \setminus M$ (see Fig. 4.2). 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 $P\big(j^{(c)}\big)$, $c = 1, \ldots, r''_j$, so that each $P\big(j^{(c)}\big)$ gets a facility weight of exactly 1. We do this by distributing the facilities remaining in N_j (i.e., the unclustered facilities in F_j) arbitrarily among these r''_j sets, splitting facilities across sets if necessary. By this we mean that if $z_{ij}(c) \geq 0$ denotes the amount of facility i allotted to copy c, then $\sum_i z_{ij}(c) = 1$ for every copy c, $\sum_c z_{ij}(c) = x_{ij}$ for every facility i, and if $i \in F_j \setminus N_j$ was allotted to $P\big(j^{(c)}\big)$ in the clustering step, then $z_{ij}(c) = x_{ij}$ and $z_{ij}(c') = 0$ for every other copy c' (see Fig. 4.4). Such a distribution of facilities is always possible since $\mathsf{facwt}\big(F_j, j\big) = r''_j$. The preferred set of copy $j^{(c)}$ is the set of facilities for which $z_{ij}(c) > 0$: $P\big(j^{(c)}\big) = \{i : z_{ij}(c) > 0\}$. Let $\{\hat{x}_{ij}, \hat{y}_i, \hat{z}_{ij}(c)\}$ denote respectively $\{x_{ij}, y_i, z_{ij}(c)\}/(1-\gamma)$. Now imagine that we have a randomized process with the following properties.

(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 $i \in P\big(j^{(c)}\big)$ with probability $\hat{z}_{ij}(c)$.

As we 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 above hypothetical randomized process, we write an expression $S_j^{\mathsf{loc}}(c)$ for the service cost of each copy $j^{(c)}$. Suppose $j^{(c)}$ is the center of a cluster M, so $M \subseteq P\big(j^{(c)}\big)$. Let $M = \{i_1, \ldots, i_m\}$ with $c_{i_1 j} \leq \ldots \leq c_{i_m j}$, and let $d_l = c_{i_l j}$. 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 $y_i = x_{ij} = z_{ij}(c)$ for each facility i in M, so we can write

$$S_j^{\mathsf{loc}}(c) = \hat{y}_{i_1} d_1 + (1 - \hat{y}_{i_1})\hat{y}_{i_2} d_2 + \cdots + (1 - \hat{y}_{i_1})(1 - \hat{y}_{i_2}) \ldots (1 - \hat{y}_{i_{m-1}}) d_m. \tag{2}$$

To avoid clutter, we have not explicitly indicated the functional dependence of $S_j^{\mathsf{loc}}(c)$ on the variables $\hat{y}_{i_1}, \ldots, \hat{y}_{i_m}$.

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

$$\begin{aligned} S_j^{\mathsf{loc}}(c) = \hat{z}_{i_1} d_1 + (1 - \hat{z}_{i_1})\hat{z}_{i_2} d_2 + \cdots + (1 - \hat{z}_{i_1})(1 - \hat{z}_{i_2}) \ldots (1 - \hat{z}_{i_{m-1}})\hat{z}_{i_m} d_m \\ + (1 - \hat{z}_{i_1}) \ldots (1 - \hat{z}_{i_m})\Big(\hat{y}_{b_1} c_1 + \cdots + (1 - \hat{y}_{b_1}) \ldots (1 - \hat{y}_{b_{q-1}}) c_q\Big). \end{aligned} \tag{3}$$

The rationale behind the expression is the same as earlier: we assign $j^{(c)}$ to the nearest open facility in $P\big(j^{(c)}\big)$ and if no such facility is open (in which case, some facility in $M \setminus P\big(j^{(c)}\big)$ must be open), to the nearest facility opened from cluster M.

Now for each j, we round the $\hat{z}_{ij}(c)$ values without increasing $\sum_c S_j^{\mathsf{loc}}(c)$, 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 $S_j^{\mathsf{loc}}(c)$ 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 $+\epsilon$ and $\hat{z}_{ij}(c')$ by $-\epsilon$, where ϵ is either $-\hat{z}_{ij}(c)$ or $+\hat{z}_{ij}(c')$, then since $S_j^{\mathsf{loc}}(c)$ and $S_j^{\mathsf{loc}}(c')$ are linear in $\hat{z}_{ij}(c)$ and $\hat{z}_{ij}(c')$ respectively, we can always make one of these local moves without increasing $\sum_c S_j^{\mathsf{loc}}(c)$. In fact, notice that the values of $S_j^{\mathsf{loc}}(c'')$ for copies $c'' \neq c, c'$, and that of $S_k^{\mathsf{loc}}(c)$ for any copy $k^{(c)}$ where $k \neq j$, remain unchanged. Thus we decrease the number of $z_{ij}(.)$ 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 4.4 shows a possible outcome of this process. Note that the $\hat{y}_i$ values are not changed in this step.

[(x, y) is an optimal solution to (FTFL-P). $\gamma < \frac{1}{2}$ is a parameter.]

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 $x_{ij} = y_i = 1$, 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 $L_2 = \{i : \gamma \le y_i < 1\}$. Open all the facilities in L_2. For a client j, define $L_j = \{i : \gamma \le x_{ij} < 1\}$. 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 $\sum_{i:x_{ij}<\gamma} x_{ij} \ge r''_j$.)

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

Clustering. For each client j, initialize $\mathsf{back}(j) \leftarrow r''_j$ and $a_j \leftarrow 1$; $\mathsf{back}(j)$ denotes the backup requirement of j, and a_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 $\mathcal{S} = \{j : \mathsf{back}(j) \ge 1\}$ be the set of candidate cluster centers.

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

C1. For each $j \in \mathcal{S}$, define $C_j(a_j)$ as follows. Let $i_1, \ldots, i_l$ be the first l facilities (in the sorted order) in N_j such that $\sum_{p<l} x_{i_p j} < 1 \le \sum_{p \le l} x_{i_p j}$. Define $C_j(a_j) = \sum_{p<l} c_{i_p j} x_{i_p j} + c_{i_l j}(1 - \sum_{p<l} x_{i_p j})$.

C2. Choose $j \in \mathcal{S}$ with minimum $C_j(a_j)$ 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 $\mathsf{facwt}\big(\{i_1, \ldots, i_m\}, j\big) \ge 1 - \gamma$ (see Fig. 4.2).

C3. For each k (including j) such that $N_k \cap M \neq \emptyset$, initialize $P\big(k^{(a_k)}\big) \leftarrow F_k \cap M = N_k \cap M$. Update $\mathsf{back}(k) \leftarrow \mathsf{back}(k) - 1,\ a_k \leftarrow a_k + 1$, and $N_k \leftarrow N_k \setminus M$ (see Fig. 4.2). We call M the *backup cluster* of copy $k^{(a_k)}$.

Pipage Rounding. For every client j, distribute the facilities remaining in N_j arbitrarily among the preferred sets $P\big(j^{(c)}\big)$, 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 Fig. 4.4) such that

(i) $z_{ij}(c) \ge 0$ and $\sum_i z_{ij}(c) = 1$ 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 $P\big(j^{(c)}\big)$ in the clustering step, then $z_{ij}(c) = x_{ij}$,

and set $P\big(j^{(c)}\big) = \{i : z_{ij}(c) > 0\}$. Let $\{\hat{x}_{ij}, \hat{y}_i, \hat{z}_{ij}(c)\}$ denote respectively $\{x_{ij}, y_i, z_{ij}(c)\}/(1 - \gamma)$.

Local Per-Client Rounding. For every client j, copy c, let $S_j^{\mathsf{loc}}(c)$ 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 ϵ is either $-z_{ij}(c)$ or $z_{ij}(c')$, so that $\sum_c S_j^{\mathsf{loc}}(c)$ does not increase. Repeat this until $z_{ij}(c) \in \{0, x_{ij}\}$ for every i, j, c. Update the sets $P\big(j^{(c)}\big)$ accordingly.

Global Rounding. Define $S_j^{\mathsf{glb}}(c) = S_j^{\mathsf{loc}}(c)$ if $j^{(c)}$ is a cluster-center, and by (4) otherwise. Define $T(\hat{y}_1, \ldots, \hat{y}_n) = \sum_{i:y_i<\gamma} f_i \hat{y}_i$, and $\mathsf{cost}(\hat{y}_1, \ldots, \hat{y}_n) = T(\hat{y}_1, \ldots, \hat{y}_n) + \sum_{j,c} S_j^{\mathsf{glb}}(c)$.
Define $h_{i,i'}(\epsilon) = \mathsf{cost}(\hat{y}_1, \ldots, \hat{y}_{i-1}, \hat{y}_i + \epsilon, \hat{y}_{i+1}, \ldots, \hat{y}_{i'-1}, \hat{y}_{i'} - \epsilon, \hat{y}_{i'+1}, \ldots, \hat{y}_n)$.

While there exists a cluster M with no fully-open facility do the following: pick indices i, i' such that $\hat{y}_i, \hat{y}_{i'} \in (0, 1)$. Let $\theta_1 = \min(\hat{y}_i, 1 - \hat{y}_{i'})$ and $\theta_2 = \min(1 - \hat{y}_i, \hat{y}_{i'})$. Compute $\epsilon^* \in [-\theta_1, \theta_2]$ such that $h_{i,i'}(\epsilon^*) = \min_{\epsilon \in [\theta_1, \theta_2]} h_{i,i'}(\epsilon)$ and update $\hat{y}_i \leftarrow \hat{y}_i + \epsilon^*,\ \hat{y}_{i'} \leftarrow \hat{y}_{i'} - \epsilon^*$.
For each remaining fractional $\hat{y}_i$, round $\hat{y}_i$ to 0 or 1, whichever decreases the value of $\mathsf{cost}(.)$.

For each copy $j^{(c)},\ c = 1, \ldots, r''_j$, 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 $P\big(j^{(c)}\big)$ if one is open, and to the nearest open facility from its backup cluster otherwise.

Figure 4.3: Summary of the "randomized" algorithm for FTFL.

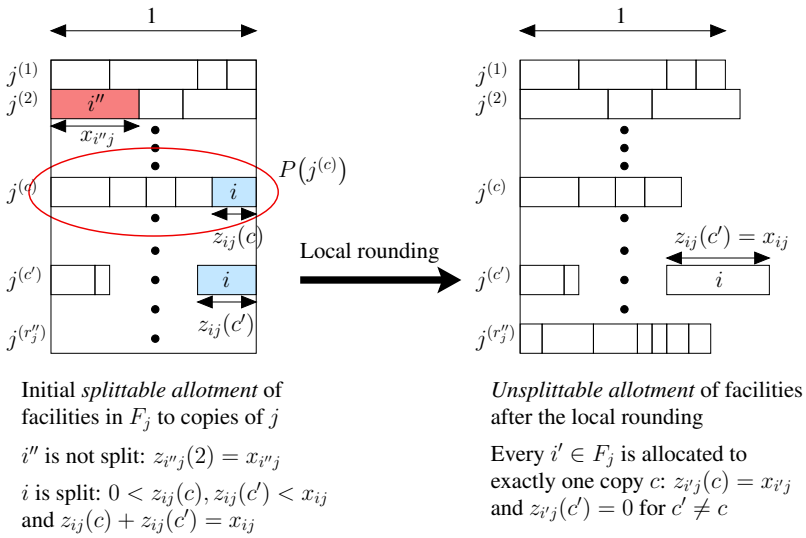


Figure 4.4: Local-Per-Client Rounding

Global Rounding. Now we round the $\hat{y}_i$ variables to 0-1 values. Each facility i with $y_i < \gamma$ is opened with probability $\hat{y}_i$, so we can write the facility cost as $T(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n) = \sum_{i:y_i<\gamma} f_i \hat{y}_i$, where $n = |\mathcal{F}|$. The service cost of a copy $j^{(c)}$ is given by the expression $S_j^{\mathsf{loc}}(c)$ at the end of the local rounding step. For a cluster center $j^{(c)}$, we set $S_j^{\mathsf{glb}}(c) = S_j^{\mathsf{loc}}(c)$. For a non-cluster-center copy $j^{(c)}$ we modify the expression for $S_j^{\mathsf{loc}}(c)$ 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 facilities for which $\hat{z}_{ij}(c) > 0$. So for any $i_l \in P(j^{(c)})$ we have $\hat{z}_{i_l j}(c) = \hat{x}_{i_l j}$ and this is equal to $\hat{y}_{i_l}$ for all but at most one facility. Therefore, the value of $S_j^{\mathsf{loc}}(c)$ depends only on the $\hat{y}_{i_l}$ values and perhaps one $\hat{x}_{i_l j}$ value; we would like to get an expression for the service cost of $j^{(c)}$ that depends only on the $\hat{y}_{i_l}$ values. So we modify the expression for $S_j^{\mathsf{loc}}(c)$ as follows: we substitute $\hat{x}_{i_l j}$ with $\hat{y}_{i_l}$, wherever it occurs. We use $S_j^{\mathsf{glb}}(c)$ 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 $c_{i_l j}$ for $l = 1, \ldots, m$, and c_l denote $c_{b_l j}$ for $l = 1, \ldots, q$. We define

$$S_j^{\mathsf{glb}}(c) = \hat{y}_{i_1 j} d_1 + (1 - \hat{y}_{i_1})\hat{y}_{i_2} d_2 + \cdots + (1 - \hat{y}_{i_1})(1 - \hat{y}_{i_2}) \ldots (1 - \hat{y}_{i_{m-1}})\hat{y}_{i_m} d_m \\ + (1 - \hat{y}_{i_1}) \ldots (1 - \hat{y}_{i_m}) \Big(\hat{y}_{b_1} c_1 + \cdots + (1 - \hat{y}_{b_1}) \ldots (1 - \hat{y}_{b_{q-1}}) c_q \Big). \quad (4)$$

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

$$\mathsf{cost}(\hat{y}_1, \ldots, \hat{y}_n) = T(\hat{y}_1, \ldots, \hat{y}_n) + \sum_{j,c} S_j^{\mathsf{glb}}(c).$$

We now convert the $\hat{y}_i$ values to $\{0, 1\}$-values without increasing $\mathsf{cost}(\hat{y}_1, \ldots, \hat{y}_n)$. 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 copy c, $S_j^{\mathsf{glb}}(c)$ is clearly an upper bound on the service cost of copy $j^{(c)}$. Hence $\sum_c S_j^{\mathsf{glb}}(c)$ is an upper bound on the service cost of j. Therefore $\mathsf{cost}(.)$ 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

$$h_{i,i'}(\epsilon) = \mathsf{cost}(\hat{y}_1, \ldots, \hat{y}_{i-1}, \hat{y}_i + \epsilon, \hat{y}_{i+1}, \ldots, \hat{y}_{i'-1}, \hat{y}_{i'} - \epsilon, \hat{y}_{i'+1}, \ldots, \hat{y}_n). \tag{5}$$

In the analysis, we show that $h_{i,i'}(.)$ is concave in ϵ 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 $h_{i,i'}(.)$ attains its minimum value (which is at most $h_{i,i'}(0) = \mathsf{cost}(\hat{y}_1, \ldots, \hat{y}_n)$) at one of the end points $\epsilon^* = -\theta_1$ 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 $\mathsf{cost}(.)$. 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 $\mathsf{cost}(.)$. (Note that $\mathsf{cost}(.)$ 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 $j^{(c)}$ (under this randomized process) is bounded by $S_j^{\mathsf{glb}}(c)$, 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 $S_j^{\mathsf{loc}}(c)$ 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 $\sum_c S_j^{\mathsf{loc}}(c)$ 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 $S_j^{\mathsf{loc}}(c)$ to the "global" expression $S_j^{\mathsf{glb}}(c)$ 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, $\mathsf{back}(k) \leq \mathsf{facwt}\big(N_k, k\big)$,

(ii) Each client k has at least $r''_k - \mathsf{back}(k)$ 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 all iterations up to the current iteration and consider the current iteration. Let M be the cluster created in this iteration and let client j be the cluster center. If there exists $i \in M$ such that $x_{ij} < y_i$, then it must be that i is the farthest facility serving j

and so $M = N_j$. But then $\mathsf{facwt}\big(N_j \setminus \{i\}, j\big) < 1 - \gamma$, and since $x_{ij} < \gamma$ we have $\mathsf{facwt}\big(N_j, j\big) = x_{ij} + \mathsf{facwt}\big(N_j \setminus \{i\}, j\big) < 1 \leq \mathsf{back}(j)$, contradicting the induction hypothesis. So $x_{ij} = y_i$ for all $i \in M$ and invariant (iii) is maintained. To show that (i) and (ii) hold, for any client k that is served by some facility in M, $\mathsf{facwt}\big(N_k, k\big)$ decreases by at most $\mathsf{facwt}\big(M, j\big) \leq 1$, while $\mathsf{back}(k)$ decreases by exactly 1. For each such k we also designate M as a backup cluster of the current active copy $k^{(a_k)}$, so the number of designated backup clusters increases by 1. For any other client k', $\mathsf{back}(k')$, $\mathsf{facwt}\big(N_{k'}, k'\big)$ and the number of backup clusters remain the same. ■

Since the clusters created are disjoint, Lemma 5.1 shows that at the end of clustering step, each copy $k^{(c)}$ has a distinct backup cluster allocated to it. 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. (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 $P\big(k^{(c')}\big) \cap M = \emptyset$.*

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.

$P\big(k^{(c)}\big)$ is initialized to $F_k \cap M$ in step C3, so no other preferred set $P\big(k^{(c')}\big)$ 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 $P\big(k^{(c')}\big)$, 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 $a_k = c$ at this point. Let i' be a facility in $F_k \cap M$ (which is non-empty). Since $x_{i'k} > 0$ we have $c_{i'k} \leq \alpha_k$ by complementary slackness. We will show that $\max_{i'' \in M} c_{i''j} \leq C_j(a_j)/\gamma$. This shows that

$$c_{ik} \leq c_{ij} + c_{i'j} + c_{i'k} \leq \alpha_k + 2C_j(a_j)/\gamma \leq \alpha_k + 2C_k(c)/\gamma,$$

where the last inequality follows since we chose j as the current cluster center and not k.

Let $A \subseteq N_j$ be the set of facilities, considered in sorted order, that gather a net x_{ij}-weight of 1 (the last facility may be partially included). $C_j(a_j)$ is the average distance to the facilities in A, and $M \subseteq A$ consists of the first m facilities (in sorted order) that gather an x_{ij}-weight of at least $1 - \gamma$. So if f is the last facility in M with $c_{fj} = \max_{i'' \in M} c_{i''j}$, and $B = (A \setminus M) \cup \{f\}$, then $C_j(a_j) \geq (\sum_{i'' \in B} x_{i''j})(\min_{i'' \in B} c_{i''j}) \geq \gamma \cdot c_{fj}$. ■

Define $\bar{C}_j = \sum_{i \in F_j} c_{ij} x_{ij}$. This is the cost that the LP pays to connect the r''_j copies of j.

Lemma 5.4 *For any j, we have $\sum_{c=1}^{r''_j} C_j(c) \leq \bar{C}_j$.*

Proof : Consider the ordered (by increasing c_{ij}) set of facilities F_j. For any $c, 1 \leq c \leq r''_j$ let $A \subseteq F_j$ be the set of facilities taken in order having an x_{ij}-weight of exactly c (the last facility may be chosen partially). Define $\tilde{C}_j(c)$ as the average distance to the set of facilities in A having a facility weight of exactly 1, picked

by starting from the last facility in A. Clearly $\tilde{C}_j(c) = \max\{\text{average distance to } S : S \subseteq A, \mathsf{facwt}\big(S, j\big) = 1\}$ and $\sum_c \tilde{C}_j(c) = \bar{C}_j$. Consider any iteration when $a_j = c$. At that point, $\mathsf{facwt}\big(N_j \cap A, j\big) \geq 1$ since prior to this point, whenever we remove some facility weight (of at most 1) from $N_j \cap A$ in an iteration we also increment a_j by 1, so in all we could have removed a facility weight of at most $c-1$ from A. $C_j(c)$ is the average distance to the set of facilities in N_j, starting from the one closest to j, that has x_{ij}-weight 1, so $C_j(c) \leq \tilde{C}_j(c)$ which completes the proof. ■

Lemma 5.5 *Let $d_1 \leq \ldots \leq d_{m+1}$ and $0 \leq p_l \leq 1$ for $l = 1, \ldots, m$. Consider the following expression.*

$$E(p_1, \ldots, p_m) = p_1 d_1 + (1-p_1)p_2 d_2 + \cdots + (1-p_1)\ldots(1-p_{m-1})p_m d_m + (1-p_1)\ldots(1-p_m)d_{m+1}.$$

The following properties hold.

(i) $E(.) \leq (1-p)\big(\sum_{l \leq m} p_l d_l\big)/\big(\sum_{l \leq m} p_l\big) + p \cdot d_{m+1}$ *where* $p = \prod_{l \leq m}(1-p_l)$,

(ii) $E(.)$ *is non-increasing in each variable* p_i,

(iii) Consider any two indices $i < i'$. *Let* $\Delta(\epsilon) = E(\ldots, p_i + \epsilon, \ldots, p_{i'} - \epsilon, \ldots)$ *and* $\theta_1 = \min(p_i, 1-p_{i'})$, $\theta_2 = \min(1-p_i, p_{i'})$. *Then* $\Delta(.)$ *is concave in* $[-\theta_1, \theta_2]$.

Proof : Part (i) was proved by Sviridenko [20] using the Chebyshev Integral Inequality (see [8]). 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 non-increasing and g_2 is monotonically non-decreasing. Then,

$$\int_a^b g_1(x)g_2(x)dx \leq \frac{\big(\int_a^b g_1(x)dx\big)\big(\int_a^b g_2(x)dx\big)}{b-a}.$$

We will use this to bound the sum of the first m terms of $E(.)$ by $(1-p)\big(\sum_{l \leq m} p_l d_l\big)/\big(\sum_{l \leq m} p_l\big)$. Take $g_1(x)$ and $g_2(x)$ to be functions defined on the interval $\big[0, P = \sum_{l \leq m} p_l\big)$ with $g_1(x) = \Pi_{l=1}^{i-1}(1-p_l)$ and $g_2(x) = d_i$ over the interval $\big[\sum_{l=1}^{i-1} p_l, \sum_{l=1}^{i} p_l\big)$ for $i = 1, \ldots, m$. This gives,

$$p_1 d_1 + (1-p_1)p_2 d_2 + \cdots + (1-p_1)(1-p_2)\ldots(1-p_{m-1})p_m d_m = \int_0^P g_1(x)g_2(x)dx \;\leq\; \frac{\big(\int_0^P g_1(x)dx\big)\big(\int_0^P g_2(x)dx\big)}{P} \;=\; \frac{\sum_{l \leq m} p_l d_l}{\sum_{l \leq m} p_l}\Big(1 - \prod_{l \leq m}(1-p_l)\Big).$$

To show (ii), we write $E(.)$ as $A + (1-p_1)\ldots(1-p_{i-1})(p_i d_i + (1-p_i)D)$, where $A = p_1 d_1 + \cdots + (1-p_1)\ldots(1-p_{i-2})p_{i-1}d_{i-1}$ and $D = p_{i+1}d_{i+1} + \cdots + (1-p_{i+1})\ldots(1-p_m)d_{m+1}$. Then, $D \geq d_i$ since $d_l \geq d_i$ for every $l \geq i+1$ and $p_{i+1} + \cdots + (1-p_{i+1})\ldots(1-p_m) = 1$. Consequently, if we increase p_i, then $E(.)$ decreases.

We prove (iii) by writing $\Delta(\epsilon) = A\epsilon^2 + B\epsilon + D$ and showing that $A \leq 0$. Clearly $\Delta(\epsilon)$ is quadratic in ϵ since each term of $E(.)$ is a polynomial function of ϵ of degree at most 2. The terms that contribute to the coefficient A are the 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 $D = p_{i'+1}d_{i'+1} + \cdots + (1-p_{i'+1})\ldots(1-p_m)d_{m+1}$. Again, since $d_l \geq d_{i'}$ for every $l \geq i'+1$, we have $D \geq d_{i'}$ and hence $A \leq 0$. ■

We can now bound $\sum_c S_j^{\mathsf{loc}}(c)$ and thus bound the service cost incurred.

Lemma 5.6 *Consider any client j. At any point in the local rounding step, the quantity $\sum_c S_j^{\mathsf{loc}}(c)$ is bounded by $\bar{C}_j\big(1 + e^{-1/(1-\gamma)}(\frac{2}{\gamma} - 1)\big) + e^{-1/(1-\gamma)} \cdot r''_j \alpha_j$.*

Proof : Since $\sum_c S_j^{\mathsf{loc}}(c)$ 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 $z_{ij}(c)$-weighted average distance from j to the facilities in $P\big(j^{(c)}\big)$, i.e., $D_j(c) = \sum_{i \in P\left(j^{(c)}\right)} z_{ij}(c) c_{ij}$. Clearly $\sum_c D_j(c) = \sum_{i \in F_j} c_{ij} x_{ij} = \bar{C}_j$. We will show that for each copy c,

$$S_j^{\mathsf{loc}}(c) \leq \Big(1 - e^{-1/(1-\gamma)}\Big) D_j(c) + e^{-1/(1-\gamma)} \Big(\alpha_j + 2C_j(c)/\gamma\Big). \tag{6}$$

Adding (6) over all copies $c = 1, \ldots, r''_j$ and using the fact that $\sum_{c=1}^{r''_j} C_j(c) \leq \bar{C}_j$ (Lemma 5.4), proves the lemma.

So fix copy c. Consider first the case when $j^{(c)}$ is not a cluster center. The expression for $S_j^{\mathsf{loc}}(c)$ is given by (3). Recall that $P\big(j^{(c)}\big) = \{i_1, \ldots, i_m\}$, $\hat{z}_{i_l}$ denotes $\hat{z}_{i_l j}(c) = z_{i_l j}(c)/(1-\gamma)$ and $d_l = c_{i_l j}$. Then, $(\sum_{l \leq m} \hat{z}_{i_l} d_l)/(\sum_{l \leq m} \hat{z}_{i_l}) = D_j(c)$. Let $p = \prod_{l \leq m}(1 - \hat{z}_{i_l}) \leq e^{-\sum_{l \leq m} \hat{z}_{i_l}} = e^{-1/(1-\gamma)}$, since $\sum_{l \leq m} z_{i_l} = 1$. In the last bracketed term in (3), each distance $c_l = c_{b_l j}$ for $l = 1, \ldots, q$ is at most $\alpha_j + 2C_j(c)/\gamma$ by Lemma 5.3, so since $\hat{y}_{b_1} + \cdots + (1 - \hat{y}_{b_1}) \ldots (1 - \hat{y}_{b_{q-1}}) = 1$, we can upper bound the coefficient of $(1 - \hat{z}_{i_1}) \ldots (1 - \hat{z}_{i_m})$ by $\alpha_j + 2C_j(c)/\gamma$. Substituting this upper bound, we get an expression that has the form as $E(.)$ in Lemma 5.5. So by part (i) of the lemma, we can bound $S_j^{\mathsf{loc}}(c)$ by $(1-p)D_j(c) + p \cdot (\alpha_j + 2C_j(c)/\gamma)$, which is at most the bound in (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 $S_j^{\mathsf{loc}}(c)$ is given by (2) where $M = \{i_1, \ldots, i_m\} \subseteq P\big(j^{(c)}\big)$. Let $\{i_{m+1}, \ldots, i_q\}$ be the facilities in $P\big(j^{(c)}\big) \setminus M$ 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 $S_j^{\mathsf{loc}}(c)$ to the function

$$f(x) = \hat{y}_{i_1} d_1 + \cdots + (1 - \hat{y}_{i_1}) \ldots (1 - \hat{y}_{i_{m-1}}) x d_m + \\ (1 - \hat{y}_{i_1}) \ldots (1 - \hat{y}_{i_{m-1}})(1 - x)\Big(\hat{y}_{i_{m+1}} d_{m+1} + \cdots + (1 - \hat{y}_{i_{m+1}}) \ldots (1 - \hat{y}_{i_q}) \alpha_j\Big).$$

Let $p = \prod_{l \leq q}(1 - \hat{y}_{i_l})$. We have, $S_j^{\mathsf{loc}}(c) = f(1) \leq f(\hat{y}_{i_m}) \leq (1-p)D_j(c) + p \cdot \alpha_j$ which is at most the bound in (6). The first inequality is due to the non-decreasing property from part (ii) of Lemma 5.5, and the second inequality is from part (i) of the same lemma. ∎

Lemma 5.7 *For any copy $j^{(c)}$, at the beginning of the global rounding step, we have $S_j^{\mathsf{glb}}(c) \leq S_j^{\mathsf{loc}}(c)$.*

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

Concavity of the function $h_{i,i'}(.)$ defined by (5) now follows since: (1) each $S_j^{\mathsf{glb}}(c)$ is of the same form as $E(.)$ in Lemma 5.5; (2) the contribution of $S_j^{\mathsf{glb}}(c)$ to $h_{i,i'}(.)$ is $S_j^{\mathsf{glb}}(c)(\ldots, \hat{y}_i + \epsilon, \ldots, \hat{y}_{i'} - \epsilon, \ldots)$ and this function of ϵ is either, constant if neither $\hat{y}_i$ nor $\hat{y}_{i'}$ appear in the expression for $S_j^{\mathsf{glb}}(c)$ (see (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 $\mathsf{cost}(.)$ is $T(\hat{y}_1, \ldots, \hat{y}_n)$ which is a linear function, therefore the part of $h_{i,i'}(.)$ corresponding to $T(\ldots, \hat{y}_i + \epsilon, \ldots, \hat{y}_{i'} - \epsilon, \ldots)$ is a linear function of ϵ. We can finally prove the main theorem.

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

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, and has 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.

Since the global rounding step does not increase the cost, we can bound the cost incurred in Phase 3 by the value of $T(\hat{y}_1,\ldots,\hat{y}_n)+\sum_{j,c} S_j^{\mathsf{glb}}(c)$ at the beginning of this step. The first term is $\sum_{i:y_i<\gamma} f_i y_i/(1-\gamma)$. The second term is bounded by $\sum_{j,c} S_j^{\mathsf{loc}}(c)$ (Lemma 5.7) which in turn is bounded by

$$\sum_j \Big(\bar{C}_j\big(1+e^{-1/(1-\gamma)}(\tfrac{2}{\gamma}-1)\big)+e^{-1/(1-\gamma)}\cdot r''_j\alpha_j\Big). \qquad \text{(by Lemma 5.6)}$$

We know that $\sum_{i:x_{ij}<\gamma} \geq r''_j$ for every client j, so using complementary slackness we get that $\sum_j r''_j\alpha_j$ is at most $\sum_{i:y_i<\gamma} f_i y_i + \sum_j \sum_{i:x_{ij}<\gamma} c_{ij}x_{ij}$. So the total cost of Phase 3 is at most

$$\begin{aligned}
&\frac{1}{1-\gamma}\cdot\sum_{i:y_i<\gamma} f_i y_i + \Big(1+e^{-1/(1-\gamma)}\big(\tfrac{2}{\gamma}-1\big)\Big)\sum_j\sum_{i:x_{ij}<\gamma} c_{ij}x_{ij} + e^{-1/(1-\gamma)}\cdot\sum_j r''_j\alpha_j \\
&\quad\leq \Big(\frac{1}{1-\gamma}+e^{-1/(1-\gamma)}\Big)\sum_{i:y_i<\gamma} f_i y_i + \Big(1+\tfrac{2}{\gamma}\cdot e^{-1/(1-\gamma)}\Big)\sum_j\sum_{i:x_{ij}<\gamma} c_{ij}x_{ij}. \qquad (7)
\end{aligned}$$

The total cost incurred in Phases 1 and 2 is at most, $\frac{1}{\gamma}\cdot\big(\sum_{i:y_i\geq\gamma} f_i y_i + \sum_j\sum_{i:x_{ij}\geq\gamma} c_{ij}x_{ij}\big)$ since each facility opened has $y_i\geq\gamma$ and if a copy of j connected to facility i then $x_{ij}\geq\gamma$. Combining this with (7), we see that the total cost is bounded by $\max\big(\frac{1}{\gamma}, \frac{1}{1-\gamma}+e^{-1/(1-\gamma)}, 1+\frac{2}{\gamma}\cdot e^{-1/(1-\gamma)}\big)\cdot OPT$. ■

6 A 1.52-approximation algorithm for uniform requirements

We now show that a modification of the algorithm given by Jain, Mahdian and Saberi [10] 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, Ye and Zhang [16] 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., [22], chapter 13). We will simultaneously construct a primal and 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 its copies have been connected, and *inactive* otherwise. If j is active we define the *active copy* of j, 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 that copy $j^{(c)}$ is connected to. If j is inactive we define l_j to be $\max_c$(distance between j and $i(j^{(c)})$). At any time t, the *contribution* of j to a *closed* facility i is $\max\big(\alpha_j^{(a_j)} - c_{ij}, 0\big)$, 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 happen:

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 $l_j = c_{i(j^{(c)})j}$ and connect it to i. Clearly l_j does not increase.

2. An active j reaches an open facility i (i.e., $\alpha_j^{(a_j)} = c_{ij}$) 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 $\sum_j \sum_{1 \le c \le r} \alpha_j^{(c)}$. We first define a dual solution (α, β, z) using the $\alpha_j^{(c)}$ variables. Let $\alpha_j = \alpha_j^{(r)} = \max_c \alpha_j^{(c)}$. Define $\theta_{ij} = \alpha_j - \alpha_j^{(c)}$, if $j^{(c)}$ is connected to i *and* j *was active when it got connected to* i, and 0 otherwise. In the former case we say that $j^{(c)}$ is *primarily connected* to i. Let $z_i = \sum_j \theta_{ij}$. Note that $\sum_j r\alpha_j - \sum_i z_i = \sum_{j,c} \alpha_j^{(c)}$, so the dual variables exactly pay for the cost of the primal. We will show that for each i, $\sum_j \alpha_j - z_i \le \gamma(\sum_j c_{ij} + f_i)$. Then we can define $\beta_{ij} = \alpha_j - \gamma c_{ij}$ so that the solution (α, β, z) would be infeasible by a factor of at most γ, implying an approximation ratio of γ. Further if we show that $\sum_{j \in S} \alpha_j - z_i \le \gamma_c \sum_{j \in S} c_{ij} + \gamma_f f_i$ 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 adding the corresponding inequalities weighted by y_i, we get that $\sum_j r\alpha_j - \sum_i z_i \le \gamma_f F^* + \gamma_c C^*$. Here F^* and C^* denote respectively the facility and connection cost of a fractional LP solution. (We need to be a little careful here since we may have $0 < x_{ij} < y_i$ 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 $\sum_T y_T = y_i$ and for each client j, $\sum_{T : j \in D_T} y_T = x_{ij}$.)

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 $\sum_{j \in S}(\alpha_j - \theta_{ij})$ and $(f_i + \sum_{j \in S} c_{ij})$. Let $f = f_i, d_j = c_{ij}$, and $v_j = \alpha_j - \theta_{ij}$. Note that $v_j = \alpha_j^{(c)}$ if $j^{(c)}$ is primarily connected to f, and $\alpha_j^{(r)}$ otherwise. We number the demands $1 \ldots k$ so that $v_1 \le \ldots \le v_k$. Consider the time $t = v_i - \epsilon$. At this time each demand $j < i$ is either inactive or a copy $j^{(c)}$ is primarily connected to f. Let $r_{j,i} = v_j$ if $j^{(c)}$ is primarily connected to f and l_j (at the time $v_i - \epsilon$) otherwise. Note that $r_{j,j+1} \ge \ldots \ge r_{j,k}$.

We now write some inequalities involving the variables $v_j, d_j, f, r_{j,i}$. Again consider time $t = v_i - \epsilon$. At this time the contribution of j to i is $\max(r_{j,i} - d_j, 0)$ if $j < i$ and $\max(t - d_j, 0)$ if $j \ge i$. Since the total contribution to a facility never exceeds its facility cost, we have,

$$\sum_{j<i} \max(r_{j,i} - d_j, 0) + \sum_{j \ge i} \max(v_i - d_j, 0) \le f \qquad \text{for all } i. \tag{8}$$

Now we use the triangle inequality. If f is open at time $t = v_i - \epsilon$ then $v_i \leq d_i$. Otherwise every demand $j < i$ is inactive at time t (since $j^{(c)}$ it 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, i is still active at time t. So it is connected to less than r facilities at time t, implying that there is some facility to which j is connected and 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,

$$v_i \leq r_{j,i} + d_i + d_j \qquad \text{for all } i, j < i. \tag{9}$$

Using the above inequalities we can write a mathematical program to bound the ratio of $\sum_{j \in S}(\alpha_j - \theta_{ij})$ and $(f_i + \sum_{j \in S} c_{ij})$. The variables $v_j, d_j, f, r_{j,i}$ obtained by running the algorithm form a feasible solution to the optimization problem below which can be written as a linear program (LP), so the ratio is bounded by the LP optimum value.

$$\begin{aligned}
\gamma_k \;=\; \max \quad & \frac{\sum_{j \leq k} v_j}{f + \sum_{j \leq k} d_j} && && \text{(LP)}\\
\text{such that} \quad & v_1 \leq \ldots \leq v_k && &&\\
& r_{j,j+1} \geq \ldots \geq r_{j,k} && \text{for all } j &\\
& v_i \leq r_{j,i} + d_i + d_j && \text{for all } i, j < i &\\
& \sum_{j<i} \max(r_{j,i} - d_j, 0) + \sum_{j \geq i} \max(v_i - d_j, 0) \leq f && \text{for all } i &\\
& v_j, d_j, f, r_{j,i} \geq 0 && &&
\end{aligned}$$

This is the same as the so-called factor-revealing LP in [10], so all the results in [10] hold for this algorithm too. In particular we have the following (Lemma 13, Theorem 4 in [10]).

Lemma 6.1 $\gamma_k \leq 1.61$ *for all* k.

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

We say that an algorithm is a (γ_f, γ_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 (γ_f, γ_c) pair for which the algorithm is a (γ_f, γ_c)-approximation algorithm. Theorem 9 in [10] and Lemma 2 in [16] establish the following.

Theorem 6.3 *Let* $\gamma_f \geq 1$. *Define* γ'_k *as* $\max \frac{\sum_{j \leq k} v_j - \gamma_f f}{\sum_{j \leq k} d_j}$ *subject to the same set of constraints as* (LP)*, and* $\gamma_c = \sup_k \{\gamma'_k\}$. *The algorithm above is a* (γ_f, γ_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 above algorithm is a (1,2)- and a (1.11,1.78)-approximation algorithm.*

6.3 Scaling and greedy augmentation

It is possible to improve the performance of the above algorithm by using scaling and greedy augmentation [6, 3]. The combined algorithm is as follows :

Algorithm FTUFL(δ)

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

The lemma below was proved in [6, 3], and in the context of fault-tolerant facility location in [7].

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 $F+F^* \max\left\{0, \ln\left(\frac{C-C^*}{F^*}\right)\right\}+F^*+C^*$.*

Lemma 6.4 implies the following :

Lemma 6.5 ([3, 16]) *Let $\mathcal{A}$ be a (γ_f, γ_c)-approximation algorithm. The above 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 $(\gamma_f, \gamma_c) = (1.11, 1.78)$ 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 facilities. 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 $\sum_i y_i \leq k$. This modifies the objective function of the dual (FTFL-D) to $\max \sum_j r_j \alpha_j - \sum_i z_i - k\Delta$, and constraint (1) changes to $\sum_j \beta_{ij} \leq f_i + z_i + \Delta$. Let (KP) and (KD) 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 α_j, z_i. We have that $2F + C = \sum_j \alpha_j - \sum_i z_i$, and by Theorem 6.3 we know that for $\gamma_f = 1, \gamma_c \leq 2$, so for any facility i and set of demands $S, \sum_j \alpha_j - 2f_i - z_i \leq 2\sum_{j \in S} c_{ij}$. So if we set $(\alpha', z') = (\alpha/2, z/2)$ and $\beta'_{ij} = \alpha'_j - c_{ij}$, then (α', β', z') is a feasible dual solution and $2F + C \leq 2(\sum_j \alpha'_j - \sum_i z'_i)$. In the sequel whenever we say that "we run the primal-dual algorithm" we mean run this modified algorithm where we first scale the facility costs by a factor of 2 and then run the original primal-dual algorithm. Also when we say 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 above so that $2\hat{F} + \hat{C} \leq 2(\sum_j \hat{\alpha} - \sum_i \hat{z}_i)$.

Consider fixing Δ and running the above algorithm with the facility costs modified to $f_i + \Delta$ (i.e. we first scale $(f_i + \Delta)$ by 2 and then run the original primal-dual algorithm). Suppose the algorithm returns a primal solution (F, C) in which exactly k facilities are opened, and a dual solution (α, β, z). So the primal solution is a feasible solution to (KP) and $(\alpha, \beta, z, \Delta)$ is a feasible solution to (KD). Also, $2(F + k\Delta) + C \leq 2(\sum_j \alpha_j - \sum_i z_i) \implies F + C \leq 2(\sum_j \alpha_j - \sum_i z_i - k\Delta) \leq 2OPT_k$, so we have a solution of cost at most $2OPT_k$. The basic idea then is to "guess" the right value of Δ so that when the facility costs are modified to $f_i + \Delta$, the algorithm ends up opening k facilities. This idea was first used by Jain and Vazirani [12] for the (non-fault-tolerant, i.e., $r_j = 1$) k-median problem. If at $\Delta = 0$, the algorithm opens at most k facilities, then by the same reasoning as above we have a feasible solution of cost at most $2OPT_k$. So assume that we open more than k facilities at $\Delta = 0$. When Δ is very large, say $nr \max_{ij} c_{ij}$, the algorithm will open just r facilities and connect all demands to these facilities. We can show that there is a value $\Delta = \Delta_0$ 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 $[0, nr \max_{ij} c_{ij}]$ and terminating the search when the length of the search interval becomes less than $2^{(-\operatorname{poly}(n)+L)}$ where $L = \log(\max_{ij} c_{ij})$. The proof of this is very similar to the proof in the conference version of [12].

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 to (KP) and opens exactly k facilities. Let a, b be such that $ak_1 + bk_2 = k,\ a + b = 1$. Then,

$$2(aF_1 + bF_2) + (aC_1 + bC_2) \leq 2\Big(\sum_j \alpha_j - \sum_i z_i - k\Delta\Big) \leq 2 \cdot OPT_k. \tag{10}$$

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 (10) we get that $F_1 + C_1 \leq 4 \cdot OPT_k$. Otherwise we open a subset of the facilities opened in y_2. We call a facility opened in y_1 a *"small"* facility and a facility 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 an 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 $b = \frac{k-k_1}{k_2-k_1}$, 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* $2\sum_i c_{ij}(ax_{1,ij} + bx_{2,ij})$.

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, i.e., $S_j = \{i : x_{1,ij} = 1\}$. 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 $|m(S_j)| = |S_j| = |L_j| \implies |m(S_j) \setminus L_j| = |L_j \setminus m(S_j)|$, so the number of sets T_c that do not contain 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 $\mathrm{E}\left[X^{(c)}\right] = ac_{i_1 j} + bc_{i_2 j}$. 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 $\mathrm{E}\left[X^{(c)}\right] = bc_{i_3 j} + a(ac_{i_1 j} + bc_{i_2 j})$. 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 i_3. So, $c_{i_2 j} \le c_{i_1 j} + c_{i_1 i_2} \le c_{i_1 j} + c_{i_1 i_3} \le 2c_{i_1 j} + c_{i_3 j}$. Therefore,

$$\mathrm{E}\left[X^{(c)}\right] \le bc_{i_3 j} + a(ac_{i_1 j} + 2bc_{i_1 j} + bc_{i_3 j}) = a(1+b)c_{i_1 j} + b(1+a)c_{i_3 j} \le 2(ac_{i_1 j} + bc_{i_3 j})$$

So for every copy c, if $i, i' \in T_c$ where $i \in S_j$ and $i' \in L_j$, we have $\mathrm{E}\left[X^{(c)}\right] \le 2(ac_{ij} + bc_{i'j}) = 2(ac_{ij}x_{1,ij} + bc_{i'j}x_{2,i'j})$ since $x_{1,ij} = x_{2,i'j} = 1$. 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 $\mathrm{E}\left[X\right] \le 2(\sum_{i \in S_j} ac_{ij}x_{1,ij} + \sum_{i \in L_j} bc_{ij}x_{2,ij}) = 2\sum_i c_{ij}(ax_{1,ij} + bx_{2,ij})$ where the last equality follows since if $i \notin S_j$ then $x_{1,ij} = 0$, and if $i \notin L_j$ then $x_{2,ij} = 0$. ■

Theorem 7.3 *The above algorithm returns a solution of expected cost at most 4 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 $2\sum_i c_{ij}(ax_{1,ij} + bx_{2,ij})$. Also we have $C_1 = \sum_{j,i} c_{ij}x_{1,ij}$ and $C_2 = \sum_{j,i} c_{ij}x_{2,ij}$. So summing over all j, we see that the expected total service cost is at most $2(aC_1 + bC_2)$. Combining this with Lemma 7.1, the expected total cost of the solution returned is at most $(aF_1 + bF_2) + 2(aC_1 + bC_2) \le 4 \cdot OPT_k$, from (10). ■

Acknowledgments

We thank the anonymous referees for their useful suggestions.

References

[1] A. Ageev and M. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.

[2] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

[3] M. Charikar and S. Guha. Improved combinatorial algorithms for facility location problems. *SIAM Journal on Computing*, 34(4):803–824, 2005.

[4] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[5] F. Chudak and D. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25, 2003.

[6] S. Guha and S. Khuller. Greedy strikes back: improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

[7] S. Guha, A. Meyerson, and K. Munagala. Improved algorithms for fault tolerant facility location. *Journal of Algorithms*, 48(2):429–440, 2003.

[8] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1952.

[9] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. Greedy facility location algorithms analyzed using dual-fitting with factor-revealing LP. *Journal of the ACM* 50(6):795–824, 2003.

[10] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 731–740, 2002.

[11] K. Jain and V.V. Vazirani. An approximation algorithm for the fault tolerant metric facility location problem. *Algorithmica*, 38(3):433–439, 2003.

[12] K. Jain and V.V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM* 48(2):274–296, 2001. Preliminary version in *FOCS '99*.

[13] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37(1):146–188, 2000.

[14] J. H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.

[15] M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. A greedy facility location algorithm analyzed using dual-fitting. In *Proceedings of 4th APPROX*, pages 127–137, 2001.

[16] M. Mahdian, Y. Ye, and J. Zhang. Approximation algorithms for metric facility location problems. *SIAM Journal on Computing*, 36(2):411–432, 2006.

[17] R.R. Mettu and C.G. Plaxton. The online median problem. *SIAM Journal on Computing*, 32(3):816–832, 2003.

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

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

[20] M. Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of 9th IPCO*, pages 240–257, 2002.

[21] C. Swamy and D. B. Shmoys. Fault-tolerant facility location. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 735–736, 2003.

[22] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, NY, 2001.