LP-rounding Algorithms for the Fault-Tolerant Facility Placement Problem

Li Yan and Marek Chrobak
Department of Computer Science
University of California at Riverside

Abstract

The Fault-Tolerant Facility Placement problem (FTFP) is a generalization of the classic Uncapacitated Facility Location Problem (UFL). In FTFP we are given a set of facility sites and a set of clients. Opening a facility at site i costs f_i and connecting client j to a facility at site i costs d_{ij}. We assume that the connection costs (distances) d_{ij} satisfy the triangle inequality. Multiple facilities can be opened at any site. Each client j has a demand r_j, which means that it needs to be connected to r_j different facilities (some of which could be located on the same site). The goal is to minimize the sum of facility opening cost and connection cost.

The main result of this paper is a 1.575-approximation algorithm for FTFP, based on LP-rounding. The algorithm first reduces the demands to values polynomial in the number of sites. Then it uses a technique that we call adaptive partitioning, which partitions the instance by splitting clients into unit demands and creating a number of (not yet opened) facilities at each site. It also partitions the optimal fractional solution to produce a fractional solution for this new instance. The partitioned instance satisfies a number of properties that allow us to exploit existing LP-rounding methods for UFL to round our partitioned solution to an integral solution, preserving the approximation ratio. In particular, our 1.575-approximation algorithm is based on the ideas from the 1.575-approximation algorithm for UFL by Byrka *et al.*, with changes necessary to satisfy the fault-tolerance requirement.

1 Introduction

In the *Fault-Tolerant Facility Placement* problem (FTFP), we are given a set $\mathbb{F}$ of *sites* at which facilities can be built, and a set $\mathbb{C}$ of *clients* with some demands that need to be satisfied by different facilities. A client j has demand r_j. Building one facility at a site i incurs a cost f_i, and connecting one unit of demand from client j to a facility at site $i \in \mathbb{F}$ costs d_{ij}. Throughout the paper we assume that the connection costs (distances) d_{ij} form a metric, that is, they are symmetric and satisfy the triangle inequality. In a feasible solution, some number of facilities, possibly zero, are opened at each site i, and demands from each client are connected to those open facilities, with the constraint that demands from the same client have to be connected to different facilities. Note that facilities at the same site are considered different.

It is easy to see that if all $r_j = 1$ then FTFP reduces to the classic Uncapacitated Facility Location problem (UFL). If we add a constraint that each site can have at most one facility built, then the problem becomes equivalent to the Fault-Tolerant Facility Location problem (FTFL). One implication of the one-facility-per-site restriction in FTFL is that $\max_{j\in\mathbb{C}} r_j \leq |\mathbb{F}|$, while in FTFP the values of r_j's can be much bigger than $|\mathbb{F}|$.

The UFL problem has a long history; in particular, great progress has been achieved in the past two decades in developing techniques for designing constant-ratio approximation algorithms for UFL. Shmoys, Tardos and Aardal [15] proposed an approach based on LP-rounding, that they used to achieve a ratio of 3.16. This was then improved by Chudak [5] to 1.736, and later by Sviridenko [16] to 1.582. The best known "pure" LP-rounding algorithm is due to Byrka *et al.* [3] with ratio 1.575. Byrka and Aardal [2] gave a hybrid algorithm that combines LP-rounding and dual-fitting (based on [10]), achieving a ratio of 1.5. Recently, Li [13] showed that, with a more refined analysis and randomizing the scaling parameter used in [2], the ratio can be improved to 1.488. This is the best known approximation result for UFL. Other techniques include the primal-dual algorithm with ratio 3 by Jain and Vazirani [11], the dual fitting method by Jain *et al.* [10] that gives ratio 1.61, and a local search heuristic by Arya *et al.* [1] with approximation ratio 3. On the hardness side, UFL is easily shown to be NP-hard, and it is known that it is not possible to approximate UFL in polynomial time with ratio less than 1.463, provided that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{O(\log\log n)})$ [6]. An observation by Sviridenko strengthened the underlying assumption to $\mathsf{P} \neq \mathsf{NP}$ (see [18]).

FTFL was first introduced by Jain and Vazirani [12] and they adapted their primal-dual algorithm for UFL to obtain a ratio of $3\ln(\max_{j\in\mathbb{C}} r_j)$. All subsequently discovered constant-ratio approximation algorithms use variations of LP-rounding. The first such algorithm, by Guha *et al.* [7], adapted the approach for UFL from [15]. Swamy and Shmoys [17] improved the ratio to 2.076 using the idea of pipage rounding introduced in [16]. Most recently, Byrka *et al.* [4] improved the ratio to 1.7245 using dependent rounding and laminar clustering.

FTFP is a natural generalization of UFL. It was first studied by Xu and Shen [19], who extended the dual-fitting algorithm from [10] to give an approximation algorithm with a ratio claimed to be 1.861. However their algorithm runs in polynomial time only if $\max_{j\in\mathbb{C}} r_j$ is polynomial in $O(|\mathbb{F}|\cdot|\mathbb{C}|)$ and the analysis of the performance guarantee in [19] is flawed[1]. To date, the best approximation ratio for FTFP in the literature is 4, established by Yan and Chrobak [20], while the only known lower bound is the 1.463 lower bound for UFL from [6], as UFL is a special case of FTFP.

The main result of this paper is an LP-rounding algorithm for FTFP with approximation ratio

[1] Confirmed through private communication with the authors.

1.575, matching the best ratio for UFL achieved via the LP-rounding method [3] and significantly improving our earlier bound in [20]. In Section 3 we prove that, for the purpose of LP-based approximations, the general FTFP problem can be reduced to the restricted version where all demand values are polynomial in the number of sites. This *demand reduction* trick itself gives us a ratio of 1.7245, since we can then treat an instance of FTFP as an instance of FTFL, by creating a sufficient (but polynomial) number of facilities at each site and using the algorithm from [4].

The reduction to polynomial demands suggests an approach where clients' demands are split into unit demands. These unit demands can be thought of as "unit-demand clients", and a natural approach would be to adapt LP-rounding methods from [8, 5, 3] to this new set of unit-demand clients. Roughly, these algorithms iteratively pick a client that minimizes a certain cost function (that varies for different algorithms) and open one facility in the neighborhood of this client. The remaining clients are then connected to these open facilities. In order for this to work, we also need to convert the optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of the original instance into a solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ of the modified instance which then can be used in the LP-rounding process. This can be thought of as partitioning the fractional solution, as each connection value x^*_{ij} must be somehow divided between the r_j unit demands of client j. The partitioned fractional solution needs to possess certain properties to apply rounding algorithms similar to those for UFL. In Section 4 we present our *adaptive partitioning* technique by first stating a set of desired properties, and then present a natural way of partitioning the instance (and the optimal fractional solution). We also prove that the partitioned solution satisfies all prescribed properties. Using adaptive partitioning we were able to extend the algorithms for UFL from [8, 5, 3] to FTFP. We illustrate the fundamental ideas of our approach in Section 5, showing how they can be used to design an LP-rounding algorithm with ratio 3. The algorithm relies heavily on the properties of the instance and its associated fractional solution produced by adaptive partitioning; in particular, these properties allow us to assign facilities to demands so that two demands from the same client are assigned to different facilities. In Section 6 we refine the algorithm to improve the approximation ratio to $1+2/e \approx 1.736$. Finally, in Section 7, we improve it even further to 1.575 – the main result of this paper.

Summarizing, our contributions are two-fold: One, we show that the existing LP-rounding algorithms for UFL can be extended to a much more general problem FTFP, retaining the approximation ratio. We believe that, should even better LP-rounding algorithms be developed for UFL in the future, using our demand reduction and adaptive partitioning methods, it should be possible to extend them to FTFP. In fact, some improvement of the ratio should be achieved by randomizing the scaling parameter γ used in our algorithm, as Li showed in [13] for UFL. (Since the ratio 1.488 for UFL in [13] uses also dual-fitting algorithms [14], we would not obtain the same ratio for FTFP yet using only LP-rounding.)

Two, our ratio of 1.575 is significantly better than the best currently known ratio of 1.7245 for the closely-related FTFL problem. This suggests that in the fault-tolerant scenario the capability of creating additional copies of facilities on the existing sites makes the problem easier from the point of view of approximation.

2 The LP Formulation

The FTFP problem has a natural Integer Programming (IP) formulation. Let y_i represent the number of facilities built at site i and let x_{ij} represent the number of connections from client j to facilities at site i. If we relax the integrality constraints, we obtain the following LP:

$$\begin{aligned} \text{minimize} \quad & cost(\boldsymbol{x}, \boldsymbol{y}) = \textstyle\sum_{i\in\mathbb{F}} f_i y_i + \sum_{i\in\mathbb{F}, j\in\mathbb{C}} d_{ij} x_{ij} \\ \text{subject to} \quad & y_i - x_{ij} \geq 0 \qquad \forall i \in \mathbb{F}, j \in \mathbb{C} \\ & \textstyle\sum_{i\in\mathbb{F}} x_{ij} \geq r_j \qquad \forall j \in \mathbb{C} \\ & x_{ij} \geq 0, y_i \geq 0 \qquad \forall i \in \mathbb{F}, j \in \mathbb{C} \end{aligned} \tag{1}$$

The dual program is:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{j\in\mathbb{C}} r_j \alpha_j \\ \text{subject to} \quad & \textstyle\sum_{j\in\mathbb{C}} \beta_{ij} \leq f_i \qquad \forall i \in \mathbb{F} \\ & \alpha_j - \beta_{ij} \leq d_{ij} \qquad \forall i \in \mathbb{F}, j \in \mathbb{C} \\ & \alpha_j \geq 0, \beta_{ij} \geq 0 \qquad \forall i \in \mathbb{F}, j \in \mathbb{C} \end{aligned} \tag{2}$$

In each of our algorithms we will fix some optimal solutions of the LPs (1) and (2) that we will denote by $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$, respectively.

With $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ fixed, we can define the optimal facility cost as $F^* = \sum_{i\in\mathbb{F}} f_i y_i^*$ and the optimal connection cost as $C^* = \sum_{i\in\mathbb{F}, j\in\mathbb{C}} d_{ij} x_{ij}^*$. Then $\text{LP}^* = cost(\boldsymbol{x}^*, \boldsymbol{y}^*) = F^* + C^*$ is the joint optimal value of (1) and (2). We can also associate with each client j its fractional connection cost $C_j^* = \sum_{i\in\mathbb{F}} d_{ij} x_{ij}^*$. Clearly, $C^* = \sum_{j\in\mathbb{C}} C_j^*$. Throughout the paper we will use notation OPT for the optimal integral solution of (1). OPT is the value we wish to approximate, but, since $\text{OPT} \geq \text{LP}^*$, we can instead use LP^* to estimate the approximation ratio of our algorithms.

Completeness and facility splitting. Define $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to be *complete* if $x_{ij}^* > 0$ implies that $x_{ij}^* = y_i^*$ for all i, j. In other words, each connection either uses a site fully or not at all. As shown by Chudak and Shmoys [5], we can modify the given instance by adding at most $|\mathbb{C}|$ sites to obtain an equivalent instance that has a complete optimal solution, where "equivalent" means that the values of F^*, C^* and LP^* are not affected. Roughly, the argument is this: We notice that, without loss of generality, for each client k there exists at most one site i such that $0 < x_{ik}^* < y_i^*$. We can then perform the following *facility splitting* operation on i: introduce a new site i', let $y_{i'}^* = y_i^* - x_{ik}^*$, redefine y_i^* to be x_{ik}^*, and then for each client j redistribute x_{ij}^* so that i retains as much connection value as possible and i' receives the rest. Specifically, we set

$$y_{i'}^* \leftarrow y_i^* - x_{ik}^*, \qquad y_i^* \leftarrow x_{ik}^*,$$

and

$$x_{i'j}^* \leftarrow \max(x_{ij}^* - x_{ik}^*, 0), \quad x_{ij}^* \leftarrow \min(x_{ij}^*, x_{ik}^*) \quad \text{for all } j \neq k.$$

This operation eliminates the partial connection between k and i and does not create any new partial connections. Each client can split at most one site and hence we shall have at most $|\mathbb{C}|$ more sites.

By the above paragraph, without loss of generality we can assume that the optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is complete. This assumption will in fact greatly simplify some of the arguments in the paper. Additionally, we will frequently use the facility splitting operation in our algorithms to obtain fractional solutions with desirable properties.

3 Reduction to Polynomial Demands

This section presents a *demand reduction* trick that reduces the problem for arbitrary demands to a special case where demands are bounded by $|\mathbb{F}|$, the number of sites. (The formal statement is a little more technical – see Theorem 2.) Our algorithms in the sections that follow process individual demands of each client one by one, and thus they critically rely on the demands being bounded polynomially in terms of $|\mathbb{F}|$ and $|\mathbb{C}|$ to keep running time polynomial.

The reduction is based on an optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of LP (1). From optimality of the solution, we can assume that $\sum_{i\in\mathbb{F}} x^*_{ij} = r_j$ for all $j \in \mathbb{C}$. As explained in Section 2, we can assume that $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is complete, that is $x^*_{ij} > 0$ implies $x^*_{ij} = y^*_i$ for all i, j. We split this solution into two parts, namely $(\boldsymbol{x}^*, \boldsymbol{y}^*) = (\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) + (\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$, where

$$\hat{y}_i = \lfloor y^*_i \rfloor, \quad \hat{x}_{ij} = \lfloor x^*_{ij} \rfloor \text{ and}$$
$$\dot{y}_i = y^*_i - \lfloor y^*_i \rfloor, \quad \dot{x}_{ij} = x^*_{ij} - \lfloor x^*_{ij} \rfloor$$

for all i, j. Now we construct two FTFP instances $\hat{\mathcal{I}}$ and $\dot{\mathcal{I}}$ with the same parameters as the original instance, except that the demand of each client j is $\hat{r}_j = \sum_{i\in\mathbb{F}} \hat{x}_{ij}$ in instance $\hat{\mathcal{I}}$ and $\dot{r}_j = \sum_{i\in\mathbb{F}} \dot{x}_{ij} = r_j - \hat{r}_j$ in instance $\dot{\mathcal{I}}$. It is obvious that if we have integral solutions to both $\hat{\mathcal{I}}$ and $\dot{\mathcal{I}}$ then, when added together, they form an integral solution to the original instance. Moreover, we have the following lemma.

Lemma 1. (i) $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ *is a feasible integral solution to instance* $\hat{\mathcal{I}}$.
(ii) $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$ *is a feasible fractional solution to instance* $\dot{\mathcal{I}}$.
(iii) $\dot{r}_j \le |\mathbb{F}|$ *for every client* j.

Proof. (i) For feasibility, we need to verify that the constraints of LP (1) are satisfied. Directly from the definition, we have $\hat{r}_j = \sum_{i\in\mathbb{F}} \hat{x}_{ij}$. For any i and j, by the feasibility of $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ we have $\hat{x}_{ij} = \lfloor x^*_{ij} \rfloor \le \lfloor y^*_i \rfloor = \hat{y}_i$.

(ii) From the definition, we have $\dot{r}_j = \sum_{i\in\mathbb{F}} \dot{x}_{ij}$. It remains to show that $\dot{y}_i \ge \dot{x}_{ij}$ for all i, j. If $x^*_{ij} = 0$, then $\dot{x}_{ij} = 0$ and we are done. Otherwise, by completeness, we have $x^*_{ij} = y^*_i$. Then $\dot{y}_i = y^*_i - \lfloor y^*_i \rfloor = x^*_{ij} - \lfloor x^*_{ij} \rfloor = \dot{x}_{ij}$.

(iii) From the definition of $\dot{x}_{ij}$ we have $\dot{x}_{ij} < 1$. Then the bound follows from the definition of $\dot{r}_j$. □

Notice that our construction relies on the completeness assumption; in fact, it is easy to give an example where $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$ would not be feasible if we used a non-complete optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. Note also that the solutions $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$ are in fact optimal for their corresponding instances, for if a better solution to $\hat{\mathcal{I}}$ or $\dot{\mathcal{I}}$ existed, it could give us a solution to $\mathcal{I}$ with a smaller objective value.

Theorem 2. *Suppose that there is a polynomial-time algorithm $\mathcal{A}$ that, for any instance of* FTFP *with maximum demand bounded by $|\mathbb{F}|$, computes an integral solution that approximates the fractional optimum of this instance within factor $\rho \geq 1$. Then there is a ρ-approximation algorithm $\mathcal{A}'$ for* FTFP.

Proof. Given an FTFP instance with arbitrary demands, Algorithm $\mathcal{A}'$ works as follows: it solves the LP (1) to obtain a fractional optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, then constructs instances $\hat{\mathcal{I}}$ and $\dot{\mathcal{I}}$ described above, applies algorithm $\mathcal{A}$ to $\dot{\mathcal{I}}$, and finally combines (by adding the values) the integral solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of $\hat{\mathcal{I}}$ and the integral solution of $\dot{\mathcal{I}}$ produced by $\mathcal{A}$. This clearly produces a feasible integral solution for the original instance $\mathcal{I}$. The solution produced by $\mathcal{A}$ has cost at most $\rho \cdot cost(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$, because $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$ is feasible for $\dot{\mathcal{I}}$. Thus the cost of $\mathcal{A}'$ is at most

$$cost(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) + \rho \cdot cost(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}) \leq \rho(cost(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) + cost(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})) = \rho \cdot \mathrm{LP}^* \leq \rho \cdot \mathrm{OPT},$$

where the first inequality follows from $\rho \geq 1$. This completes the proof. □

4 Adaptive Partitioning

In this section we develop our second technique, which we call *adaptive partitioning*. Given an FTFP instance and an optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to LP (1), we split each client j into r_j individual *unit demand points* (or just *demands*), and we split each site i into no more than $|\mathbb{F}| + R|\mathbb{C}|$ *facility points* (or *facilities*), where $R = \max_{j \in \mathbb{C}} r_j$. We denote the demand set by $\overline{\mathbb{C}}$ and the facility set by $\overline{\mathbb{F}}$, respectively. We will also partition $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ into a fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ for the split instance. We will typically use symbols ν and μ to index demands and facilities respectively, that is $\bar{\boldsymbol{x}} = (\bar{x}_{\mu\nu})$ and $\bar{\boldsymbol{y}} = (\bar{y}_\mu)$. As before, the *neighborhood of a demand* ν is $\overline{N}(\nu) = \left\{\mu \in \overline{\mathbb{F}} : \bar{x}_{\mu\nu} > 0\right\}$. We will use notation $\nu \in j$ to mean that ν is a demand of client j; similarly, $\mu \in i$ means that facility μ is on site i. Different demands of the same client (that is, $\nu, \nu' \in j$) are called *siblings*. Further, we use the convention that $f_\mu = f_i$ for $\mu \in i$, $\alpha^*_\nu = \alpha^*_j$ for $\nu \in j$ and $d_{\mu\nu} = d_{\mu j} = d_{ij}$ for $\mu \in i$ and $\nu \in j$. We define $C^{\mathrm{avg}}_\nu = \sum_{\mu \in \overline{N}(\nu)} d_{\mu\nu} \bar{x}_{\mu\nu} = \sum_{\mu \in \overline{\mathbb{F}}} d_{\mu\nu} \bar{x}_{\mu\nu}$. Similar to C^*_j, one can think of C^{avg}_ν as the average connection cost of demand ν, if we chose a connection to facility μ with probability $\bar{x}_{\mu\nu}$. In our partitioned fractional solution we guarantee for every ν that $\sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu} = 1$.

Some demands in $\overline{\mathbb{C}}$ will be designated as *primary demands* and the set of primary demands will be denoted by P. In addition, we will use the overlap structure between demand neighborhoods to define a mapping that assigns each demand $\nu \in \overline{\mathbb{C}}$ to some primary demand $\kappa \in P$. As shown in the rounding algorithms in later sections, for each primary demand we guarantee exactly one open facility in its neighborhood, while for a non-primary demand, there is constant probability that none of its neighbors open. In this case we estimate its connection cost by the distance to the facility opened in its assigned primary demand's neighborhood. For this reason the connection cost of a primary demand must be "small" compared to the non-primary demands assigned to it. We also need sibling demands assigned to different primary demands for fault-tolerant requirements. Specifically, this partitioning will be constructed to satisfy a number of properties that are detailed below.

OLD VERSION .....

(P1) Vectors $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$ satisfy the following properties:

(a) $\sum_{\mu\in\overline{\mathbb{F}}} \bar{x}_{\mu\nu} = 1$ for each demand $\nu \in \overline{\mathbb{C}}$.

(b) $\sum_{\mu\in i,\nu\in j} \bar{x}_{\mu\nu} = x^*_{ij}$ for each site $i \in \mathbb{F}$ and client $j \in \mathbb{C}$.

(c) $\sum_{\mu\in i} \bar{y}_\mu = y^*_i$ for each site $i \in \mathbb{F}$.

(d) $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is complete, that is $\bar{x}_{\mu\nu} \neq 0$ implies $\bar{x}_{\mu\nu} = \bar{y}_\mu$, for all $\mu \in \overline{\mathbb{F}}, \nu \in \overline{\mathbb{C}}$.

(e) If $\nu, \nu' \in \overline{\mathbb{C}}$ are different siblings then $\overline{N}(\nu) \cap \overline{N}(\nu') = \emptyset$.

The first three conditions say that $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is a partition of $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, with unit-value demands.

(P2) Primary demands satisfy the following properties:

(a) For each site $i \in \mathbb{F}$, $\sum_{\mu\in i} \sum_{\kappa\in P} \bar{x}_{\mu\kappa} \leq y^*_i$. (Recall that $\bar{x}_{\mu\kappa} \neq 0$ iff $\mu \in \overline{N}(\kappa)$.)

(b) For any two different primary demands $\kappa, \kappa' \in P$ we have $\overline{N}(\kappa) \cap \overline{N}(\kappa') = \emptyset$.

(c) Each demand $\nu \in \overline{\mathbb{C}}$ is assigned to one primary demand $\kappa \in P$ with $\overline{N}(\nu) \cap \overline{N}(\kappa) \neq \emptyset$.

(d) Let ν, ν' be different siblings and let κ be the primary demand that ν is assigned to. Then $\overline{N}(\nu') \cap \overline{N}(\kappa) = \emptyset$. In particular, by Property P2(c), this implies that different sibling demands are assigned to different primary demands.

(e) If a demand $\nu \in \overline{\mathbb{C}}$ is assigned to a primary demand $\kappa \in P$, then $C^{\text{avg}}_\nu + \alpha^*_\nu \geq C^{\text{avg}}_\kappa + \alpha^*_\kappa$.

NEW VERSION .....

(PS) *Partitioned solution.* Vector $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is a partition of $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, with unit-value demands, that is:

1. $\sum_{\mu\in\overline{\mathbb{F}}} \bar{x}_{\mu\nu} = 1$ for each demand $\nu \in \overline{\mathbb{C}}$.
2. $\sum_{\mu\in i,\nu\in j} \bar{x}_{\mu\nu} = x^*_{ij}$ for each site $i \in \mathbb{F}$ and client $j \in \mathbb{C}$.
3. $\sum_{\mu\in i} \bar{y}_\mu = y^*_i$ for each site $i \in \mathbb{F}$.

(CO) *Completeness.* Solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is complete, that is $\bar{x}_{\mu\nu} \neq 0$ implies $\bar{x}_{\mu\nu} = \bar{y}_\mu$, for all $\mu \in \overline{\mathbb{F}}, \nu \in \overline{\mathbb{C}}$.

(PD) *Primary demands.* Primary demands satisfy the following conditions:

1. For any two different primary demands $\kappa, \kappa' \in P$ we have $\overline{N}(\kappa) \cap \overline{N}(\kappa') = \emptyset$.
2. For each site $i \in \mathbb{F}$, $\sum_{\mu\in i} \sum_{\kappa\in P} \bar{x}_{\mu\kappa} \leq y^*_i$. (Recall that $\bar{x}_{\mu\kappa} \neq 0$ iff $\mu \in \overline{N}(\kappa)$.)
3. Each demand $\nu \in \overline{\mathbb{C}}$ is assigned to one primary demand $\kappa \in P$ such that
   (a) $\overline{N}(\nu) \cap \overline{N}(\kappa) \neq \emptyset$, and
   (b) $C^{\text{avg}}_\nu + \alpha^*_\nu \geq C^{\text{avg}}_\kappa + \alpha^*_\kappa$.

(SI) *Siblings.* For any pair ν, ν' of different siblings we have

1. $\overline{N}(\nu) \cap \overline{N}(\nu') = \emptyset$.
2. If ν is assigned to a primary demand κ then $\overline{N}(\nu') \cap \overline{N}(\kappa) = \emptyset$. In particular, by Property PD(3(a)), this implies that different sibling demands are assigned to different primary demands.

As we shall demonstrate in later sections, these properties allow us to extend known UFL rounding algorithms to obtain an integral solution to our FTFP problem with a matching approximation ratio. Our partitioning is "adaptive" in the sense that it is constructed one demand at a time, and the connection values for the demands of a client depend on the choice of earlier demands, of this or other clients, and their connection values.

Implementation of Adaptive Partitioning We now describe an algorithm for partitioning the instance and the fractional solution so that the properties (PS), (CO), (PD), and (SI) are satisfied. Recall that $\overline{\mathbb{F}}$ and $\overline{\mathbb{C}}$, respectively, denote the sets of facilities and demands that will be created in this stage, and $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is the partitioned solution to be computed. At a high level, for each site i and client j we would split the x^*_{ij} value and give each piece to exactly one of its r_j demands. In the end we require each demand ν has its connection values with all facilities sum up to 1.

The adaptive partitioning algorithm consists of two phases: Phase 1 is called the partition phase and Phase 2 is called the augmenting phase. Phase 1 is done in iterations, where in each iteration we find the best client and create a new demand ν out of it. This demand either becomes a primary demand itself, or it is assigned to some existing primary demand. We call a client j *exhausted* when all its r_j demands have been created and assigned to some primary demands. Phase 1 completes when all clients are exhausted. We then do an augmenting phase 2 to ensure every demand has a neighborhood with total connection values equal to 1.

For each site i we will initially create one "big" facility μ with initial value $\bar{y}_\mu = y^*_i$. While we partition the instance, creating new demands and connections, this facility may end up being split into more facilities to preserve completeness of the fractional solution. Also, we will gradually decrease the fractional connection vector for each client j, to account for the demands created for j and their connection values. These decreased connection values will be stored in an auxiliary vector $\widetilde{\boldsymbol{x}}$. The intuition is that $\widetilde{\boldsymbol{x}}$ represents the part of $\boldsymbol{x}^*$ that still has not been partitioned into demands and future demands can use $\widetilde{\boldsymbol{x}}$ for their connections. For technical reasons, $\widetilde{\boldsymbol{x}}$ will be indexed by facilities (rather than sites) and clients, that is $\widetilde{\boldsymbol{x}} = (\widetilde{x}_{\mu j})$. At the beginning, we set $\widetilde{x}_{\mu j} \leftarrow x^*_{ij}$ for each $j \in \mathbb{C}$, where $\mu \in i$ is the single facility created initially at site i. At each step, whenever we create a new demand ν for a client j, we will define its values $\bar{x}_{\mu\nu}$ and appropriately reduce the values $\widetilde{x}_{\mu j}$, for all facilities μ. We will deal with two types of neighborhoods, with respect to $\widetilde{\boldsymbol{x}}$ and $\bar{\boldsymbol{x}}$, that is $\widetilde{N}(j) = \{\mu \in \overline{\mathbb{F}} : \widetilde{x}_{\mu j} > 0\}$ for $i \in \mathbb{C}$ and $\overline{N}(\nu) = \{\mu \in \overline{\mathbb{F}} : \bar{x}_{\mu\nu} > 0\}$ for $\nu \in \overline{\mathbb{C}}$. During this process we preserve the completeness of the fractional solutions $\widetilde{\boldsymbol{x}}$ and $\bar{\boldsymbol{x}}$. More precisely, the following properties will hold for every facility μ after every iteration:

(c1) For each demand ν either $\bar{x}_{\mu\nu} = 0$ or $\bar{x}_{\mu\nu} = \bar{y}_\mu$. This is the same condition as condition (CO), yet we repeat here as (c1) needs to hold after every iteration, while condition (CO) only applies to the final partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$.

(c2) For each client j, either $\widetilde{x}_{\mu j} = 0$ or $\widetilde{x}_{\mu j} = \bar{y}_\mu$.

A full description of the algorithm is given in Pseudocode 1. Initially, the set U of non-exhausted clients contains all clients, the set $\overline{\mathbb{C}}$ of demands is empty, the set $\overline{\mathbb{F}}$ of facilities consists of one facility μ on each site i with $\bar{y}_\mu = y^*_i$, and the set P of primary demands is empty (Lines 1–4). In one iteration of the while loop (Line 5), in Lines 6–8, for each client j we compute a quantity called $\mathrm{tcc}(j)$ (tentative connection cost), that represents the average distance from j to the set $\widetilde{N}_1(j)$ of the nearest facilities μ whose total connection value to j (the sum of $\widetilde{x}_{\mu j}$'s) equals 1. This set is computed by Procedure NearestUnitChunk() (see Pseudocode 2, Lines 1–9), which adds

facilities to $\widetilde{N}_1(j)$ in order of increasing distance, until the total connection value is exactly 1. (The procedure actually uses the $\bar{y}_\mu$ values, which are equal to the connection values by completeness.) This may require splitting the last added facility and adjusting the connection values so that conditions (c1) and (c2) are preserved.

Pseudocode 1 Algorithm: Adaptive Partitioning

Input: $\mathbb{F}$, $\mathbb{C}$, $(\boldsymbol{x}^*, \boldsymbol{y}^*)$
Output: $\overline{\mathbb{F}}$, $\overline{\mathbb{C}}$, $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ ▷ Unspecified $\bar{x}_{\mu\nu}$'s and $\widetilde{x}_{\mu j}$'s are assumed to be 0

1: $\widetilde{\boldsymbol{r}} \leftarrow \boldsymbol{r}, U \leftarrow \mathbb{C}, \overline{\mathbb{F}} \leftarrow \emptyset, \overline{\mathbb{C}} \leftarrow \emptyset, P \leftarrow \emptyset$ ▷ Phase 1
2: **for** each site $i \in \mathbb{F}$ **do**
3: create a facility μ at i and add μ to $\overline{\mathbb{F}}$
4: $\bar{y}_\mu \leftarrow y_i^*$ and $\widetilde{x}_{\mu j} \leftarrow x_{ij}^*$ for each $j \in \mathbb{C}$
5: **while** $U \neq \emptyset$ **do**
6: **for** each $j \in U$ **do**
7: $\widetilde{N}_1(j) \leftarrow$ NearestUnitChunk$(j, \overline{\mathbb{F}}, \widetilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ ▷ see Pseudocode 2
8: $\mathrm{tcc}(j) \leftarrow \sum_{\mu \in \widetilde{N}_1(j)} d_{\mu j} \cdot \widetilde{x}_{\mu j}$
9: $p \leftarrow \arg\min_{j \in U} \{\mathrm{tcc}(j) + \alpha_j^*\}$
10: create a new demand ν for client p
11: **if** $\widetilde{N}_1(p) \cap \overline{N}(\kappa) \neq \emptyset$ for some primary demand $\kappa \in P$ **then**
12: assign ν to κ
13: $\bar{x}_{\mu\nu} \leftarrow \widetilde{x}_{\mu p}$ and $\widetilde{x}_{\mu p} \leftarrow 0$ for each $\mu \in \widetilde{N}(p) \cap \overline{N}(\kappa)$
14: **else**
15: make ν primary, $P \leftarrow P \cup \{\nu\}$, assign ν to itself
16: set $\bar{x}_{\mu\nu} \leftarrow \widetilde{x}_{\mu p}$ and $\widetilde{x}_{\mu p} \leftarrow 0$ for each $\mu \in \widetilde{N}_1(p)$
17: $\overline{\mathbb{C}} \leftarrow \overline{\mathbb{C}} \cup \{\nu\}, \widetilde{r}_p \leftarrow \widetilde{r}_p - 1$
18: **if** $\widetilde{r}_p = 0$ **then** $U \leftarrow U \setminus \{p\}$
19: **for** each client $j \in \mathbb{C}$ **do** ▷ Phase 2
20: **for** each demand $\nu \in j$ **do** ▷ each client j has r_j demands
21: **if** $\sum_{\mu \in \overline{N}(\nu)} \bar{x}_{\mu\nu} < 1$ **then** AugmentToUnit$(\nu, j, \overline{\mathbb{F}}, \widetilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ ▷ see Pseudocode 2

The next step is to pick a client p with minimum $\mathrm{tcc}(p) + \alpha_p^*$ and create a demand ν for p (Lines 9–10). If $\widetilde{N}_1(p)$ overlaps the neighborhood of some existing primary demand κ (if there are multiple such κ's, pick any of them), we assign ν to κ, and ν acquires all the connection values $\widetilde{x}_{\mu p}$ between client p and facility μ in $\widetilde{N}(p) \cap \overline{N}(\kappa)$ (Lines 11–13). Note that although we check for overlap with $\widetilde{N}_1(p)$, we then move all facilities in the intersection with $\widetilde{N}(p)$, a bigger set, into $\overline{N}(\nu)$. The other case is when $\widetilde{N}_1(p)$ is disjoint from the neighborhoods of all existing primary demands. Then, in Lines 15–16, ν becomes itself a primary demand and we assign ν to itself. It also inherits the connection values to all facilities $\mu \in \widetilde{N}_1(p)$ from p (recall that $\widetilde{x}_{\mu p} = \bar{y}_\mu$), with all other $\bar{x}_{\mu\nu}$ values set to 0.

At this point all primary demands satisfy Property PS(1), but this may not be true for non-primary demands. For those demands we still may need to adjust the $\bar{x}_{\mu\nu}$ values so that $\mathrm{conn}(\nu) \stackrel{\text{def}}{=} \sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu}$ equals 1. This is accomplished by Procedure AugmentToUnit() (definition in Pseudocode 2, Lines 10–20) that allocates to $\nu \in j$ some of the remaining connection values

Pseudocode 2 Helper functions used in Pseudocode 1

function NEARESTUNITCHUNK($j, \overline{\mathbb{F}}, \widetilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$) ▷ upon return, $\sum_{\mu\in\widetilde{N}_1(j)} \widetilde{x}_{\mu j} = 1$
  Let $\widetilde{N}(j) = \{\mu_1, ..., \mu_{n_j}\}$ where $d_{\mu_1 j} \le d_{\mu_2 j} \le \ldots \le d_{\mu_{n_j} j}$
  Let l be such that $\sum_{k=1}^{l} \bar{y}_{\mu_k} \ge 1$ and $\sum_{k=1}^{l-1} \bar{y}_{\mu_k} < 1$
  Create a new facility σ at the same site as μ_l and add it to $\overline{\mathbb{F}}$ ▷ split μ_l
  Set $\bar{y}_\sigma \leftarrow \sum_{k=1}^{l} \bar{y}_{\mu_k} - 1$ and $\bar{y}_{\mu_l} \leftarrow \bar{y}_{\mu_l} - \bar{y}_\sigma$
  For each $\nu \in \mathbb{C}$ with $\bar{x}_{\mu_l\nu} > 0$ set $\bar{x}_{\mu_l\nu} \leftarrow \bar{y}_{\mu_l}$ and $\bar{x}_{\sigma\nu} \leftarrow \bar{y}_\sigma$
  For each $j' \in \mathbb{C}$ with $\widetilde{x}_{\mu_l j'} > 0$ (including j) set $\widetilde{x}_{\mu_l j'} \leftarrow \bar{y}_{\mu_l}$ and $\widetilde{x}_{\sigma j'} \leftarrow \bar{y}_\sigma$
  (All other new connection values are set to 0)
  return $\widetilde{N}_1(j) = \{\mu_1, \ldots, \mu_{l-1}, \mu_l\}$
function AUGMENTTOUNIT($\nu, j, \overline{\mathbb{F}}, \widetilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$) ▷ ν is a demand of client j
  while $\sum_{\mu\in\overline{\mathbb{F}}} \bar{x}_{\mu\nu} < 1$ **do** ▷ upon return, $\sum_{\mu\in\overline{N}(\nu)} \bar{x}_{\mu\nu} = 1$
    Let η be any facility such that $\widetilde{x}_{\eta j} > 0$
    if $1 - \sum_{\mu\in\overline{\mathbb{F}}} \bar{x}_{\mu\nu} \ge \widetilde{x}_{\eta j}$ **then**
      $\bar{x}_{\eta\nu} \leftarrow \widetilde{x}_{\eta j}, \widetilde{x}_{\eta j} \leftarrow 0$
    else
      Create a new facility σ at the same site as η and add it to $\overline{\mathbb{F}}$ ▷ split η
      Let $\bar{y}_\sigma \leftarrow 1 - \sum_{\mu\in\overline{\mathbb{F}}} \bar{x}_{\mu\nu}, \bar{y}_{\mu^*} \leftarrow \bar{y}_\eta - \bar{y}_\sigma$
      Set $\bar{x}_{\sigma\nu} \leftarrow \bar{y}_\sigma,\ \bar{x}_{\eta\nu} \leftarrow 0,\ \widetilde{x}_{\eta j} \leftarrow \bar{y}_\eta,\ \widetilde{x}_{\sigma j} \leftarrow 0$
      For each $\nu' \ne \nu$ with $\bar{x}_{\eta\nu'} > 0$ set $\bar{x}_{\eta\nu'} \leftarrow \bar{y}_\eta,\ \bar{x}_{\sigma\nu'} \leftarrow \bar{y}_\sigma$
      For each $j' \ne j$ with $\widetilde{x}_{\eta j'} > 0$ set $\widetilde{x}_{\eta j'} \leftarrow \bar{y}_\eta, \widetilde{x}_{\sigma j'} \leftarrow \bar{y}_\sigma$

$\widetilde{x}_{\mu j}$ of client j (Lines 19–21). AUGMENTTOUNIT() will repeatedly pick any μ with $\widetilde{x}_{\mu j} > 0$. If $\widetilde{x}_{\mu j} \le 1 - \text{conn}(\nu)$, then the connection value $\widetilde{x}_{\mu j}$ is reassigned to ν. Otherwise, $\widetilde{x}_{\mu j} > 1 - \text{conn}(\nu)$, in which case we split μ so that connecting ν to one of the created copies of μ will make $\text{conn}(\nu)$ equal 1, and we'll be done.

Notice that we start with $|\mathbb{F}|$ facilities and in each iteration each client causes at most one split. We have a total of no more than $R|\mathbb{C}|$ iterations as in each iteration we create one demand. (Recall that $R = \max_j r_j$.) So the total number of facilities we created will be at most $|\mathbb{F}| + R|\mathbb{C}|^2$, which is polynomial in $|\mathbb{F}| + |\mathbb{C}|$ due to our earlier bound on R.

Correctness. We now show that all the required properties (PS), (CO), (PD) and (SI) are satisfied by the above construction.

Properties (PS) and (CO) follow directly from the algorithm. (CO) is implied by the completeness condition (c1) that the algorithm maintains after each iteration. PS(1) follows from the result of calling Procedure AUGMENTTOUNIT() in Line 21. To see PS(2), at each step the algorithm maintains an invariant that, for every $i \in \mathbb{F}$ and $j \in \mathbb{C}$, we have $\sum_{\mu\in i}\sum_{\nu\in j} \bar{x}_{\mu\nu} + \sum_{\mu\in i} \widetilde{x}_{\mu j} = x^*_{ij}$. In the end, we will create r_j demands for each client j, with each demand $\nu \in j$ satisfying PS(1), and thus $\sum_{\nu\in j}\sum_{\mu\in\overline{\mathbb{F}}} \bar{x}_{\mu\nu} = r_j$. This implies that $\widetilde{x}_{\mu j} = 0$ for every facility $\mu \in \overline{\mathbb{F}}$, and PS(2) follows. PS(3) follows from that every time we split a facility μ into μ' and μ'', the sum of $\bar{y}_{\mu'}$ and $\bar{y}_{\mu''}$ is equal to the old value $\bar{y}_\mu$.

Now we deal with properties in group (PD). PD(1) follows directly from the algorithm since we only create a new primary demand if its neighborhood is disjoint from all existing primary demands and its neighborhood is fixed to $\widetilde{N}_1(p)$ immediately. PD(2) follows from the fact that

neighborhoods of primary demands are disjoint and property (CO). One way to see this is that the double summation is the same as summing over a subset of facilities split from site i. PD(3(a)) follows from the way assignment is done by the algorithm. When demand ν of client p is assigned to a primary demand κ in Lines 11–13 of Pseudocode 1, we move all facilities in $\widetilde{N}(p) \cap \overline{N}(\kappa)$ (the intersection is nonempty) into $\overline{N}(\nu)$, and we never remove a facility from $\overline{N}(\nu)$. We postpone the proof of PD(3(b)) to Lemma 5.

Finally we argue properties in group (SI) hold. SI(1) is easy, since each facility μ is added to at most one sibling's neighborhood by setting $\bar{x}_{\mu\nu}$ to $\bar{y}_\mu$ while other siblings ν' have $\bar{x}_{\mu\nu'} = 0$. Note that right after a demand $\nu \in p$ is created, its neighborhood is disjoint from the neighborhood of p, that is $\overline{N}(\nu) \cap \widetilde{N}(p) = \emptyset$, by Lines 11–13 of the algorithm. Thus all demands of p created later will have neighborhoods disjoint from $\overline{N}(\nu)$. Furthermore, Procedure AugmentToUnit() preserves this property, because when it adds an existing facility to $\overline{N}(\nu)$ then it removes it from $\widetilde{N}(p)$, and in case of splitting, one resulting facility is added to $\overline{N}(\nu)$ and the other to $\widetilde{N}(p)$. For the proof of SI(2), we defer to Lemma 3.

It remains to show Properties PD(3(b)) and SI(2). We show them in the lemmas below, thus completing the description of our adaptive partition process.

Lemma 3. *Property SI(2) holds after the Adaptive Partitioning stage.*

Proof. Let $\nu_1, \ldots, \nu_{r_j}$ be the demands of a client $j \in \mathbb{C}$, listed in the order of creation, and, for each $q = 1, 2, \ldots, r_j$, denote by κ_q the primary demand that ν_q is assigned to. We claim that, for every $q = 1, \ldots, r_j$, at the time when ν_q is about to be created, we have

$$\widetilde{N}(j) \cap \bigcup_{s=1}^{q-1} \overline{N}(\nu_s) \;=\; \emptyset \;(*) \quad \text{and} \quad \widetilde{N}(j) \cap \bigcup_{s=1}^{q-1} \overline{N}(\kappa_s) \;=\; \emptyset \;(**).$$

Indeed, this follows by an easy induction on q, because when we create ν_q, the neighborhood $\overline{N}(\nu_q)$ inherits all facilities in $\widetilde{N}(j) \cap \overline{N}(\kappa_q)$, which are then removed from $\widetilde{N}(j)$. (Equation (*) above was in fact already indirectly shown when we proved Property SI(1).) This implies that, for $s < q$, $\overline{N}(\nu_q)$ is disjoint from $\overline{N}(\kappa_s)$ and $\overline{N}(\kappa_q)$ is disjoint from $\overline{N}(\nu_s)$. In the augmenting phase 2, when we add facilities using AugmentToUnit() in Line 19–21 of Pseudocode 1, the neighborhoods of all primary demands, including $\overline{N}(\kappa_1), \ldots, \overline{N}(\kappa_{r_j})$, have already been fixed. Further, these added facilities in phase 2 do not appear in any neighborhood $\overline{N}(\kappa_1), \ldots, \overline{N}(\kappa_{r_j})$ and each of these facilities is added to the neighborhood of exactly one sibling. Therefore all the disjointness conditions are preserved after the augment phase, proving Property SI(2). □

We need one lemma before proving our last property PD(3(b)). For a client j and a demand ν, we use notation $\mathrm{tcc}_\nu(j)$ for the value of $\mathrm{tcc}(j)$ at the time when ν was created. (It is not necessary that $\nu \in j$ but we assume that j is not exhausted at that time.)

Lemma 4. *Let η and ν be two demands, with η created earlier than ν, and let $j \in \mathbb{C}$ be a client that is not exhausted when ν is created. Then we have*

(a) $tcc_\eta(j) \le tcc_\nu(j)$, and

(b) if $\nu \in j$ then $tcc_\eta(j) \le C_\nu^{\mathrm{avg}}$.

Proof. We focus on the time when η is about to be created, after the call to NearestUnitChunk() in Pseudocode 1, Line 7. Let $\widetilde{N}(j) = \{\mu_1, ..., \mu_q\}$ with all facilities μ_s ordered according to nondecreasing distance from j. Consider the following linear program:

$$\begin{aligned} \text{minimize} \quad & \sum\nolimits_s d_{\mu_s j} z_s \\ \text{subject to} \quad & \sum\nolimits_s z_s \geq 1 \\ & 0 \leq z_s \leq \widetilde{x}_{\mu_s j} \quad \text{for all } s \end{aligned}$$

This is a fractional minimum knapsack covering problem (with knapsack size equal 1) and its fractional solution is the greedy solution, which is exactly $\text{tcc}_\eta(j)$. On the other hand, we claim that $\text{tcc}_\nu(j)$ and C_ν^{avg} both represent feasible solutions to this linear program. Indeed, each of these quantities involves some later values $\widetilde{x}_{\mu j}$, where μ could be one of the facilities μ_s or a new facility resulted from splitting. For each distance $d_{\mu_s j}$, however, the sum of all $\widetilde{x}_{\mu j}$, for all facilities that were split from μ_s, cannot exceed the value $\widetilde{x}_{\mu_s j}$ at the time when η was created, because splitting facilities preserves this sum and creating new demands for j can only decrease it. Therefore both quantities $\text{tcc}_\nu(j)$ and C_ν^{avg} correspond to some choice of the z_s variables (adding up to 1), and the lemma follows. □

Lemma 5. *Property PD(3(b)) holds after the Adaptive Partitioning stage.*

Proof. Suppose that demand $\nu \in j$ is assigned to some primary demand $\kappa \in p$. Then

$$C_\kappa^{\text{avg}} + \alpha_\kappa^* \;=\; \text{tcc}_\kappa(p) + \alpha_p^* \;\leq\; \text{tcc}_\kappa(j) + \alpha_j^* \;\leq\; C_\nu^{\text{avg}} + \alpha_\nu^*.$$

We now justify this derivation. By definition we have $\alpha_\kappa^* = \alpha_p^*$. Further, by the algorithm, if κ is a primary demand of client p, then C_κ^{avg} is equal to $\text{tcc}(p)$ computed when κ is created, which is exactly $\text{tcc}_\kappa(p)$. Thus the first equation is true. The first inequality follows from the choice of p as the primary demand in Line 9 Pseudocode 1. The last inequality holds because $\alpha_j^* = \alpha_\nu^*$ (due to $\nu \in j$), and because $\text{tcc}_\kappa(j) \leq C_\nu^{\text{avg}}$, where this inequality is tight for $\nu = \kappa$ (and $j = p$) and it follows from Lemma 4 if $\nu \neq \kappa$. □

We have now shown all properties do hold for our partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$. In the following sections we give the generalization of three algorithms for UFL, achieving approximation ratio 3, $1 + 2/e = 1.736$ and 1.575 respectively. The last ratio is also the currently best known ratio using only LP-rounding algorithm for UFL.

5 Algorithm EGUP with Ratio 3

With the partitioned FTFP instance and its associated fractional solution in place, we now begin to introduce our rounding algorithms which are extensions from those for the UFL problems.

The algorithm we describe in this section achieves ratio 3. Although this is still quite far from our best ratio 1.575 that we derive later, we include this algorithm in the paper to show, in a relatively simple setting, how the properties of our partitioned fractional solution are used in rounding it to an integral solution. The rounding approach we use here is an extension of the corresponding method for UFL described by [8].

Algorithm EGUP**.** At a high level, we would open exactly one facility for each primary demand κ, and each non-primary demand is connected to the facility opened for the primary demand it was assigned to.

More precisely we apply a rounding process, guided by the fractional values $(\bar{y}_\mu)$ and $(\bar{x}_{\mu\nu})$, that produces an integral solution. This integral solution is obtained by choosing a subset of facilities in $\overline{\mathbb{F}}$ to open, and for each demand in $\overline{\mathbb{C}}$, specifying an open facility that this demand will be connected to. For each primary demand $\kappa \in P$, we want to open one facility $\phi(\kappa) \in \overline{N}(\kappa)$. To this end, we use randomization: for each $\mu \in \overline{N}(\kappa)$, we choose $\phi(\kappa) = \mu$ with probability $\bar{x}_{\mu\kappa}$, ensuring that exactly one $\mu \in \overline{N}(\kappa)$ is chosen. Note that $\sum_{\mu\in\overline{N}(\kappa)} \bar{x}_{\mu\kappa} = 1$, so this distribution is well-defined. We open this facility $\phi(\kappa)$ and connect to $\phi(\kappa)$ all demands that are assigned to κ.

In our description above, the algorithm is presented as a randomized algorithm. It can be derandomized using the method of conditional expectations, which is commonly used in approximation algorithms for facility location problems and standard enough that presenting it here would be redundant. Readers less familiar with this field are recommended to consult [5], where the method of conditional expectations is applied in a context very similar to ours.

Analysis. We now bound the expected facility cost and connection cost by establishing the two lemmas below.

Lemma 6. *The expectation of facility cost F_{EGUP} of our solution is at most F^*.*

Proof. By Property PD(1), the neighborhoods of primary demands are disjoint. Also, for any primary demand $\kappa \in P$, the probability that a facility $\mu \in \overline{N}(\kappa)$ is chosen as the open facility $\phi(\kappa)$ is $\bar{x}_{\mu\kappa}$. Hence the expected total facility cost is

$$\begin{aligned}\text{Exp}[F_{\text{EGUP}}] &= \textstyle\sum_{\kappa\in P}\sum_{\mu\in\overline{N}(\kappa)} f_\mu \bar{x}_{\mu\kappa} \\ &= \textstyle\sum_{\kappa\in P}\sum_{\mu\in\overline{\mathbb{F}}} f_\mu \bar{x}_{\mu\kappa} \\ &= \textstyle\sum_{i\in\mathbb{F}} f_i \sum_{\mu\in i}\sum_{\kappa\in P} \bar{x}_{\mu\kappa} \\ &\le \textstyle\sum_{i\in\mathbb{F}} f_i y_i^* = F^*,\end{aligned}$$

where the inequality follows from Property PD(2). □

Lemma 7. *The expectation of connection cost C_{EGUP} of our solution is at most $C^* + 2 \cdot \text{LP}^*$.*

Proof. Consider a demand ν assigned to a primary demand $\kappa \in P$. Let μ be any facility in $\overline{N}(\nu) \cap \overline{N}(\kappa)$. Since μ is in both $\overline{N}(\nu)$ and $\overline{N}(\kappa)$, we have $d_{\mu\nu} \le \alpha_\nu^*$ and $d_{\mu\kappa} \le \alpha_\kappa^*$ (This follows from the complementary slackness conditions since $\alpha_\nu^* = \beta_{\mu\nu}^* + d_{\mu\nu}$ for each $\mu \in \overline{N}(\nu)$.). Thus, applying the triangle inequality, for any fixed choice of facility $\phi(\kappa)$ we have

$$d_{\phi(\kappa)\nu} \le d_{\phi(\kappa)\kappa} + d_{\mu\kappa} + d_{\mu\nu} \le d_{\phi(\kappa)\kappa} + \alpha_\kappa^* + \alpha_\nu^*.$$

Therefore the expected distance from ν to its facility $\phi(\kappa)$ is

$$\begin{aligned}\text{Exp}[d_{\phi(\kappa)\nu}] &\le C_\kappa^{\text{avg}} + \alpha_\kappa^* + \alpha_\nu^* \\ &\le C_\nu^{\text{avg}} + \alpha_\nu^* + \alpha_\nu^* = C_\nu^{\text{avg}} + 2\alpha_\nu^*,\end{aligned}$$

where the second inequality follows from Property PD(3(b)). From the definition of C_ν^{avg} and Property PS(2), for any $j \in \mathbb{C}$ we have

$$\begin{aligned}\sum\nolimits_{\nu\in j} C_\nu^{\text{avg}} &= \sum\nolimits_{\nu\in j}\sum\nolimits_{\mu\in\overline{\mathbb{F}}} d_{\mu\nu}\bar{x}_{\mu\nu} \\ &= \sum\nolimits_{i\in\mathbb{F}} d_{ij}\sum\nolimits_{\nu\in j}\sum\nolimits_{\mu\in i}\bar{x}_{\mu\nu} \\ &= \sum\nolimits_{i\in\mathbb{F}} d_{ij}x^*_{ij} = C^*_j.\end{aligned}$$

Thus, summing over all demands, the expected total connection cost is

$$\begin{aligned}\text{Exp}[C_{\text{EGUP}}] &\le \sum\nolimits_{j\in\mathbb{C}}\sum\nolimits_{\nu\in j}(C_\nu^{\text{avg}} + 2\alpha^*_\nu) \\ &= \sum\nolimits_{j\in\mathbb{C}}(C^*_j + 2r_j\alpha^*_j) = C^* + 2\cdot\text{LP}^*,\end{aligned}$$

completing the proof of the lemma. □

Theorem 8. *Algorithm* EGUP *is a* 3*-approximation algorithm.*

Proof. By Property SI(2), different demands from the same client are assigned to different primary demands, and by PD(1) each primary demand opens a different facility. This ensures that our solution is feasible, namely each client j is connected to r_j different facilities (some possibly located on the same site). As for the total cost, Lemma 6 and Lemma 7 imply that the total cost is at most $F^* + C^* + 2\cdot\text{LP}^* = 3\cdot\text{LP}^* \le 3\cdot\text{OPT}$. □

This completes the presentation and analysis of our 3-approximation algorithm EGUP.

6 Algorithm ECHU with Ratio 1.736

In this section we improve the approximation ratio to $1 + 2/e \approx 1.736$. The new algorithm, named Algorithm ECHU, starts with the same partitioned fracitional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ as Algorithm EGUP. The improvement on approximation ratio comes from a more slightly modified rounding process and much refined analysis. Note that the facility opening cost of Algorithm EGUP does not exceed that of the fractional optimum solution, while the connection cost is quite far from the optimum, due to the cost of indirect connections (that is, connections from non-primary demands). The basic idea behind the improvement, following the approach of Chudak and Shmoys' [5], is then to open more facilities and use direct connections when available so as to reduce the number of indirect connections. A naïve generalization of the method from [5] may produce infeasible solutions, with different demands from the same client being connected to the same facility. This could happen, for example, when one demand of a client connects to a facility in its neighborhood while another uses the same facility via an indirect connection. We show, however, that the fractional solution we obtained from Adaptive Partitioning in the last section possesses additional properties that allows us to prevent such conflicts.

The details of the modified rounding process is given in Pseudocode 3. We will use the term *facility cluster* for the neighborhood of a primary client. Facilities that do not belong to these clusters will be called *non-clustered*.

As before, we open exactly one facility $\phi(\kappa)$ in the facility cluster of a primary demand κ, with the same probability distribution as before (Line 2). For any non-primary demand ν assigned to

κ, we refer to $\phi(\kappa)$ as the *target* facility of ν. In Algorithm EGUP, ν was connected to $\phi(\kappa)$, but in Algorithm ECHU we may open a facility in ν's neighborhood and connect ν to this facility. Specifically, the two changes in the algorithm are as follows: (1) Each non-clustered facility μ is opened, independently, with probability $\bar{y}_\mu$ (Lines 4–5). Notice that due to completeness of the partitioned fractional solution, we have $\bar{y}_\mu = \bar{x}_{\mu\nu}$ for some demand ν, and $\sum_{\mu\in\overline{\mathbb{F}}} \bar{x}_{\mu\nu} = 1$, therefore $\bar{y}_\mu \leq 1$. (2) When connecting demands to facilities, a primary demand κ is connected to the only facility $\phi(\kappa)$ opened in its neighborhood, as before (Line 3). For a non-primary demand ν, if its neighborhood has an open facility, we connect ν to the closest open facility in its neighborhood (Line 8). Otherwise, we connect ν to its target facility (Line 10).

Pseudocode 3 Algorithm ECHU, Stage 2: Constructing Integral Solution

```
1:  for each κ ∈ P do
2:      choose one φ(κ) ∈ N̄(κ), with each μ ∈ N̄(κ) chosen as φ(κ) with probability ȳ_μ
3:      open φ(κ) and connect κ to φ(κ)
4:  for each μ ∈ C̄ − ⋃_{κ∈P} N̄(κ) do
5:      open μ with probability ȳ_μ (independently)
6:  for each non-primary demand ν ∈ C̄ do
7:      if any facility in N̄(ν) is open then
8:          connect ν to the nearest open facility in N̄(ν)
9:      else
10:         connect ν to φ(κ) where κ is ν's primary demand
```

Analysis. We shall first argue that the integral solution thus constructed is feasible, and then we bound the total cost of the solution. Regarding feasibility, the only constraint that is not explicitly enforced by the algorithm is the fault-tolerance requirement; namely that each client j is connected to r_j different facilities. Thus it is sufficient to prove the following lemma.

Lemma 9. *For each client $j \in \mathbb{C}$, all demands from j are connected to different facilities.*

Proof. Let ν_1, ν_2 be two sibling demands and κ_1, κ_2 be the primary demands they are assigned to, respectively. Property P1(e) and P2(d) imply that

$$(\overline{N}(\kappa_1) \cup \overline{N}(\nu_1)) \cap \overline{N}(\nu_2) = \emptyset.$$

Therefore each facility is accessible to at most one demand among siblings that are created from the same client. □

We now show the solution has a total cost bounded by $(1 + 2/e) \cdot \mathrm{LP}^*$ in expectation. By Property P2(b), every facility may appear in at most one primary demand's neighborhood, and the facilities open in Line 4–5 of Pseudocode 3 do not appear in any primary demand's neighborhood. Therefore, by linearity of expectation, the expected facility cost of algorithm ECHU is $\sum_{\mu\in\overline{\mathbb{F}}} f_\mu \bar{y}_\mu = \sum_{i\in\mathbb{F}} f_i \sum_{\mu\in i} \bar{y}_\mu = \sum_{i\in\mathbb{F}} f_i y_i^* = F^*$, where the second equality follows from Property P1(c).

To bound the connection cost, we adapt an argument of Chudak and Shmoys' [5]. Consider a demand ν. This demand can either get connected directly to some facility in $\overline{N}(\nu)$ or indirectly to facility $\phi(\kappa) \in \overline{N}(\kappa)$, where κ is the primary demand to which ν is assigned.

We first estimate the expected cost of the indirect connection of ν, when ν is connected to its target facility $\phi(\kappa)$, where κ is the primary demand that ν is assigned to and $\phi(\kappa)$ is the only facility opened by κ. We have the following lemma.

Lemma 10. *The expected cost of an indirect connection of a demand ν to its target facility is bounded by $C^{\text{avg}}(\nu) + 2\alpha^*_\nu$.*

Proof. Let κ be the primary demand that ν is assigned to and $\phi(\kappa)$ be the only facility opened by κ. Then $\phi(\kappa)$ is the target facility of ν. By the definition of κ, there exists a facility $\mu \in \overline{N}(\nu) \cap \overline{N}(\kappa)$. For any fixed value of $\phi(\kappa)$, the cost of connecting ν to $\phi(\kappa)$ is $\bar{d} = d(\phi(\kappa), \nu) \le d(\phi(\kappa), \kappa) + d_{\mu\kappa} + d_{\mu\nu} \le d(\phi(\kappa), \kappa) + \alpha^*_\kappa + \alpha^*_{\nu'}$, where the last inequality follows from the complementary slackness conditions. We can bound the expectation of $\bar{d}$ as follows:

$$\text{Exp}[\bar{d}] = \text{Exp}[d(\phi(\kappa), \kappa)] + \alpha^*_\kappa + \alpha^*_\nu \le \text{tcc}(\kappa) + \alpha^*_\kappa + \alpha^*_\nu \le \text{tcc}(\nu) + 2\alpha^*_\nu \;\le\; C^{\text{avg}}_\nu + 2\alpha^*_\nu. \qquad (3)$$

The second inequality follows from Property P2(e) and the last inequality is from Lemma 4. To see the first inequality, notice that the expectation is computed for facility set $X(\kappa) = \overline{N}(\kappa) \setminus \overline{N}(\nu)$ with each facility μ chosen with probability proportional to $\bar{y}_\mu$. Given a facility set A, let $d(A, \nu) = \sum_{\mu \in A} d_{\mu\nu}\bar{y}_\mu / \sum_{\mu \in A} \bar{y}_\mu$. Notice that $\text{tcc}(\kappa) = d(\overline{N}(\kappa), \kappa)$ by definition. We claim that $d(X(\kappa), \nu) \le \text{tcc}(\kappa) + \alpha^*_\kappa + \alpha^*_\nu$. To prove this claim, we consider two cases.

Case 1: The first case is when there exists some $\mu' \in \overline{N}(\kappa) \cap \overline{N}(\nu)$ such that $d_{\mu'\kappa} \le d(\overline{N}(\kappa), \kappa)$. In this case, denoting $d_{\kappa\nu} = \min_{\mu \in \overline{\mathbb{F}}}(d_{\mu\kappa} + d_{\mu\nu})$, we have

$$d(\kappa, \nu) \le d_{\mu'\kappa} + d_{\mu'\nu} \le d(\overline{N}(\kappa), \kappa) + \alpha^*_\nu = \text{tcc}(\kappa) + \alpha^*_\nu.$$

It follows that for every $\mu \in X(\kappa)$, we have

$$d(\mu, \nu) \le d_{\mu\kappa} + d(\kappa, \nu) \le \alpha^*_\kappa + \text{tcc}(\kappa) + \alpha^*_\nu.$$

Case 2: In the second case, every $\mu' \in \overline{N}(\nu) \cap \overline{N}(\kappa)$ has $d_{\mu'\kappa} > d(\overline{N}(\kappa), \kappa)$. Then we have $d(X(\kappa), \kappa) \le d(\overline{N}(\kappa), \kappa) = \text{tcc}(\kappa)$, therefore

$$d(X(\kappa), \nu) \le d(\overline{N}(\kappa), \kappa) + d_{\mu\kappa} + d_{\mu\nu} \le \text{tcc}(\kappa) + \alpha^*_\kappa + \alpha^*_\nu.$$

This completes the justification of (3). □

We now estimate the expected connection cost of demand ν. Let $\overline{N}(\nu) = \{\mu_1, \ldots, \mu_l\}$ and let $d_s = d_{\mu_s\nu}$ for $s = 1, \ldots, l$. By reordering, we can assume that $d_1 \le d_2 \le \ldots \le d_l$. By the algorithm the connection cost is no more than that obtained through the random process that opens each $\mu_s \in \overline{N}(\nu)$ independently with probability $\bar{x}_{\mu_s\nu} = \bar{y}_{\mu_s}$ (because the solution is complete) and connects ν to the nearest such open facility, if any of them opens; otherwise ν is connected indirectly to $\phi(\kappa)$ at cost $\bar{d}$. The intuition is that we only use a facility μ_s if none of $\mu_1, \ldots, \mu_{s-1}$ is open. Suppose μ_s is a facility in $\overline{N}(\kappa)$ for some primary demand κ. If none of $\mu_1, \ldots, \mu_{s-1}$ is in $\overline{N}(\kappa)$, then μ_s opens with probability $\bar{y}_{\mu_s}$. If some of them do appear in $\overline{N}(\kappa)$, the fact that they are closed can only increase the probability that μ_s opens, by the choice of $\phi(\kappa)$. For a detailed proof, see [5].

Denoting by C_ν the connection cost for ν, we thus have

$$\begin{aligned}\mathrm{Exp}[C_\nu] &\le d_1\bar{y}_{\mu_1} + d_2\bar{y}_{\mu_2}(1-\bar{y}_{\mu_1}) + \ldots + d_l\bar{y}_{\mu_l}\textstyle\prod_{s=1}^{l-1}(1-\bar{y}_{\mu_s}) + \mathrm{Exp}[\bar{d}]\prod_{s=1}^{l}(1-\bar{y}_{\mu_s}) \\ &\le (1-\textstyle\prod_{i=1}^{l}(1-\bar{y}_{\mu_i}))\sum_{i=1}^{l} d_i\bar{y}_{\mu_i} + \prod_{i=1}^{l}(1-\bar{y}_{\mu_i})\mathrm{Exp}[\bar{d}] \\ &\le (1-\frac{1}{e})\textstyle\sum_{i=1}^{l} d_i\bar{y}_i + \frac{1}{e}\mathrm{Exp}[\bar{d}] \le (1-\frac{1}{e})C_\nu^{\mathrm{avg}} + \frac{1}{e}(C_\nu^{\mathrm{avg}} + 2\alpha_\nu^*) = C_\nu^{\mathrm{avg}} + \frac{2}{e}\alpha_\nu^*,\end{aligned}$$

where the second inequality was shown in [5]. (In the appendix we give a simpler proof of this inequality using only elementary techniques.) Notice that the completeness of the fractional solution allows us to write $\bar{y}_\mu$ instead of $\bar{x}_{\mu\nu}$.

Summing over all demands of a client j, we obtain the expected connection cost of client j:

$$\mathrm{Exp}[C_j] \le \textstyle\sum_{\nu\in j}(C_\nu^{\mathrm{avg}} + \frac{2}{e}\alpha_\nu^*) = C_j^* + \frac{2}{e}r_j\alpha_j^*.$$

Summing over all clients j gives the expected connection cost being bounded by $C^* + \frac{2}{e}\mathrm{LP}^*$. Therefore, we have established that our algorithm constructs a feasible integral solution with an overall expected cost bounded by

$$F^* + C^* + \frac{2}{e}\cdot \mathrm{LP}^* = (1+2/e)\cdot \mathrm{LP}^* \le (1+2/e)\cdot \mathrm{OPT}.$$

Summarizing, we obtain the result of this section.

Theorem 11. *Algorithm* ECHU *is a* $(1+2/e)$*-approximation algorithm for* FTFP.

7 Algorithm EBGS with Ratio 1.575

In this section we give our main result, a 1.575-approximation algorithm for FTFP [2]. This ratio matches the ratio of the best known LP-rounding algorithm for UFL [3]. Our EBGS algorithm consists of two stages, with stage 1 being a more refined partitioning of a fractional optimal solution, which allows us to apply a similar rounding process and analysis as [3].

Recall that in Section 6, we have an integral solution with facility cost bounded by F^* and connection cost bounded by $C^* + 2/e\cdot \mathrm{LP}^*$. A natural idea is to look for a way to reduce the connection cost, at the expense of increasing the facility cost by a small fraction.

To help motivate our improved algorithm, we start by giving an overview of the related BGS algorithm for UFL in [3]. To help introduce our notation, we present the BGS algorithm in a slightly different but equivalent way. The BGS algorithm uses some constant $\gamma \ge 1$, to be fixed later. As usual, it first computes a primal fractional optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to LP (1). Notice that for the UFL problem we have all $r_j = 1$. By splitting facilities and adjusting appropriately facility opening and connection values, we can assume that solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ satisfies the enhanced complete property: For each client j and facility i, $x_{ij}^* > 0$ implies $x_{ij}^* = y_i^*$. Moreover, for each client j, if we order facilities i in its neighborhood with nondecreasing d_{ij}, then there exists some l such that $\sum_{i=1}^{l} x_{ij}^* = 1/\gamma$. As before we also have $\sum_{i\in N(j)} x_{ij}^* = 1$ from optimality, where $N(j) = \{i \in \mathbb{F} : x_{ij}^* > 0\}$ is client j's neighborhood. This enhanced complete property allow us to introduce two more definitions.

[2] 1.575 is $\min_{\gamma\ge 1}\max\{\gamma, 1+2/e^\gamma, \frac{1/e+1/e^\gamma}{1-1/\gamma}\}$.

For each client j, its neighborhood $N(j)$ is partitioned into two parts, the close and the far neighborhood, that is $N(j) = N_{\text{cls}}(j) \cup N_{\text{far}}(j)$, where (i) $N_{\text{cls}}(j) \cap N_{\text{far}}(j) = \emptyset$, (ii) $\sum_{i \in N_{\text{cls}}(j)} x^*_{ij} = 1/\gamma$, (iii) $\sum_{i \in N_{\text{far}}(j)} x^*_{ij} = 1 - 1/\gamma$, and (iv) if $i \in N_{\text{cls}}(j)$ and $i' \in N_{\text{far}}(j)$ then $d_{ij} \leq d_{i'j}$. Intuitively, $N_{\text{cls}}(j)$ (also called *j's cluster*) consists of the first $1/\gamma$ "chunk" of the facilities nearest to j, while $N_{\text{far}}(j)$ contains the remaining facilities. We define the corresponding connection costs: $C^{\text{avg}}_{\text{cls}}(j) = \gamma \sum_{i \in N_{\text{cls}}(j)} d_{ij} x^*_{ij}$ and $C^{\text{avg}}_{\text{far}}(j) = \frac{\gamma}{\gamma - 1} \sum_{i \in N_{\text{far}}(j)} d_{ij} x^*_{ij}$ as the average distance from client j to facilities in $N_{\text{cls}}(j)$ and $N_{\text{far}}(j)$ respectively, and $C^{\text{max}}_{\text{cls}}(j) = \max_{i \in N_{\text{cls}}(j)} d_{ij}$ is the maximum distance from j to any facility in $N_{\text{cls}}(j)$. The overall average connection cost for j is $C^{\text{avg}}(j) = \sum_{i \in N(j)} d_{ij} x^*_{ij}$.

The BGS algorithm then runs in iterations. In each iteration, it picks a client p with minimum value of $C^{\text{avg}}_{\text{cls}}(p) + C^{\text{max}}_{\text{cls}}(p)$. If $N_{\text{cls}}(p)$ overlaps with that of some existing primary client p', then p is assigned to p'. Otherwise p becomes a primary client and assigned to itself. Once each client has been assigned to a primary client, the algorithm starts opening facilities. For each primary client p it opens one facility $\phi(p)$ in p's cluster $N_{\text{cls}}(p)$, with each facility $i \in N_{\text{cls}}(p)$ chosen as $\phi(p)$ with probability γy^*_i. In addition, each non-clustered facility i opens with probability γy^*_i, independently. For connections, each client j will first try to connect to the nearest open facility in $N(j)$. If none in $N(j)$ opens, then j gets connected to $\phi(p)$, called its *target* facility, where p is the primary client to which j is assigned.

It was shown in [2] that the above rounding process implies that $N_{\text{cls}}(j)$ has at least one open facility with probability no less than $1 - 1/e$ and that $N(j)$ has an open facility with probability at least $1 - 1/e^{\gamma}$. Hence with probability at most $1/e^{\gamma}$, a client uses an indirect connection to its target facility, whose cost is bounded by a technical lemma in [2].

Lemma 12. *[2] Define $d(j, A) = \sum_{i \in A} d_{ij} y^*_i / \sum_{i \in A} y_i$, let $\gamma < 2$ be a constant, and let p be the primary client that j was assigned to. Then either $N(p) - (N_{cls}(j) \cup N_{far}(j)) = \emptyset$ or $d(j, N(p) \setminus (N_{cls}(j) \cup N_{far}(j))) \leq C^{\text{avg}}_{cls}(j) + C^{\text{max}}_{cls}(j) + C^{\text{avg}}_{far}(j)$.*

Given Lemma 12, the expected connection cost for a client j is then

$$\begin{aligned}
\text{Exp}[C_j] &\leq C^{\text{avg}}_{\text{cls}}(j)(1 - 1/e) + C^{\text{avg}}_{\text{far}}(j)(1/e - 1/e^{\gamma}) + (C^{\text{avg}}_{\text{cls}}(j) + C^{\text{max}}_{\text{cls}}(j) + C^{\text{avg}}_{\text{far}}(j))1/e^{\gamma} \\
&\leq C^{\text{avg}}_{\text{cls}}(j)(1 - 1/e) + C^{\text{avg}}_{\text{far}}(j)(1/e - 1/e^{\gamma}) + (C^{\text{avg}}_{\text{cls}}(j) + 2C^{\text{avg}}_{\text{far}}(j))1/e^{\gamma} \\
&\leq C^{\text{avg}}(j)((1 - \rho_j)(\frac{1/e + 1/e^{\gamma}}{1 - 1/\gamma}) + \rho_j(1 + 2/e^{\gamma})),
\end{aligned}$$

where we define $\rho_j = C^{\text{avg}}_{\text{cls}}(j)/C^{\text{avg}}(j)$. It is easy to see that $C^{\text{avg}}(j) = \sum_{i \in \mathbb{F}} d_{ij} x^*_{ij} = \frac{1}{\gamma} C^{\text{avg}}_{\text{cls}}(j) + (1 - \frac{1}{\gamma}) C^{\text{avg}}_{\text{far}}(j)$. Then $C^{\text{avg}}_{\text{far}}(j) = (\gamma C^{\text{avg}}(j) - C^{\text{avg}}_{\text{cls}}(j))/(\gamma - 1)$. The approximation ratio is then bounded by the expression $\max\{\gamma, \frac{1/e + 1/e^{\gamma}}{1 - 1/\gamma}, 1 + 2/e^{\gamma}\}$, which is minimized for $\gamma = 1.575$.

We now describe our Algorithm EBGS for the FTFP problem. As before, our algorithm first partitions the instance and then applies the rounding approach similar to the BGS algorithm [3] described above. However, to apply the rounding approach similar to [3], we need to introduce the close and far neighborhood of each demand, and our properties of the partitioned fractional solution need to guarantee certain close neighborhoods overlap while other neighborhoods are disjoint. It turns out that the properties required are much more delicate but nonetheless, we show a modified partition algorithm actually delivers a partitioned fractional solution that satisfies all the properties. The rounding stage that construct an integral solution is relatively straightforward, as is the analysis of approximation ratio.

Stage 1: Adaptive Partitioning. The partitioning stage of Algorithm EBGS is similar to that in Section 6, with changes inspired by the BGS algorithm in [3] to obtain an improved ratio. In Stage 1, we also begin with $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ that is complete, that is $x^*_{ij} = y^*_i$ if $x^*_{ij} > 0$. Then we run our partition algorithm to obtain a fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ for the modified instance where clients are divided into unit-value demands and sites are divided into facilities. We adapt all notation for the modified instance from Sections 5 and 6. In particular, the neighborhood of a demand and its average connection cost is defined as before: $\overline{N}(\nu) = \{\mu \in \overline{\mathbb{F}} : \bar{x}_{\mu\nu} > 0\}$ and $C^{\text{avg}}(\nu) = \sum_{\mu \in \overline{N}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu}$, for each $\nu \in \overline{\mathbb{C}}$.

In our algorithm, the neighborhood $\overline{N}(\nu)$ of each demand ν will be partitioned into two parts, called the *close neighborhood* $\overline{N}_{\text{cls}}(\nu)$ and the *far neighborhood* $\overline{N}_{\text{far}}(\nu)$, whose formal definitions are given below as Property P1(e). Their respective average connection costs are defined by $C^{\text{avg}}_{\text{cls}}(\nu) = \gamma \sum_{\mu \in \overline{N}_{\text{cls}}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu}$ and $C^{\text{avg}}_{\text{far}}(\nu) = \frac{\gamma}{\gamma-1} \sum_{\mu \in \overline{N}_{\text{far}}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu}$. We will also use notation $C^{\max}_{\text{cls}}(\nu) = \max_{\mu \in \overline{N}_{\text{cls}}(\nu)} d_{\mu\nu}$ for the maximum distance from ν to its close neighborhood.

To better motivate our approach and for ease of presentation, we give only the desired properties of the partitioned instance and the fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ produced by our algorithm. The details on how to construct such a fractional solution, along with a proof of all the properties are deferred to the end of this section.

(P1) Vectors $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$ satisfy the following properties:

- (a) For each demand ν, $\sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu} = 1$.
- (b) $\sum_{\mu \in i, \nu \in j} \bar{x}_{\mu\nu} = x^*_{ij}$ for each site $i \in \mathbb{F}$ and client $j \in \mathbb{C}$.
- (c) $\sum_{\mu \in i} \bar{y}_\mu = y^*_i$ for each site $i \in \mathbb{F}$.
- (d) If $\bar{x}_{\mu\nu} \neq 0$ then $\bar{x}_{\mu\nu} = \bar{y}_\mu$, for each facility $\mu \in \overline{\mathbb{F}}$ and demand $\nu \in \overline{\mathbb{C}}$ (completeness).
- (e) For each demand ν, its neighborhood is divided into *close* and *far* neighborhood, that is $\overline{N}(\nu) = \overline{N}_{\text{cls}}(\nu) \cup \overline{N}_{\text{far}}(\nu)$, where (n1) $\overline{N}_{\text{cls}}(\nu) \cap \overline{N}_{\text{far}}(\nu) = \emptyset$, (n2) $\sum_{\mu \in \overline{N}_{\text{cls}}(\nu)} \bar{x}_{\mu\nu} = 1/\gamma$, (n3) $\sum_{i \in \overline{N}_{\text{far}}(\nu)} \bar{x}_{\mu\nu} = 1 - 1/\gamma$, and (n4) if $\mu \in \overline{N}_{\text{cls}}(\nu)$ and $\mu' \in \overline{N}_{\text{far}}(\nu)$ then $d_{\mu\nu} \leq d_{\mu'\nu}$.

(P2) Primary demands satisfy the following properties:

- (a) $\sum_{\kappa \in P} \sum_{\mu \in i \cap \overline{N}_{\text{cls}}(\kappa)} \bar{x}_{\mu\kappa} \leq y^*_i$, for each site $i \in \mathbb{F}$. Note that in the sum we do not count contributions from the far neighborhood of primary demands.
- (b) Each demand $\nu \in \overline{\mathbb{C}}$ is assigned to exactly one primary demand $\kappa \in P$ with an overlapping close neighborhood, that is $\overline{N}_{\text{cls}}(\nu) \cap \overline{N}_{\text{cls}}(\kappa) \neq \emptyset$.
- (c) Different demands from the same client are assigned to different primary demands.
- (d) For any two different sibling demands ν and ν', if ν is assigned to a primary demand κ then $(\overline{N}(\nu) \cup \overline{N}_{\text{cls}}(\kappa)) \cap \overline{N}(\nu') = \emptyset$. This condition ensures that no conflict can arise between sibling demands when they connect to facilities.
- (e) If a demand $\nu \in \overline{\mathbb{C}}$ is assigned to a primary demand $\kappa \in P$, then $C^{\text{avg}}_{\text{cls}}(\nu) + C^{\max}_{\text{cls}}(\nu) \geq C^{\text{avg}}_{\text{cls}}(\kappa) + C^{\max}_{\text{cls}}(\kappa)$.

Stage 2: Computing Integral Solution. Given the partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ with the desired properties, we then start opening facilities and making connections to obtain an integral

solution. As before we open exactly one facility in each cluster (the close neighborhood of a primary demand), but now each facility μ is chosen with probability $\gamma\bar{y}_\mu$. The non-clusterd facilities μ, those that do not belong to $\overline{N}_{\text{cls}}(\kappa)$ for any primary demand κ, are opened independently with probability $\gamma\bar{y}_\mu$ each. This implies that the expected facility cost of our algorithm is bounded by γF^*, using essentially the same argument as in the previous section (with the the factor γ accounting for using probabilities $\gamma\bar{y}_\mu$ instead of $\bar{y}_\mu$).

For connections, each primary demand κ will connect to the only facility $\phi(\kappa)$ open in its cluster $\overline{N}_{\text{cls}}(\kappa)$. For each non-primary demand ν, if there is an open facility in $\overline{N}(\nu)$ then we connect ν to the nearest such facility. Otherwise, we connect ν to $\phi(\kappa)$, where κ is the primary demand that ν is assigned to. This facility $\phi(\kappa)$ will be called the *target facility* of ν.

Analysis. The feasibility of our integral solution follows from Property P2(c), that each sibling demand is assigned to a different primary demand, and the disjoint-neighborhood property Property P2(d). These properties ensure that each facility is accessible to at most one demand among all demands of the same client.

We now bound the cost. Properties P2(b) and P2(e) allow us to bound the expected distance from a demand ν to its target facility by $C^{\text{avg}}_{\text{cls}}(\nu) + C^{\text{max}}_{\text{cls}}(\nu) + C^{\text{avg}}_{\text{far}}(\nu)$, in the event that none of ν's neighbors opens, using a similar argument as Lemma 2.2 in [3] [3]. We are then able to show that the expected connection cost for demand ν using an argument similar to [3].

$$\begin{aligned}
\text{Exp}[C_\nu] &\le C^{\text{avg}}_{\text{cls}}(\nu)(1-1/e) + C^{\text{avg}}_{\text{far}}(\nu)(1/e - 1/e^\gamma) + (C^{\text{avg}}_{\text{cls}}(\nu) + C^{\text{max}}_{\text{cls}}(\nu) + C^{\text{avg}}_{\text{far}}(\nu))1/e^\gamma \\
&\le C^{\text{avg}}_{\text{cls}}(\nu)(1-1/e) + C^{\text{avg}}_{\text{far}}(\nu)(1/e - 1/e^\gamma) + (C^{\text{avg}}_{\text{cls}}(\nu) + 2C^{\text{avg}}_{\text{far}}(\nu))1/e^\gamma \\
&\le C^{\text{avg}}(\nu)((1-\rho_\nu)(\frac{1/e+1/e^\gamma}{1-1/\gamma}) + \rho_\nu(1+2/e^\gamma)) \\
&\le C^{\text{avg}}(\nu)\cdot\max\{\frac{1/e+1/e^\gamma}{1-1/\gamma}, 1+\frac{2}{e^\gamma}\},
\end{aligned}$$

where $\rho_\nu = C^{\text{avg}}_{\text{cls}}(\nu)/C^{\text{avg}}(\nu)$. It is easy to see that ρ_ν is between 0 and 1 for every demand ν. Since $\sum_{\nu\in j} C^{\text{avg}}(\nu) = \sum_{\nu\in j}\sum_{\mu\in\overline{\mathbb{F}}} d_{\mu\nu}\bar{x}_{\mu\nu} = \sum_{i\in\mathbb{F}} d_{ij}x^*_{ij} = C^*_j$, summing over all clients j we have total connection cost bounded by $C^*\max\{\frac{1/e+1/e^\gamma}{1-1/\gamma}, 1+\frac{2}{e^\gamma}\}$. The expected facility cost is bounded by γF^*, as argued earlier. Hence the total cost is bounded by $\max\{\gamma, \frac{1/e+1/e^\gamma}{1-1/\gamma}, 1+\frac{2}{e^\gamma}\}\cdot \text{LP}^*$. Picking $\gamma = 1.575$ we obtain the desired ratio.

Stage 1 Implementation. We now return to Stage 1. Our adaptive partitioning scheme in the EBGS algorithm is very similar to that in the EGUP algorithm, except that we now cut a chunk of $1/\gamma$ when creating a primary demand, instead of 1. In the augment step we will add additional facilities to the neighborhood of all demands to make them each have a neighborhood with total connection value of 1.

Our scheme still runs in iterations. Consier any client j. Let $\widetilde{N}(j)$ be the set of facilities μ such that $\widetilde{x}_{\mu j} > 0$. Order facilities in $\widetilde{N}(j)$ such that $d_{1j} \le d_{2j} \le \ldots \le d_{|\widetilde{N}(j)|j}$ and $\sum_{s=1}^{l}\widetilde{x}_{sj} = 1/\gamma$ for some integer l (we split the l^{th} facility if necessary to make the sum equal to $1/\gamma$). Then the set of facilities $1, 2, \ldots, l$ is defined as $\widetilde{N}_\gamma(j)$. The value of $\text{tcc}(j)$ is now redefined as the average distance from j to $\widetilde{N}_\gamma(j)$. Also let $\text{dmax}(j)$ be $\max_{\mu\in\widetilde{N}_\gamma(j)} d_{\mu j}$. In each iteration, we find a client p with minimum value $\text{tcc}(p) + \text{dmax}(p)$. We then create a demand ν for client p. Now we have two cases:

[3] The full proof of the lemma appears in [2] as Lemma 3.3.

Case 1: If $\widetilde{N}_\gamma(p)$ overlaps $\overline{N}_{\text{cls}}(\kappa)$ for any existing primary demand κ, we simply assign ν to κ. As before, if there are multiple such κ, we pick any of them. We also fix $\bar{x}_{\mu\kappa} \leftarrow \widetilde{x}_{\mu p}, \widetilde{x}_{\mu p} \leftarrow 0$ for each $\mu \in \widetilde{N}(p) \cap \overline{N}_{\text{cls}}(\kappa)$. At this point the total connection value in $\overline{N}(\nu)$ might be smaller than $1/\gamma$ (it cannot be bigger) and we shall augment ν with additional facilities later to make a neighborhood with total connection value of 1 (We augment to make sum of $\bar{x}_{\mu\nu}$ for all $\mu \in \overline{N}(\nu)$ equal to 1.).

Case 2: The best client p has $\widetilde{N}_\gamma(p)$ disjoint from $\overline{N}_{\text{cls}}(\kappa)$, for all primary demands κ. In this case we make ν a primary demand. We then fix $\bar{x}_{\mu\kappa} \leftarrow \widetilde{x}_{\mu p}$ for $\mu \in \widetilde{N}_\gamma(p)$ and set the corresponding $\widetilde{x}_{\mu p}$ to 0. Note that the total connection value in $\overline{N}_{\text{cls}}(\kappa)$ is $1/\gamma$. The set $\widetilde{N}_\gamma(p)$ turns out to coincide with $\overline{N}_{\text{cls}}(\kappa)$ as we only add farther away facilities when augmenting a primary demand thereafter. Thus $\overline{N}_{\text{cls}}(\kappa)$ is defined when it is created. We also define $\text{tcc}(\kappa) = \text{tcc}(p)$ and $C^{\text{avg}}_{\text{cls}}(\kappa) = \gamma \sum_{\mu \in \overline{N}_{\text{cls}}(\kappa)} d_{\mu\kappa} \bar{x}_{\mu\kappa}, C^{\max}_{\text{cls}}(\kappa) = \max_{\mu \in \overline{N}_{\text{cls}}(\kappa)} d_{\mu\kappa}$.

Once all clients are exhausted, that is, each has r_j demands created and assigned to some primary demand, we do an augment step. For each demand with total connection value less than 1, we use our AugmentToUnit() procedure to add additional facilities to its neighborhood to make its total connection value equal 1. We remark here that the augment step does not change the close neighborhood of a primary demand, as it already contains all the nearest facilities with total connection value $1/\gamma$. For non-primary demands, the issue is more subtle, because $\overline{N}(\nu)$ has taken all overlapping facilities in $\overline{N}_{\text{cls}}(\kappa) \cap \widetilde{N}(j)$, which might be close to κ but far from j. It seems that facilities added in the augment step might actually be closer to ν than some of the overlapping facilities already in $\overline{N}(\nu)$. As a result facilities added in the augment step might appear in ν's close neighborhood $\overline{N}_{\text{cls}}(\nu)$, yet they are not in $\overline{N}_{\text{cls}}(\kappa)$ of the primary demand κ that ν is assigned to. This could be detrimental to ensuring Property P2(b), which requires the close neighborhood of demand ν and that of its primary demand κ need overlap. In the previous analysis of approximation ratio, our bound on the expected connection cost depends on the intersection of $\overline{N}_{\text{cls}}(\nu)$ and $\overline{N}_{\text{cls}}(\kappa)$ being nonempty. In the following, however, we give a proof on all properties P1(a) to P1(e) and P2(a) to P2(e).

Proof of the Properties. The proof of Property P1(a), P1(b), P1(c), P1(d), P1(e) are very similar to that in the EGUP algorithm's partition stage. These properties are directly enforced by the partition algorithm, by splitting a facility whenever some of the properties are violated. Property P2(a) follows from the way we adjust $\widetilde{x}_{\mu j}$ and $\bar{x}_{\mu\nu}$ values when splitting a facility, and the fact that the close neighborhood of primary demands are disjoint. Property P2(e) follows from our choice of primary clients, and the proof is very similar to the proof of Lemma **??**. The main observation is that for any client j, the value $\text{tcc}(j)$ can only get worse (larger) as the partition stage proceeds in iterations.

We now show Property P2(c) and P2(d) hold. First we start with a simple observation, that in the end of our partition algorithm, neighborhoods of sibling demands are disjoint. This follows from the fact that when the algorithm adds a facility to the neighborhood of a demand, it never adds the same facility to that of a sibling demand. An immediate implication is that close neighborhoods of sibling demands are disjoint as well. Now consider an iteration when we create a demand ν for some client j. If ν is set to be a new primary demand, then $\overline{N}_{\text{cls}}(\nu)$ is necessarily disjoint from the close neighborhood of its siblings. Therefore no sibling will get assigned to ν, and by the algorithm ν is assigned to itself. Otherwise ν is assigned to some existing primary demand κ. According to the algorithm, all facilities in $\overline{N}_{\text{cls}}(\kappa) \cap \widetilde{N}(j)$ are moved into $\overline{N}(\nu)$ and excluded from $\widetilde{N}(j)$. It follows that after that iteration, we have $\widetilde{N}(j)$ and $\overline{N}_{\text{cls}}(\kappa)$ being disjoint. Therefore later demands cannot

possibly have a neighborhood overlapping $\overline{N}_{\text{cls}}(\kappa)$. At this point, earlier sibling demands each has a neighborhood being a subset of $\overline{N}_{\text{cls}}(\kappa')$ where κ' is the primary demand it was assigned. Since we know $\overline{N}_{\text{cls}}(\kappa) \cap \overline{N}_{\text{cls}}(\kappa') = \emptyset$ from the partition algorithm, we have that no sibling demand has a neighborhood overlapping $\overline{N}_{\text{cls}}(\kappa)$ at the end of this iteration. Although we add more facilities to the neighborhoods of sibling demands in the augment step, those facilities cannot appear in $\overline{N}_{\text{cls}}(\kappa)$ either, because all overlapping facilities have already been moved from $\widetilde{N}(j)$ into $\overline{N}(\nu)$. As a result, at most one sibling demand can have a neighborhood overlapping $\overline{N}_{\text{cls}}(\kappa)$. Thus we have Property P2(d) established. Since the algorithm only assign a demand to a primary demand with common facility, this proves Property P2(c).

We now show our last property P2(b). Consider an iteration when we create demand ν for client p and assign it to κ. We have that $\widetilde{N}_\gamma(p)$ having a nonempty intersection with $\overline{N}_{\text{cls}}(\kappa)$. Let $B(p) = \widetilde{N}_\gamma(p) \cap \overline{N}_{\text{cls}}(\kappa)$, we claim that $B(p)$ must be a subset of $\overline{N}_{\text{cls}}(\nu)$, after $\overline{N}(\nu)$ is finalized with a total connection value of 1. To see this, first observe that $B(p)$ is a subset of $\overline{N}(\nu)$, which in turn is a subset of $\widetilde{N}(p)$, after taking into account of facility split. Consider an arbitrary set of facilities A, and define $\text{dmax}(\nu, A)$ as the minimum distance τ such that $\sum_{\mu \in A\,:\, d_{\mu\nu} \le \tau} \bar{y}_\mu \ge 1/\gamma$, then adding additional facilities into A cannot make $\text{dmax}(\nu, A)$ larger. It follows that $\text{dmax}(\nu, \overline{N}_{\text{cls}}(\nu)) \ge \text{dmax}(\nu, \widetilde{N}(p))$. Since we have $d_{\mu\nu} = d_{\mu p}$ by definition, it is easy to see that every $\mu \in B(p)$ satisfies $d_{\mu\nu} \le \text{dmax}(\nu, \widetilde{N}(p)) \le \text{dmax}(\nu, \overline{N}_{\text{cls}}(\nu))$ and hence they all belong to $\overline{N}_{\text{cls}}(\nu)$. We need to be a bit more careful here when we have a tie in $d_{\mu\nu}$ but we can assume ties are always broken in favor of facilities in $B(p)$ when deciding $\overline{N}_{\text{cls}}(\nu)$. Finally, since $B(p)$ is nonempty, we have that the close neighborhood of a demand ν and its primary demand κ must overlap. This proves Property P2(b).

The above concludes our discussion on the Stage 1 partitioning process and we have thus established that our EBGS algorithm is a 1.575-approximation algorithm.

8 Discussions

In this paper we show a sequence of LP-rounding approximation algorithms for FTFP, with the best algorithm achieving ratio 1.575. The two techniques we introduced, namely the demand reduction and adaptive partitioning, are very flexible. Should any new LP-rounding algorithms be discovered for UFL, we believe that with our approach they can be adapted to FTFP as well, preserving the approximation ratio. In fact, by randomizing the scaling parameter γ, as Li showed in [13], we could further improve the ratio to below 1.575, although we would not be able to match the 1.488 bound for UFL in [13], because this would also require appropriately extending dual-fitting algorithms [14] to FTFP, which we have so far been unable to do.

One of the main open problems in this area is whether FTFL can be approximated with the same ratio as UFL, and our is was partly motivated by this question. The techniques we introduced are not directly applicable to FTFL, however, mainly because our partitioning approach involves facility splitting that could result in several sibling demands being served by facilities on the same site. Nonetheless we hope our construction might inspire new algorithm for the FTFL algorithm with a matching ratio with LP-based algorithms for the UFL problem as well.

References

[1] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

[2] Jaroslaw Byrka and Karen Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM J. Comput.*, 39(6):2212–2231, 2010.

[3] Jaroslaw Byrka, MohammadReza Ghodsi, and Aravind Srinivasan. LP-rounding algorithms for facility-location problems. *CoRR*, abs/1007.3611, 2010.

[4] Jaroslaw Byrka, Aravind Srinivasan, and Chaitanya Swamy. Fault-tolerant facility location, a randomized dependent LP-rounding algorithm. In *Proceedings of the 14th Integer Programming and Combinatorial Optimization, IPCO '10*, pages 244–257, 2010.

[5] Fabián Chudak and David Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.*, 33(1):1–25, 2004.

[6] Sudipto Guha and Samir Khuller. Greedy strikes back: improved facility location algorithms. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, SODA '98*, pages 649–657, 1998.

[7] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Improved algorithms for fault tolerant facility location. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms, SODA '01*, pages 636–641, 2001.

[8] Anupam Gupta. Lecture notes: CMU 15-854b, spring 2008, 2008.

[9] Godfrey Harold Hardy, John Edensor Littlewood, and George Pólya. *Inequalities*. Cambridge University Press, 1988.

[10] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM*, 50(6):795–824, 2003.

[11] Kamal Jain and Vijay Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

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

[13] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. In *Proceedings of the 38th International Conference on Automata, Languages and Programming, ICALP '11*, volume 6756, pages 77–88, 2011.

[14] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Approximation algorithms for metric facility location problems. *SIAM J. Comput.*, 36(2):411–432, 2006.

[15] David Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC '97*, pages 265–274, 1997.

[16] Maxim Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization, IPCO '02*, pages 240–257, 2002.

[17] Chaitanya Swamy and David Shmoys. Fault-tolerant facility location. *ACM Trans. Algorithms*, 4(4):1–27, 2008.

[18] Jens Vygen. *Approximation Algorithms for Facility Location Problems.* Forschungsinst. für Diskrete Mathematik, 2005.

[19] Shihong Xu and Hong Shen. The fault-tolerant facility allocation problem. In *Proceedings of the 20th International Symposium on Algorithms and Computation, ISAAC '09*, pages 689–698, 2009.

[20] Li Yan and Marek Chrobak. Approximation algorithms for the fault-tolerant facility placement problem. *Inf. Process. Lett.*, 111(11):545–549, 2011.

A An Elementary Proof of the Expected Connection Cost

In the $1 + 2/e = 1.736$-approximation, we need to show the following inequality

$$d_1y_1+d_2y_2(1-y_1)+d_3y_3(1-y_1)(1-y_2)+\ldots+d_ly_l\Pi_{s=1}^{l-1}(1-y_s) \leq (d_1y_1+d_2y_2+\ldots+d_ly_l)(1-\Pi_{s=1}^{l}(1-y_s)) \tag{4}$$

for $d_1 \leq d_2 \leq \ldots \leq d_l$ and $\sum_{s=1}^{l} y_s = 1, y_s \geq 0$.

In this section we give a new proof of this inequality, much simpler than the existing proof in [5] [4]. We derive this inequality from the following generalized version of the Chebyshev Sum Inequality:

$$\sum\nolimits_i p_i \sum\nolimits_j p_j a_j b_j \leq \sum\nolimits_i p_i a_i \sum\nolimits_j p_j b_j, \tag{5}$$

where each summation below runs from 1 to l and the sequences (a_i), (b_i) and (p_i) satisfy the following conditions: $p_i \geq 0, a_i \geq 0, b_i \geq 0$ for all i, $a_1 \leq a_2 \leq \ldots \leq a_l$, and $b_1 \geq b_2 \geq \ldots \geq b_l$.

Given inequality (5), we can obtain our inequality (4) by simple substitution

$$p_i \leftarrow y_i, a_i \leftarrow d_i, b_i \leftarrow \Pi_{s=1}^{i-1}(1 - y_s) \tag{6}$$

For the sake of completeness, we include the proof of inequality (5), due to Hardy, Littlewood

[4] It is known that Sviridenko has proved a similar lemma using Chebyshev's sum inequality, but the cited reference [16] contains a different proof from ours here.

and Polya [9]. The idea is to evaluate the following sum:

$$\begin{aligned}
S &= \sum_i p_i \sum_j p_j a_j b_j - \sum_i p_i a_i \sum_j p_j b_j \\
&= \sum_i \sum_j p_i p_j a_j b_j - \sum_i \sum_j p_i a_i p_j b_j \\
&= \sum_j \sum_i p_j p_i a_i b_i - \sum_j \sum_i p_j a_j p_i b_i \\
&= \tfrac{1}{2} \cdot \sum_i \sum_j (p_i p_j a_j b_j - p_i a_i p_j b_j + p_j p_i a_i b_i - p_j a_j p_i b_i) \\
&= \tfrac{1}{2} \cdot \sum_i \sum_j p_i p_j (a_i - a_j)(b_i - b_j) \leq 0.
\end{aligned}$$

The last inequality holds because $(a_i - a_j)(b_i - b_j) \leq 0$, since the sequences (a_i) and (b_i) are ordered oppositely.