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

Li Yan and Marek Chrobak

Computer Science, U of California, Riverside
{lyan,marek}@cs.ucr.edu

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 d_{ij} 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 (possibly on the same site).

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, then the problem is equivalent to the Fault-Tolerant Facility Location problem (FTFL). Note that in FTFL we have $\max_{j\in\mathbb{C}} r_j \leq |\mathbb{F}|$, while in FTFP the values of r_j's can be much bigger than $|\mathbb{F}|$.

Great progress has been achieved lately in designing approximation algorithms for UFL. Shmoys *et al.* [16] proposed an approach based on LP-rounding, achieving a ratio of 3.16. This was then improved by Chudak [5] to 1.736, and later by Sviridenko [17] to 1.582. Byrka [2] gave an improved algorithm with ratio 1.5, by a combination of LP-rounding with dual-fitting techniques. Recently, Li [13] refined the method from [2] to obtain ratio 1.488, which is now the best known approximation result for UFL. Other techniques include the primal-dual algorithm by Jain and Vazirani [11], the dual fitting method by Jain *et al.* [10], and a local search heuristic by Arya *et al.* [1]. On the hardness side, 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 this assumption to $\mathsf{P} \neq \mathsf{NP}$ [19].

FTFL was first introduced by Jain and Vazirani [12] who gave a primal-dual algorithm with ratio $3\ln(\max_{j\in\mathbb{C}} r_j)$. All subsequently discovered constant-ratio approximation algorithms use variations of LP-rounding, including the work by Guha *et al.* [7], Swamy and Shmoys [18], and Byrka *et al.* [4], who improved the ratio to 1.7245, the best known approximation ratio for FTFL.

FTFP is a natural generalization of UFL. It was first studied by Xu and Shen [20], who presented 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 their analysis of the approximation ratio is flawed[1]. To date, the best approximation ratio for FTFP is 4 in [21], while the only known lower bound is the 1.463 lower bound for UFL from [6], that applies to FTFP.

The main result of this paper is an LP-rounding algorithm for FTFP with approximation ratio 1.575, matching the best ratio for UFL achieved via the LP-rounding method [3] and significantly improving the bound in [21]. In Section 3 we prove that, for the purpose of LP-based approximations, we can assume that all demand values are polynomial in the number of sites. This *demand reduction* trick itself gives us ratio 1.7245, since we can then treat an instance of FTFP as an instance of FTFL and use the algorithm from [4]. It also ensures that our algorithms run in polynomial time. If all demand values r_j are equal, the problem can be solved by simple scaling and applying LP-rounding algorithms for UFL. This does not affect the approximation ratio, thus achieving ratio 1.575 for this special case (see also [14]). In Section 4, we demonstrate a technique called *adaptive partitioning*, which splits clients into unit demands and partitions the optimal fractional solution into a fractional solution of the split instance. By exploiting structural properties of the partitioned solution we were able to extend UFL rounding algorithms in [8, 5, 3], retaining the approximation ratio.

Summarizing, 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, using our demand reduction and adaptive partitioning methods, it should be possible to extend them to FTFP. In fact, some improvement of the ratio can be achieved by randomizing the parameter γ used in Section 7, as Li showed in [13] for UFL. Our ratio of 1.575 is significantly better than the best 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 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:

$$\begin{aligned} \min\ & cost(\boldsymbol{x},\boldsymbol{y}) = \textstyle\sum_{i\in\mathbb{F}} f_i y_i + \sum_{i\in\mathbb{F},j\in\mathbb{C}} d_{ij}x_{ij} && && (1)\\ \text{s.t. }\ & y_i - x_{ij} \geq 0 && \forall i\in\mathbb{F}, j\in\mathbb{C}\\ & \textstyle\sum_{i\in\mathbb{F}} x_{ij} \geq r_j && \forall j\in\mathbb{C}\\ & x_{ij}\geq 0, y_i \geq 0 && \forall i\in\mathbb{F}, j\in\mathbb{C} \end{aligned}$$

$$\begin{aligned} \max\ & \textstyle\sum_{j\in\mathbb{C}} r_j\alpha_j && && (2)\\ \text{s.t. }\ & \textstyle\sum_{j\in\mathbb{C}} \beta_{ij} \leq f_i && \forall i\in\mathbb{F}\\ & \alpha_j - \beta_{ij} \leq d_{ij} && \forall i\in\mathbb{F}, j\in\mathbb{C}\\ & \alpha_j \geq 0, \beta_{ij}\geq 0 && \forall i\in\mathbb{F}, j\in\mathbb{C} \end{aligned}$$

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 $\mathrm{LP}^* = cost(\boldsymbol{x}^*,\boldsymbol{y}^*) = F^*+C^*$ 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 $\mathrm{OPT}\geq \mathrm{LP}^*$, we can instead use LP^* to estimate the approximation ratio of our algorithms.

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

[1] Confirmed through private communication with the authors.

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.

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 1.) 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 the overall running time polynomial.

The reduction is based on an optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of LP (1). From the optimality of this solution, we can also 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 $\hat{y}_i \leftarrow \lfloor y^*_i \rfloor$, $\hat{x}_{ij} \leftarrow \lfloor x^*_{ij} \rfloor$ and $\dot{y}_i \leftarrow y^*_i - \lfloor y^*_i \rfloor$, $\dot{x}_{ij} \leftarrow x^*_{ij} - \lfloor x^*_{ij} \rfloor$ 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 1. *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 $\rho \ge 1$. 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 it 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 $\rho \cdot \mathit{cost}(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$, because $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$ is feasible for $\dot{\mathcal{I}}$. Thus the cost of $\mathcal{A}'$ is at most $\mathit{cost}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) + \rho \cdot \mathit{cost}(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}) \le \rho(\mathit{cost}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) + \mathit{cost}(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})) = \rho \cdot \mathrm{LP}^* \le \rho \cdot \mathrm{OPT}$, where the first inequality follows from $\rho \ge 1$. 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 $|\mathbb{F}|+2R|\mathbb{C}|^2$ *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 $\bar{\boldsymbol{y}}=(\bar{y}_\mu)$. The *neighborhood of a demand* ν is $\overline{N}(\nu)=\{\mu\in\overline{\mathbb{F}}:\bar{x}_{\mu\nu}>0\}$. 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 $C^{\mathrm{avg}}_\nu=\sum_{\mu\in\overline{N}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu}=\sum_{\mu\in\overline{\mathbb{F}}} d_{\mu\nu}\bar{x}_{\mu\nu}$. 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 to satisfy the fault-tolerance requirement. Specifically, this partitioning will be constructed to satisfy a number of properties that are detailed below.

(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 $i\in\mathbb{F}$, $\sum_{\mu\in i}\sum_{\kappa\in P}\bar{x}_{\mu\kappa}\le y^*_i$.
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) $C^{\mathrm{avg}}_\nu+\alpha^*_\nu\ge C^{\mathrm{avg}}_\kappa+\alpha^*_\kappa$.

(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. Moreover, we would like to point out that the adaptive partitioning process for the 1.575-approximation algorithm (Section 7) is more subtle than the 3-apprximation (Section 5) and the 1.736-approximation algorithms (Section 6), due to the introduction of close and far neighborhood.

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.

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 j 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. In Phase 2 we ensure that every demand has a total connection values equal to 1, that is condition (PS.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 already 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 allocated to existing 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 $j \in \mathbb{C}$ and $\overline{N}(\nu) = \{\mu \in \overline{\mathbb{F}} : \bar{x}_{\mu\nu} > 0\}$ for $\nu \in \overline{\mathbb{C}}$. During this process, the following properties will hold for every facility μ after every iteration:

- (c1) For each demand ν either $\bar{x}_{\mu\nu} = 0$ or $\bar{x}_{\mu\nu} = \bar{y}_\mu$.
- (c2) For each client j, either $\widetilde{x}_{\mu j} = 0$ or $\widetilde{x}_{\mu j} = \bar{y}_\mu$.

It may appear that (c1) is the same condition as (CO), yet we repeat it here as (c1) needs to hold after every iteration, while (CO) only applies to the final partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$;

A full description of the algorithm is given in Pseudocode 1. In Phase 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. In one iteration of the while loop, 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(), which adds facilities to $\widetilde{N}_1(j)$ in order of nondecreasing distance, until the total connection value is exactly 1. This may require splitting the last added facility and adjusting the connection values so that conditions (c1) and (c2) are preserved.

The next step is to pick a client p with minimum $\mathrm{tcc}(p) + \alpha_p^*$ and create a demand ν for p. 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 $\widetilde{N}(p) \cap \overline{N}(\kappa)$. 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, ν becomes itself a primary demand and 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. Phase 1 concludes when all clients j are exhausted.

In Phase 2, we adjust the $\bar{x}_{\mu\nu}$ values for non-primary demands ν so that $\mathrm{conn}(\nu) \stackrel{\text{def}}{=} \sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu}$, is equal to 1 by invoking Procedure AugmentToUnit(), which repeatedly picks any facility η with $\widetilde{x}_{\eta j} > 0$ and reassigns $\widetilde{x}_{\eta j}$ to ν, until $\mathrm{conn}(\nu)$ reaches 1.

Notice that we start with $|\mathbb{F}|$ facilities and in each iteration of the while loop 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 $R = \max_j r_j$.) In Phase 2 we do an augment step for each demand ν and this creates no more than $R|\mathbb{C}|$ new facilities. So the total number of facilities we created will be at most $|\mathbb{F}| + R|\mathbb{C}|^2 + R|\mathbb{C}| \leq |\mathbb{F}| + 2R|\mathbb{C}|^2$, 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.

(CO) is implied by the completeness condition (c1) that the algorithm maintains after each iteration. (PS.1) is a result of calling Procedure AugmentToUnit() in Line 18. To see that (PS.2) holds, note that at each step the algorithm maintains the 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

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

(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) holds because 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 of $\bar{y}_\mu$.

Now we deal with properties in group (PD). First, (PD.1) follows directly from the algorithm, Pseudocode 1 (Line 13), since every primary demand has its neighborhood fixed when created, and that neighborhood is disjoint from those of the existing primary demands. Property (PD.2) follows from (PD.1), (CO) and (PS.3). In more detail, it can be justified as follows. By (PD.1), for each $\mu \in i$ there is at most one $\kappa \in P$ with $\bar{x}_{\mu\kappa} > 0$ and we have $\bar{x}_{\mu\kappa} = \bar{y}_\mu$ due do (CO). Let $K \subseteq i$ be the set of those μ's for which such $\kappa \in P$ exists, and denote this κ by κ_μ. Then, using conditions (CO) and (PS.3), we have $\sum_{\mu \in i} \sum_{\kappa \in P} \bar{x}_{\mu\kappa} = \sum_{\mu \in K} \bar{x}_{\mu\kappa_\mu} = \sum_{\mu \in K} \bar{y}_\mu \leq \sum_{\mu \in i} \bar{y}_\mu = y_i^*$. Property (PD.3(a)) follows from the way the algorithm assigns primary demands. When demand ν of client p is assigned to a primary demand κ in Line 11 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 for (PD.3(b)) to Lemma 4.

Finally we argue that the properties in group (SI) hold. (SI.1) is easy, since for any client j, each facility μ is added to the neighborhood of at most one demand $\nu \in j$, by setting $\bar{x}_{\mu\nu}$ to $\bar{y}_\mu$, while other siblings ν' of 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 Line 11 of Pseudocode 1. Thus all demands of p created later will have neighborhoods disjoint from the set $\overline{N}(\nu)$ before the augmenting phase 2. Furthermore, Procedure AugmentToUnit() preserves this property, because when it adds a 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 $\widetilde{N}(p)$. Property (SI.2) is shown below in Lemma 2.

Lemma 2. *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. After the completion of Phase 1 of Pseudocode 1 (Lines 1–15), we have $\overline{N}(\nu_s) \subseteq \overline{N}(\kappa_s)$ for $s = 1, \ldots, r_j$. Since any two primary demands have disjoint neighborhoods, we have $\overline{N}(\nu_s) \cap \overline{N}(\kappa_q) = \emptyset$ for any $s \neq q$, that is Property (SI.2) holds right after Phase 1.

Pseudocode 2 Helper functions used in Pseudocode 1

1: **function** NearestUnitChunk($j, \overline{\mathbb{F}}, \widetilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$) ▷ upon return, $\sum_{\mu \in \widetilde{N}_1(j)} \widetilde{x}_{\mu j} = 1$
2: Let $\widetilde{N}(j) = \{\mu_1, ..., \mu_q\}$ where $d_{\mu_1 j} \leq d_{\mu_2 j} \leq \ldots \leq d_{\mu_q j}$
3: Let l be such that $\sum_{k=1}^{l} \bar{y}_{\mu_k} \geq 1$ and $\sum_{k=1}^{l-1} \bar{y}_{\mu_k} < 1$
4: Create a new facility σ at the same site as μ_l and add it to $\overline{\mathbb{F}}$ ▷ split μ_l
5: Set $\bar{y}_\sigma \leftarrow \sum_{k=1}^{l} \bar{y}_{\mu_k} - 1$ and $\bar{y}_{\mu_l} \leftarrow \bar{y}_{\mu_l} - \bar{y}_\sigma$
6: For each $\nu \in \overline{\mathbb{C}}$ with $\bar{x}_{\mu_l \nu} > 0$ set $\bar{x}_{\mu_l \nu} \leftarrow \bar{y}_{\mu_l}$ and $\bar{x}_{\sigma\nu} \leftarrow \bar{y}_\sigma$
7: 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 $\widetilde{x}_{\sigma j'} \leftarrow \bar{y}_\sigma$
8: (All other new connection values are set to 0)
9: **return** $\widetilde{N}_1(j) = \{\mu_1, \ldots, \mu_{l-1}, \mu_l\}$
10: **function** AugmentToUnit($\nu, j, \overline{\mathbb{F}}, \widetilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$) ▷ ν is a demand of client j
11: **while** $\sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu} < 1$ **do** ▷ upon return, $\sum_{\mu \in \overline{N}(\nu)} \bar{x}_{\mu\nu} = 1$
12: Let η be any facility such that $\widetilde{x}_{\eta j} > 0$
13: **if** $1 - \sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu} \geq \widetilde{x}_{\eta j}$ **then**
14: $\bar{x}_{\eta\nu} \leftarrow \widetilde{x}_{\eta j}, \widetilde{x}_{\eta j} \leftarrow 0$
15: **else**
16: Create a new facility σ at the same site as η and add it to $\overline{\mathbb{F}}$ ▷ split η
17: Let $\bar{y}_\sigma \leftarrow 1 - \sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu}, \bar{y}_\eta \leftarrow \bar{y}_\eta - \bar{y}_\sigma$
18: Set $\bar{x}_{\sigma\nu} \leftarrow \bar{y}_\sigma,\ \bar{x}_{\eta\nu} \leftarrow 0,\ \widetilde{x}_{\eta j} \leftarrow \bar{y}_\eta,\ \widetilde{x}_{\sigma j} \leftarrow 0$
19: For each $\nu' \neq \nu$ with $\bar{x}_{\eta\nu'} > 0$ set $\bar{x}_{\eta\nu'} \leftarrow \bar{y}_\eta,\ \bar{x}_{\sigma\nu'} \leftarrow \bar{y}_\sigma$
20: For each $j' \neq j$ with $\widetilde{x}_{\eta j'} > 0$ set $\widetilde{x}_{\eta j'} \leftarrow \bar{y}_\eta, \widetilde{x}_{\sigma j'} \leftarrow \bar{y}_\sigma$
21: (All other new connection values are set to 0)

After Phase 1 all neighborhoods $\overline{N}(\kappa_s), s = 1, \ldots, r_j$ have already been fixed and they do not change in Phase 2. None of the facilities in $\widetilde{N}(j)$ appear in any of $\overline{N}(\kappa_s)$ for $s = 1, \ldots, r_j$, by the way we allocate facilities in Lines 11 and 13. Therefore during the augmentation process in Phase 2, when we add facilities from $\widetilde{N}(j)$ to $\overline{N}(\nu)$, for some $\nu \in j$ (Line 16–18 of Pseudocode 1), all the required disjointness conditions will be preserved. This completes the proof.

We need one more 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 3. *Let η and ν be two demands, with η created not later than ν, and let $j \in \mathbb{C}$ be a client that is not exhausted when ν is created. Then we have (a) $tcc_\eta(j) \leq tcc_\nu(j)$, and (b) if $\nu \in j$ then $tcc_\eta(j) \leq C_\nu^{\mathrm{avg}}$.*

Proof. We focus first on the time when demand η is about to be created, right after the call to NearestUnitChunk() in Pseudocode 1, Line 6. 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 optimal fractional solution is the greedy solution, whose value is exactly $\mathrm{tcc}_\eta(j)$.

On the other hand, we claim that $\mathrm{tcc}_\nu(j)$ can be thought of as the value of some feasible solution to this linear program, and that the same is true for C_ν^{avg} if $\nu \in j$. 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 obtained from splitting. For each s, however, the sum of all values $\widetilde{x}_{\mu j}$, over the facilities μ 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 $\mathrm{tcc}_\nu(j)$ and C^{avg}_ν (for $\nu \in j$) correspond to some choice of the z_s variables (adding up to 1), and the lemma follows.

Lemma 4. *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

$$C^{\mathrm{avg}}_\kappa + \alpha^*_\kappa \;=\; \mathrm{tcc}_\kappa(p) + \alpha^*_p \;\le\; \mathrm{tcc}_\kappa(j) + \alpha^*_j \;\le\; C^{\mathrm{avg}}_\nu + \alpha^*_\nu.$$

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^{avg}_κ is equal to $\mathrm{tcc}(p)$ computed when κ is created, which is exactly $\mathrm{tcc}_\kappa(p)$. Thus the first equation is true. The first inequality follows from the choice of p in Line 8 in Pseudocode 1. The last inequality holds because $\alpha^*_j = \alpha^*_\nu$ (due to $\nu \in j$), and because $\mathrm{tcc}_\kappa(j) \le C^{\mathrm{avg}}_\nu$, which follows from Lemma 3.

We have thus proved that all properties (PS), (CO), (PD) and (SI) hold for our partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$. In the following sections we show how to use these properties to round the fractional solution to an approximate integral solution. For the 3-approximation algorithm (Section 5) and the 1.736-approximation algorithm (Section 6), the first phase of the algorithm is exactly the same partition process as described above. However, the 1.575-approximation algorithm (Section 7) demands a more sophisticated partitioning process as the interplay between close and far neighborhood of sibling demands result in more delicate properties that our partitioned fractional solution must satisfy.

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. The algorithm we describe in this section achieves ratio 3. We use this relatively simple algorithm to illustrate how the properties of our partitioned fractional solution are used in rounding to obtain an integral solution with cost close to an optimal solution. The rounding approach is an extension to the method for UFL in [8].

Algorithm EGUP. In Algorithm EGUP, we apply a rounding process, guided by the fractional values $(\bar{y}_\mu)$ and $(\bar{x}_{\mu\nu})$, that produces an integral solution. For each primary demand $\kappa \in P$, we 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 this facility $\phi(\kappa)$ and all non-primary demands that are assigned to κ. We bound the expected facility cost and connection cost by establishing the two lemmas below and establish feasibility in the theorem.

Lemma 5. *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

$$\mathrm{Exp}[F_{\mathrm{EGUP}}] = \textstyle\sum_{\kappa\in P}\sum_{\mu\in\overline{N}(\kappa)} f_\mu \bar{x}_{\mu\kappa} = \sum_{\kappa\in P}\sum_{\mu\in\overline{\mathbb{F}}} f_\mu \bar{x}_{\mu\kappa} = \sum_{i\in\mathbb{F}} f_i \sum_{\mu\in i}\sum_{\kappa\in P} \bar{x}_{\mu\kappa} \le \sum_{i\in\mathbb{F}} f_i y^*_i = F^*,$$

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

Lemma 6. *The expectation of connection cost C_{EGUP} of our solution is at most $C^* + 2\cdot \mathrm{LP}^*$.*

Proof (Proof of Lemma 6). For a primary demand κ, its expected connection cost is C^{avg}_κ because we choose facility μ with probability $\bar{x}_{\mu\kappa}$.

Consider a non-primary 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

$$\mathrm{Exp}[d_{\phi(\kappa)\nu}] \le C^{\mathrm{avg}}_\kappa + \alpha^*_\kappa + \alpha^*_\nu \le C^{\mathrm{avg}}_\nu + \alpha^*_\nu + \alpha^*_\nu = C^{\mathrm{avg}}_\nu + 2\alpha^*_\nu,$$

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

$$\sum_{\nu\in j} C^{\mathrm{avg}}_\nu = \sum_{\nu\in j} \sum_{\mu\in\overline{\mathbb{F}}} d_{\mu\nu}\bar{x}_{\mu\nu} = \sum_{i\in\mathbb{F}} d_{ij} \sum_{\nu\in j} \sum_{\mu\in i} \bar{x}_{\mu\nu} = \sum_{i\in\mathbb{F}} d_{ij} x^*_{ij} = C^*_j.$$

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

$$\mathrm{Exp}[C_{\mathrm{EGUP}}] \le \sum_{j\in\mathbb{C}} \sum_{\nu\in j} (C^{\mathrm{avg}}_\nu + 2\alpha^*_\nu) = \sum_{j\in\mathbb{C}} (C^*_j + 2r_j\alpha^*_j) = C^* + 2\cdot \mathrm{LP}^*,$$

completing the proof of the lemma.

Theorem 2. *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 5 and Lemma 6 imply that the total cost is at most $F^* + C^* + 2\cdot\mathrm{LP}^* = 3\cdot\mathrm{LP}^* \le 3\cdot\mathrm{OPT}$.

6 Algorithm ECHS with Ratio 1.736

In this section we improve the approximation ratio to $1+2/e \approx 1.736$. The improvement comes from a slightly modified rounding process and 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, since we connect a non-primary demand to a facility in its assigned primary demand's neighborhood and then estimate the distance using the triangle inequality. To improve the estimate of the connection cost, the basic idea, following the approach of Chudak and Shmoys [5], is to connect a non-primary demand to its nearest neighbor when one is available and only use the facility that its primary demand opens when none of its neighbor is open.

Algorithm ECHS. As before, the algorithm starts by solving the linear program and applying the adaptive partitioning algorithm described in Section 4 to obtain a partitioned solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$. Then we apply the rounding process to compute an integral solution (see Pseudocode 3).

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. Let ν and ν' be two different sibling demands of client j and let their assigned primary demands be κ and κ' respectively. Due to (SI.2) we know $\kappa \ne \kappa'$. From (SI.1) we have $\overline{N}(\nu) \cap \overline{N}(\nu') = \emptyset$. From (SI.2), we have $\overline{N}(\nu) \cap \overline{N}(\kappa') = \emptyset$ and $\overline{N}(\nu') \cap \overline{N}(\kappa) = \emptyset$. From (PD.1) we have $\overline{N}(\kappa) \cap \overline{N}(\kappa') = \emptyset$. It follows that $(\overline{N}(\nu) \cup \overline{N}(\kappa)) \cap (\overline{N}(\nu') \cup \overline{N}(\kappa')) = \emptyset$. Since the algorithm connects ν to some facility in $\overline{N}(\nu) \cup \overline{N}(\kappa)$ and ν' to some facility in $\overline{N}(\nu') \cup \overline{N}(\kappa')$, ν and ν' will be connected to different facilities.

This integral solution can be shown to have expected facility cost bounded by F^* and connection cost bounded by $C^* + (2/e)\cdot\mathrm{LP}^*$ (see Appendix A). As a result the expected total cost is bounded by $(1+2/e)\cdot\mathrm{LP}^*$. Summarizing, we obtain the main result of this section.

Theorem 3. *Algorithm* ECHS *is a* $(1 + 2/e)$*-approximation algorithm for* FTFP.

Pseudocode 3 Algorithm ECHS: Constructing Integral Solution

for each $\kappa \in P$ **do**
  choose one $\phi(\kappa) \in \overline{N}(\kappa)$, with each $\mu \in \overline{N}(\kappa)$ chosen as $\phi(\kappa)$ with probability $\bar{y}_\mu$ (note $\bar{x}_{\mu\kappa} = \bar{y}_\kappa$ for all $\mu \in \overline{N}(\kappa)$)
  open $\phi(\kappa)$ and connect κ to $\phi(\kappa)$
for each $\mu \in \overline{\mathbb{F}} - \bigcup_{\kappa \in P} \overline{N}(\kappa)$ **do**
  open μ with probability $\bar{y}_\mu$ (independently)
for each non-primary demand $\nu \in \overline{\mathbb{C}}$ **do**
  if any facility in $\overline{N}(\nu)$ is open **then**
    connect ν to the nearest open facility in $\overline{N}(\nu)$
  else
    connect ν to $\phi(\kappa)$ where κ is ν's primary demand

7 Algorithm EBGS with Ratio 1.575

In this section we give our main result, a 1.575-approximation algorithm for FTFP, where 1.575 is the value of $\min_{\gamma \geq 1} \max\{\gamma, 1+2/e^\gamma, \frac{1/e+1/e^\gamma}{1-1/\gamma}\}$, rounded to three decimal digits. This matches the ratio of the best known LP-rounding algorithm for UFL by Byrka *et al.* [3]. Recall that in Section 6 we showed how to compute 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 balance these two costs, by reducing the connection cost, at the expense of slightly increasing the facility cost.

Our approach can be thought of as a combination of the ideas in [3] with the techniques of demand reduction and adaptive partitioning that we introduced earlier. However, our adaptive partitioning technique needs to be carefully modified because now we will be using a more intricate neighborhood structure, with the neighborhood of each demand divided into two parts, the close and far neighborhood, and with some conditions on which pairs of neighborhoods need to overlap and which need to be disjoint. The final rounding stage that construct an integral solution is a relatively straightforward generalization of the rounding method in [3].

We begin by describing properties that our partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ needs to satisfy. The neighborhood $\overline{N}(\nu)$ of each demand ν will be divided into two disjoint parts. The first part, called the *close neighborhood* $\overline{N}_{\mathrm{cls}}(\nu)$, contains the facilities in $\overline{N}(\nu)$ nearest to ν with the total connection value equal $1/\gamma$. The second part, called the *far neighborhood* $\overline{N}_{\mathrm{far}}(\nu)$, contains the remaining facilities in $\overline{N}(\nu)$. The formal definitions of these sets are given below in Property (NB). The respective average connection costs from ν for these sets are defined by $C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) = \gamma \sum_{\mu \in \overline{N}_{\mathrm{cls}}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu}$ and $C^{\mathrm{avg}}_{\mathrm{far}}(\nu) = \frac{\gamma}{\gamma-1} \sum_{\mu \in \overline{N}_{\mathrm{far}}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu}$. We will also use notation $C^{\max}_{\mathrm{cls}}(\nu) = \max_{\mu \in \overline{N}_{\mathrm{cls}}(\nu)} d_{\mu\nu}$ for the maximum distance from ν to its close neighborhood.

Our partitioned solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ must satisfy the same partitioning and completeness properties as before, namely properties (PS) and (CO) in Section 4. In addition, it must satisfy new neighborhood property (NB) and modified properties (PD') and (SI'), listed below.

(NB) For each demand ν, its neighborhood is divided into *close* and *far* neighborhood, that is $\overline{N}(\nu) = \overline{N}_{\mathrm{cls}}(\nu) \cup \overline{N}_{\mathrm{far}}(\nu)$, where
- $\overline{N}_{\mathrm{cls}}(\nu) \cap \overline{N}_{\mathrm{far}}(\nu) = \emptyset$,
- $\sum_{\mu \in \overline{N}_{\mathrm{cls}}(\nu)} \bar{x}_{\mu\nu} = 1/\gamma$, and
- if $\mu \in \overline{N}_{\mathrm{cls}}(\nu)$ and $\mu' \in \overline{N}_{\mathrm{far}}(\nu)$ then $d_{\mu\nu} \leq d_{\mu'\nu}$.

Note that the second condition, together with (PS.1), implies that $\sum_{\mu \in \overline{N}_{\mathrm{far}}(\nu)} \bar{x}_{\mu\nu} = 1 - 1/\gamma$.

(PD') *Primary demands.* Primary demands satisfy the following conditions:
1. For any two different primary demands $\kappa, \kappa' \in P$ we have $\overline{N}_{\mathrm{cls}}(\kappa) \cap \overline{N}_{\mathrm{cls}}(\kappa') = \emptyset$.
2. For each site $i \in \mathbb{F}$, $\sum_{\kappa \in P} \sum_{\mu \in i \cap \overline{N}_{\mathrm{cls}}(\kappa)} \bar{x}_{\mu\kappa} \leq y_i^*$.
3. Each demand $\nu \in \overline{\mathbb{C}}$ is assigned to one primary demand $\kappa \in P$ such that
   (a) $\overline{N}_{\mathrm{cls}}(\nu) \cap \overline{N}_{\mathrm{cls}}(\kappa) \neq \emptyset$, and

(b) $C^{\text{avg}}_{\text{cls}}(\nu) + C^{\max}_{\text{cls}}(\nu) \geq C^{\text{avg}}_{\text{cls}}(\kappa) + C^{\max}_{\text{cls}}(\kappa)$.

(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 $\overline{N}(\nu') \cap \overline{N}_{\text{cls}}(\kappa) = \emptyset$. In particular, by Property (PD.3(a)), this implies that different sibling demands are assigned to different primary demands.

Modified adaptive partitioning. To obtain a fractional solution with the above properties, we employ a modified adaptive partitioning algorithm. As in Section 4, we have two phases. In Phase 1 we split clients into demands and create facilities on sites, while in Phase 2 we augment each demand's connection values so that its total value is 1.

Phase 1 runs in iterations. Consider any client j. As before, $\widetilde{N}(j)$ is the neighborhood of j with respect to the yet unpartitioned solution, namely the set of facilities μ such that $\widetilde{x}_{\mu j} > 0$. Order the facilities in this set as $\widetilde{N}(j) = \{\mu_1, ..., \mu_q\}$ in order of non-decreasing distance from j, that is $d_{\mu_1 j} \leq d_{\mu_2 j} \leq \ldots \leq d_{\mu_q j}$, where $q = |\widetilde{N}(j)|$. Without loss of generality, there is an index l for which $\sum_{s=1}^{l} \widetilde{x}_{sj} = 1/\gamma$, since we can always split one facility to have this property. Then we define $\widetilde{N}_{\gamma}(j) = \{\mu_1, ..., \mu_l\}$. We also use notation

$$\text{tcc}_{\gamma}(j) = D(\widetilde{N}_{\gamma}(j), j) = \sum\nolimits_{\mu \in \widetilde{N}_{\gamma}(j)} d_{\mu j} \widetilde{x}_{\mu j} \quad \text{and} \quad \text{dmax}_{\gamma}(j) = \max_{\mu \in \widetilde{N}_{\gamma}(j)} d_{\mu j}.$$

In each iteration, we find a not yet exhausted client p that minimizes the value of $\text{tcc}_{\gamma}(p) + \text{dmax}_{\gamma}(p)$. Now we have two cases:

Case 1:: $\widetilde{N}_{\gamma}(p) \cap \overline{N}_{\text{cls}}(\kappa) \neq \emptyset$, for some existing primary demand κ. In this case we assign ν to κ. As before, if there are multiple such κ, we pick any of them. We also fix $\bar{x}_{\mu\nu} \leftarrow \widetilde{x}_{\mu p}, \widetilde{x}_{\mu p} \leftarrow 0$ for each $\mu \in \widetilde{N}(p) \cap \overline{N}_{\text{cls}}(\kappa)$. As before, although we check for overlap between $\widetilde{N}_{\gamma}(p)$ and $\overline{N}_{\text{cls}}(\kappa)$, the facilities we actually move into $\overline{N}(\nu)$ include all facilities in the intersection of $\widetilde{N}(p)$, a bigger set, with $\overline{N}_{\text{cls}}(\kappa)$. We would like to point out that $\overline{N}(\nu)$ is not finalized at this time as we will add more facilities to it in the augment phase. As a result $\overline{N}_{\text{cls}}(\nu)$ is not fixed either, as we could potentially add facilities closer to ν than facilities already in $\overline{N}(\nu)$. Recall that by definition $\overline{N}_{\text{cls}}(\nu)$ consists of the facilities that closest to ν once $\overline{N}(\nu)$ is fixed with total connection value of 1.

Case 2:: $\widetilde{N}_{\gamma}(p) \cap \overline{N}_{\text{cls}}(\kappa) = \emptyset$, for all existing primary demands κ. In this case we make ν a primary demand. We then fix $\bar{x}_{\mu\nu} \leftarrow \widetilde{x}_{\mu p}$ for $\mu \in \widetilde{N}_{\gamma}(p)$ and set the corresponding $\widetilde{x}_{\mu p}$ to 0. Note that the total connection value in $\overline{N}_{\text{cls}}(\nu)$ is now exactly $1/\gamma$. The set $\widetilde{N}_{\gamma}(p)$ turns out to coincide with $\overline{N}_{\text{cls}}(\nu)$ as the facilities in $\widetilde{N}(p) \setminus \widetilde{N}_{\gamma}(p)$ are all farther away than any facilitity in $\widetilde{N}_{\gamma}(p)$. In the augmenting phase, Phase 2, we have available only facilities in some subset of $\widetilde{N}(p) \setminus \widetilde{N}_{\gamma}(p)$. Thus $\overline{N}_{\text{cls}}(\nu)$ is defined when ν is created.

Once all clients are exhausted, that is, each client j has r_j demands created, Phase 1 concludes. We then do Phase 2, the augmenting phase. For each demand ν of client j with total connection value less than 1, we use our AugmentToUnit() procedure to add additional facilities from $\widetilde{N}(j)$ to ν's neighborhood to make its total connection value equal 1, as before a facility is removed from $\widetilde{N}(j)$ once added to a demand's neighborhood. We do facility split if necessary to make $\overline{N}(\nu)$ have total connection value of 1. This completes the description of the partitioning algorithm.

We now argue that the fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ satisfies all the stated properties. Properties (PS), (CO), (NB), (PD'.1) and (SI'.1) are directly enforced by the adaptive partitioning algorithm. The proofs for other properties (PD'.2), (PD'.3(b)) and (SI'.2) are similar to those in Section 4, with the exception of (PD.3(a)), which we justify below.

The argument for (PD.3(a)) is a bit subtle, because of possible complications arising in the augmenting phase, Phase 2. This phase does not change close neighborhoods of primary demands, as each primary demand already contains all the nearest facilities with total connection value $1/\gamma$. For non-primary demands, however, $\overline{N}(\nu)$, for $\nu \in j$, takes all 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 augmenting phase might actually be closer to ν than some of the facilities already in

$\overline{N}(\nu)$. As a result, facilities added in the augmenting phase, Phase 2, might appear in $\overline{N}_{\text{cls}}(\nu)$, yet they are not in $\overline{N}_{\text{cls}}(\kappa)$, the close neighborhood of the primary demand κ that ν is assigned to. Nevertheless, we show that Property (PD.3(a)) holds.

Consider an iteration when we create a demand $\nu \in p$ and assign it to κ. Then the set $B(p) = \widetilde{N}_\gamma(p) \cap \overline{N}_{\text{cls}}(\kappa)$ is not empty. 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 the facility split. Here $\widetilde{N}(p)$ refers to the neighborhood of client p just before ν was created. For an arbitrary set of facilities A define $\text{dmax}(A,\nu)$ as the minimum distance τ such that $\sum_{\mu\in A\,:\,d_{\mu\nu}\leq\tau} \bar{y}_\mu \geq 1/\gamma$. Adding additional facilities into A cannot make $\text{dmax}(A,\nu)$ larger, so it follows that $\text{dmax}(\overline{N}_{\text{cls}}(\nu),\nu) \geq \text{dmax}(\widetilde{N}(p),\nu)$, because $\overline{N}_{\text{cls}}(\nu)$ is a subset of $\widetilde{N}(p)$. Since we have $d_{\mu\nu} = d_{\mu p}$ by definition, it is easy to see that every $\mu \in B(p)$ satisfies $d_{\mu\nu} \leq \text{dmax}(\widetilde{N}(p),\nu) \leq \text{dmax}(\overline{N}_{\text{cls}}(\nu),\nu)$ 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 defining $\overline{N}_{\text{cls}}(\nu)$. Finally, since $B(p) \neq \emptyset$, we have that the close neighborhood of a demand ν and its primary demand κ must overlap.

Algorithm EBGS. The complete algorithm starts with solving the linear program and computing the partitioning described earlier in this section. 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.

Next, we connect demands to facilities. 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 its *target facility* $\phi(\kappa)$, where κ is the primary demand that ν is assigned to.

Analysis. The feasibility of our integral solution follows from Properties (SI.1), (SI.2), and (PD.1), as these properties together ensure that each facility is accessible to at most one demand among sibling demands of the same client, regardless whether a demand connects to its neighbor or its target facility.

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 $\bar{y}_\mu$). The expected connection cost can be bounded by $C^* \max\{\frac{1/e+1/e^\gamma}{1-1/\gamma}, 1+\frac{2}{e^\gamma}\}$ (see Appendix B). 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.

Theorem 4. *Algorithm* EBGS *is a* 1.575*-approximation algorithm for* FTFP*.*

8 Final Comments

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 from Section 7, following the approach by Li [13], we could further improve the ratio to below 1.575. This is not enough, however, to match the 1.488 bound for UFL in [13], because matching this bound also requires appropriately extending dual-fitting algorithms [15] 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 work was partly motivated by this question. The techniques we introduced are not directly applicable to FTFL, 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 that further refinements of our construction might get around this issue and lead to new algorithms for FTFL with improved ratios.

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. Kewen Liao and Hong Shen. Unconstrained and constrained fault-tolerant resource allocation. In *Proceedings of the 17th Annual International Conference on Computing and Combinatorics*, COCOON'11, pages 555–566, 2011.
15. Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Approximation algorithms for metric facility location problems. *SIAM J. Comput.*, 36(2):411–432, 2006.
16. David Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing, STOC '97*, pages 265–274, 1997.
17. 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.
18. Chaitanya Swamy and David Shmoys. Fault-tolerant facility location. *ACM Trans. Algorithms*, 4(4):1–27, 2008.
19. Jens Vygen. *Approximation Algorithms for Facility Location Problems*. Forschungsinst. für Diskrete Mathematik, 2005.
20. 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.
21. Li Yan and Marek Chrobak. Approximation algorithms for the fault-tolerant facility placement problem. *Inf. Process. Lett.*, 111(11):545–549, 2011.

A Approximation Ratio Analysis of Algorithm ECHS

In this section we show that the expected facility cost is bounded by F^* and the expected connection cost is bounded by $C^* + (2/e) \cdot \mathrm{LP}^*$. As a result the expected total cost is bounded by $(1 + 2/e) \cdot \mathrm{LP}^*$.

By (PD.1), 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 ECHS is

$$\mathrm{Exp}[F_{\mathrm{ECHS}}] = \sum\nolimits_{\mu\in\overline{\mathbb{F}}} f_\mu \bar{y}_\mu = \sum\nolimits_{i\in\mathbb{F}} f_i \sum\nolimits_{\mu\in i} \bar{y}_\mu = \sum\nolimits_{i\in\mathbb{F}} f_i y_i^* = F^*,$$

where the third equality follows from (PS.3).

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 its target facility $\phi(\kappa) \in \overline{N}(\kappa)$, where κ is the primary demand to which ν is assigned.

We now estimate the expected cost $d_{\phi(\kappa)\nu}$ of the indirect connection. Let Λ_ν denote the event that none of the facilities in $\overline{N}(\nu)$ is opened. Then

$$\mathrm{Exp}[d_{\phi(\kappa)\nu} | \Lambda_\nu] = D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \nu),$$

where $D(A,\sigma) \stackrel{\mathrm{def}}{=} \sum_{\mu\in A} d_{\mu\sigma}\bar{y}_\mu / \sum_{\mu\in A} \bar{y}_\mu$, for any set A of facilities and a demand σ. Note that $C_\nu^{\mathrm{avg}} = D(\overline{N}(\nu),\nu)$, and that Λ_ν implies that $\overline{N}(\kappa)\setminus\overline{N}(\nu) \neq \emptyset$, since $\overline{N}(\kappa)$ contains at least one open facility, namely $\phi(\kappa)$.

Lemma 7. *Let ν be a demand assigned to a primary demand κ, and assume that $\overline{N}(\kappa) \setminus \overline{N}(\nu) \neq \emptyset$. Then $\mathrm{Exp}[d_{\phi(\kappa)\nu} | \Lambda_\nu] \le C_\nu^{\mathrm{avg}} + 2\alpha_\nu^*$.*

Proof. By the discussion above, we need to show that $D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \nu) \le C^{\mathrm{avg}}(\nu) + 2\alpha_\nu^*$. There are two cases to consider.

Case 1: There exists some $\mu' \in \overline{N}(\kappa) \cap \overline{N}(\nu)$ such that $d_{\mu'\kappa} \le C_\kappa^{\mathrm{avg}}$. In this case, for every $\mu \in \overline{N}(\kappa)\setminus\overline{N}(\nu)$, we have

$$d_{\mu\nu} \le d_{\mu\kappa} + d_{\mu'\kappa} + d_{\mu'\nu} \le \alpha_\kappa^* + C_\kappa^{\mathrm{avg}} + \alpha_\nu^* \le C_\nu^{\mathrm{avg}} + 2\alpha_\nu^*,$$

using the triangle inequality, complementary slackness, and (PD.3(b)). By summing over all $\mu \in \overline{N}(\kappa) \setminus \overline{N}(\nu)$, it follows that $D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \nu) \le C_\nu^{\mathrm{avg}} + 2\alpha_\nu^*$.

Case 2: Every $\mu' \in \overline{N}(\kappa) \cap \overline{N}(\nu)$ has $d_{\mu'\kappa} > C_\kappa^{\mathrm{avg}}$. Then we have $D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \kappa) \le D(\overline{N}(\kappa), \kappa) = C_\kappa^{\mathrm{avg}}$. Therefore, choosing an arbitrary $\mu' \in \overline{N}(\kappa) \cap \overline{N}(\nu)$, we obtain

$$D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \nu) \le D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \kappa) + d_{\mu'\kappa} + d_{\mu'\nu} \le C_\kappa^{\mathrm{avg}} + \alpha_\kappa^* + \alpha_\nu^* \le C_\nu^{\mathrm{avg}} + 2\alpha_\nu^*,$$

where we again use the triangle inequality, complementary slackness, and (PD.3(b)).

Since the lemma holds in both cases, the proof is now complete.

We now continue our estimation of the connection cost. We note first that for a primary demand κ, its expected connection cost is $C_\kappa^{\mathrm{avg}} = \sum_{\mu\in\overline{N}(\kappa)} d_{\mu\kappa}\bar{x}_{\mu\kappa}$ as in the previous section.

Next, we consider a non-primary demand ν. Denote by C_ν the random variable representing the connection cost for ν. We first deal with the case when $\overline{N}(\kappa) \setminus \overline{N}(\nu) = \emptyset$, which is the same as $\overline{N}(\kappa) \subseteq \overline{N}(\nu)$. This actually implies that $\overline{N}(\kappa) = \overline{N}(\nu)$ because Property (CO) implies that $\bar{x}_{\mu\nu} = \bar{y}_\mu = \bar{x}_{\mu\kappa}$ for every $\mu \in \overline{N}(\kappa)$ and we have $\sum_{\mu\in\overline{N}(\kappa)} \bar{x}_{\mu\nu} = 1$ due to (PS.1). Thus $\bar{x}_{\mu\nu} = \bar{y}_\mu = \bar{x}_{\mu\kappa}$ for $\mu \in \overline{N}(\kappa)$ and $\bar{x}_{\mu\nu} = 0$ for $\mu \in \overline{\mathbb{F}} \setminus \overline{N}(\kappa)$. As a result we have $\mathrm{Exp}[C_\nu] = C_\nu^{\mathrm{avg}}$.

Now we bound the expected connection cost of ν when $\overline{N}(\kappa) \setminus \overline{N}(\nu) \neq \emptyset$. Let $\overline{N}(\nu) = \{\mu_1, \ldots, \mu_l\}$ and let $d_s = d_{\mu_s\nu}$ and $y_s = \bar{y}_{\mu_s}$ for $s = 1, \ldots, l$. By reordering, we can assume that $d_1 \le d_2 \le \ldots \le d_l$. By

Pseudocode 3, the connection cost is no more than that obtained through the random process that opens each μ_s independently with probability y_s (note that $\bar{x}_{\mu_s\nu} = y_s$ because $\bar{x}_{\mu_s\nu} > 0$ and by (CO)), and connects ν to the nearest such open facility, if any of them opens; otherwise ν is connected indirectly to its target facility $\phi(\kappa)$. The intuition is that we only use a facility μ_s if none of $\mu_1, \ldots, \mu_{s-1}$ is open. We know that μ_s opens (unconditionally) with probability y_s. The only way that some of $\mu_1, \ldots, \mu_{s-1}$ can affect that probability is if they belong to $\overline{N}(\kappa)$ for some primary demand κ and it also happens to be that $\mu_s \in \overline{N}(\kappa)$ as well. However in this case, the condition that they are closed actually implies that the conditional probability of μ_s being open is larger than y_s. (For a detailed proof, see [5].)

Putting it all together, we estimate the (unconditional) expected connection cost of ν as follows:

$$
\begin{aligned}
\mathrm{Exp}[C_\nu] &\leq \sum\nolimits_{r=1}^{l} d_r y_r \prod\nolimits_{s=1}^{r-1}(1-y_s) + \mathrm{Exp}[d_{\phi(\kappa)\nu} | \Lambda_\nu] \cdot \prod\nolimits_{s=1}^{l}(1-y_s) \\
&\leq \Big(1 - \prod\nolimits_{s=1}^{l}(1-y_s)\Big) \cdot \sum\nolimits_{r=1}^{l} d_r y_r + \mathrm{Exp}[d_{\phi(\kappa)\nu} | \Lambda_\nu] \cdot \prod\nolimits_{s=1}^{l}(1-y_s) \\
&\leq (1 - \frac{1}{e}) \sum\nolimits_{r=1}^{l} d_r y_r + \frac{1}{e} \mathrm{Exp}[d_{\phi(\kappa)\nu} | \Lambda_\nu] \leq (1 - \frac{1}{e}) C_\nu^{\mathrm{avg}} + \frac{1}{e}(C_\nu^{\mathrm{avg}} + 2\alpha_\nu^*) = C_\nu^{\mathrm{avg}} + \frac{2}{e}\alpha_\nu^*,
\end{aligned}
$$

where the second inequality is shown in the appendix (see also [5]) and the last inequality follows from Lemma 7.

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

$$\mathrm{Exp}[C_j] \leq \sum\nolimits_{\nu \in j}(C_\nu^{\mathrm{avg}} + \tfrac{2}{e}\alpha_\nu^*) = C_j^* + \tfrac{2}{e} r_j \alpha_j^*.$$

Finally, summing over all clients j, we obtain our bound on the expected connection cost,

$$\mathrm{Exp}[C_{\mathrm{ECHS}}] \leq C^* + \frac{2}{e}\mathrm{LP}^*.$$

Therefore, we have established that our algorithm constructs a feasible integral solution with an overall expected cost

$$\mathrm{Exp}[F_{\mathrm{ECHS}} + C_{\mathrm{ECHS}}] \leq F^* + C^* + \frac{2}{e} \cdot \mathrm{LP}^* = (1 + 2/e) \cdot \mathrm{LP}^* \leq (1 + 2/e) \cdot \mathrm{OPT}.$$

B Analysis of Connection Cost of Algorithm EBGS

In this section we bound the connection cost of Algorithm EBGS. Properties (PD.3(a)) and (PD.3(b)) allow us to bound the expected distance from a demand ν to its target facility by $C_{\mathrm{cls}}^{\mathrm{avg}}(\nu) + C_{\mathrm{cls}}^{\max}(\nu) + C_{\mathrm{far}}^{\mathrm{avg}}(\nu)$, in the event that none of ν's neighbors opens, using a similar argument as Lemma 2.2 in [3] [2]. We are then able to estimate the expected connection cost for demand ν using an argument similar to [3]: with probability no less than $1 - 1/e$, ν has some facility open in its close neighborhood, with probability no less than $1 - 1/e^\gamma$, ν has some facility open in its overall neighborhood, and with probability no more than $1/e^\gamma$, ν will connect to its target facility. This gives us the bound

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

where $\rho_\nu = C_{\mathrm{cls}}^{\mathrm{avg}}(\nu)/C^{\mathrm{avg}}(\nu)$. It is easy to see that ρ_ν is between 0 and 1. Since $\sum_{\nu \in j} C^{\mathrm{avg}}(\nu) = \sum_{\nu \in j}\sum_{\mu \in \mathbb{F}} d_{\mu\nu}\bar{x}_{\mu\nu} = \sum_{i \in \mathbb{F}} d_{ij}x_{ij}^* = C_j^*$, summing over all clients j we have total connection cost bounded by $C^* \max\{\frac{1/e + 1/e^\gamma}{1 - 1/\gamma}, 1 + \frac{2}{e^\gamma}\}$.

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

C An Elementary Proof of the Inequality in Section 6

In the $1 + 2/e = 1.736$-approximation in Section 6 we need to show the following inequality

$$\sum_{r=1}^{l} d_r y_r \prod_{s=1}^{r-1}(1 - y_s) \le \left(1 - \prod_{s=1}^{l}(1 - y_s)\right) \cdot \sum_{r=1}^{l} d_r y_r \tag{3}$$

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

In this section we give a new proof of this inequality, much simpler than the existing proof in [5], and also simpler than the argument by Sviridenko [17]. We derive this inequality from the following generalized version of the Chebyshev Sum Inequality:

$$\sum_i p_i \sum_j p_j a_j b_j \le \sum_i p_i a_i \sum_j p_j b_j, \tag{4}$$

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

Given inequality (4), we can obtain our inequality (3) by simple substitution

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

For the sake of completeness, we include the proof of inequality (4), due to Hardy, Littlewood and Polya [9]. The idea is to evaluate the following sum:

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

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