LP-rounding Algorithms for the Fault-Tolerant Facility Placement Problem[☆]

Li Yan and Marek Chrobak[1,*]

Department of Computer Science
University of California at Riverside
Riverside, CA 92521, USA

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 fractional solution 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.

Keywords: Facility Location, Approximation Algorithms

[☆] A preliminary version of this article appeared in Proc. CIAC 2013.

[*] Corresponding author

Email address: `{lyan,marek}@cs.ucr.edu` (Li Yan and Marek Chrobak)

[1] Work supported by NSF grants CCF-0729071 and CCF-1217314.

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 $j \in \mathbb{C}$ has demand r_j. Building one facility at a site $i \in \mathbb{F}$ incurs a cost f_i, and connecting one unit of demand from client j to a facility at site i 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 any two 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 on it, 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 $\max_{j\in\mathbb{C}} r_j \le |\mathbb{F}|$, 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 [1] proposed an approach based on LP-rounding, that they used to achieve a ratio of 3.16. This was then improved by Chudak [2] to 1.736, and later by Sviridenko [3] to 1.582. The best known "pure" LP-rounding algorithm is due to Byrka *et al.* [4] with ratio 1.575. Byrka and Aardal [5] gave a hybrid algorithm that combines LP-rounding and dual-fitting (based on [6]), achieving a ratio of 1.5. Recently, Li [7] showed that, with a more refined analysis and randomizing the scaling parameter used in [5], 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 [8], the dual fitting method by Jain *et al.* [6] that gives ratio 1.61, and a local search heuristic by Arya *et al.* [9] 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 $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{O(\log\log n)})$ [10]. An observation by Sviridenko strengthened the underlying assumption to $\mathsf{P} \neq \mathsf{NP}$ (see [11]).

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.* [13], adapted the approach for UFL from [1]. Swamy and Shmoys [14] improved the ratio to 2.076 using the idea of pipage rounding introduced in [3]. Most recently, Byrka *et al.* [15] 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 [16], who extended the dual-fitting algorithm from [6] 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 [16] is flawed[2]. To date, the best approximation ratio for FTFP in the literature is 3.16, established by Yan and Chrobak [17], while the only known lower bound is the 1.463 lower bound for UFL from [10], as UFL is a special case of FTFP. 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 [18]).

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 [4] and significantly improving our earlier bound in [17]. 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 then using the algorithm from [15] to solve the FTFL instance.

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 [19, 2, 4] 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 divided between the r_j unit demands of client j in some way. In Section 4 we formulate a set of properties required for this partitioning to work. For example, one property guarantees that we can connect demands to facilities so that two demands from the same client are connected to different facilities. Then we present our *adaptive partitioning* technique that computes a partitioning with all the desired properties. Using adaptive partitioning we were able to extend the algorithms for UFL from [19, 2, 4] 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. 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,

[2]Confirmed through private communication with the authors.

as Li showed in [7] for UFL. (Since the ratio 1.488 for UFL in [7] uses also dual-fitting algorithms [20], 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:

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

The dual program is:

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

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 $\mathrm{LP}^* = \mathit{cost}(\boldsymbol{x}^*, \boldsymbol{y}^*) = F^* + C^*$ 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.

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 [2], 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^*, as well as OPT, 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

$$\begin{aligned}&y^*_{i'} \leftarrow y^*_i - x^*_{ik},\ y^*_i \leftarrow x^*_{ik}, \quad \text{and}\\ &x^*_{i'j} \leftarrow \max(x^*_{ij} - x^*_{ik}, 0),\ x^*_{ij} \leftarrow \min(x^*_{ij}, x^*_{ik}) \quad \text{for all } j \neq k.\end{aligned}$$

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 described above 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 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

$$\begin{aligned}\hat{y}_i &\leftarrow \lfloor y^*_i \rfloor, \quad \hat{x}_{ij} \leftarrow \lfloor x^*_{ij} \rfloor \quad \text{and}\\ \dot{y}_i &\leftarrow y^*_i - \lfloor y^*_i \rfloor, \quad \dot{x}_{ij} \leftarrow x^*_{ij} - \lfloor x^*_{ij} \rfloor\end{aligned}$$

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* $\rho \ge 1$. *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 $\bar{\boldsymbol{y}} = (\bar{y}_\mu)$. As before, 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. By definition we have $P \subseteq \overline{\mathbb{C}}$. 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 $i \in \mathbb{F}$, $\sum_{\mu\in i}\sum_{\kappa\in P} \bar{x}_{\mu\kappa} \leq y^*_i$.
3. Each demand $\nu \in \overline{\mathbb{C}}$ is assigned to one primary demand $\kappa \in P$ such that
   (a) $\overline{N}(\nu) \cap \overline{N}(\kappa) \neq \emptyset$, and
   (b) $C^{\mathrm{avg}}_\nu + \alpha^*_\nu \geq C^{\mathrm{avg}}_\kappa + \alpha^*_\kappa$.

(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. 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. We would like to point out that the adaptive partitioning process for the 1.575-approximation algorithm (Section 7) is more subtle than that for 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 partitioning 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 $\bar{x}_{\mu\nu}$ 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 we preserve the completeness (CO) 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 it 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 (Lines 5–8), 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 nondecreasing 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 the completeness condition (c2).) 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 $\mu$ at $i$ and add $\mu$ 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 $\nu$ 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 $\nu$ to $\kappa$
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 $\nu$ primary, $P \leftarrow P \cup \{\nu\}$, assign $\nu$ 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

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 $d_{\mu_1 j} \le d_{\mu_2 j} \le \ldots \le d_{\mu_q j}$
3:      Let $l$ be such that $\sum_{k=1}^{l} \bar{y}_{\mu_k} \ge 1$ and $\sum_{k=1}^{l-1} \bar{y}_{\mu_k} < 1$
4:      Create a new facility $\sigma$ at the same site as $\mu_l$ and add it to $\overline{\mathbb{F}}$      ▷ split $\mu_l$
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($\nu, j, \overline{\mathbb{F}}, \widetilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$)      ▷ $\nu$ 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 $\eta$ be any facility such that $\widetilde{x}_{\eta j} > 0$
13:         if $1 - \sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu} \ge \widetilde{x}_{\eta j}$ then
14:             $\bar{x}_{\eta\nu} \leftarrow \widetilde{x}_{\eta j}, \widetilde{x}_{\eta j} \leftarrow 0$
15:         else
16:             Create a new facility $\sigma$ at the same site as $\eta$ and add it to $\overline{\mathbb{F}}$      ▷ split $\eta$
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 $\nu' \ne \nu$ with $\bar{x}_{\eta\nu'} > 0$, set $\bar{x}_{\eta\nu'} \leftarrow \bar{y}_\eta,\ \bar{x}_{\sigma\nu'} \leftarrow \bar{y}_\sigma$
20:             For each $j' \ne j$ with $\widetilde{x}_{\eta j'} > 0$, set $\widetilde{x}_{\eta j'} \leftarrow \bar{y}_\eta, \widetilde{x}_{\sigma j'} \leftarrow \bar{y}_\sigma$
21:             (All other new connection values are set to 0)
```

(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 the total connection value for ν, that is $\mathrm{conn}(\nu) \overset{\text{def}}{=} \sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu}$, is equal 1. This is accomplished by Procedure AugmentToUnit() (definition in Pseudocode 2, Lines 10–21) that allocates to $\nu \in j$ some of the remaining connection values $\widetilde{x}_{\mu j}$ of client j (Lines 19–21). AugmentToUnit() will repeatedly pick any facility η with $\widetilde{x}_{\eta j} > 0$. If $\widetilde{x}_{\eta j} \le 1 - \mathrm{conn}(\nu)$, then the connection value $\widetilde{x}_{\eta j}$ is reassigned to ν. Otherwise, $\widetilde{x}_{\eta j} > 1 - \mathrm{conn}(\nu)$, in which case we split η so that connecting ν to one of the created copies of η will make $\mathrm{conn}(\nu)$ equal 1, and we'll be done.

Notice that we start with $|\mathbb{F}|$ facilities and in each iteration of the while loop in Line 5 (Pseudocode 1) 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.

Example. We now illustrate our partitioning algorithm with an example, where the FTFP instance has four sites and four clients. The demands are $r_1 = 1$ and $r_2 = r_3 = r_4 = 2$. The facility costs are $f_i = 1$ for all i. The distances are defined as follows: $d_{ii} = 3$ for $i = 1, 2, 3, 4$ and $d_{ij} = 1$ for all $i \neq j$. Solving the LP(1), we obtain the fractional solution given in Table 1a.

x^*_{ij}	1	2	3	4	y^*_i
1	0	$\frac{4}{3}$	$\frac{4}{3}$	$\frac{4}{3}$	$\frac{4}{3}$
2	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
3	$\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$
4	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$

(a)

$\bar{x}_{\mu\nu}$	$1'$	$2'$	$2''$	$3'$	$3''$	$4'$	$4''$	$\bar{y}_\mu$
$\dot{1}$	0	1	0	1	0	1	0	1
$\ddot{1}$	0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$
$\dot{2}$	$\frac{1}{3}$	0	0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$
$\dot{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	0	$\frac{1}{3}$	$\frac{1}{3}$
$\dot{4}$	$\frac{1}{3}$	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	$\frac{1}{3}$

(b)

Table 1: An example of an execution of the partitioning algorithm. (a) An optimal fractional solution x^*, y^*. (b) The partitioned solution. j' and j'' denote the first and second demand of a client j, and $\dot{i}$ and $\ddot{i}$ denote the first and second facility at site i.

It is easily seen that the fractional solution in Table 1a is optimal and complete ($x^*_{ij} > 0$ implies $x^*_{ij} = y^*_i$). The dual optimal solution has all $\alpha^*_j = 4/3$ for $j = 1, 2, 3, 4$.

Now we perform Phase 1, the adaptive partitioning, following the description in Pseudocode 1. To streamline the presentation, we assume that all ties are broken in favor of lower-numbered clients, demands or facilities. First we create one facility at each of the four sites, denoted as $\dot{1}$, $\dot{2}$, $\dot{3}$ and $\dot{4}$ (Line 2–4, Pseudocode 1). We then execute the "while" loop in Line 5 Pseudocode 1. This loop will have seven iterations. Consider the first iteration. In Line 7–8 we compute $\mathrm{tcc}(j)$ for each client $j = 1, 2, 3, 4$ in U. When computing $\widetilde{N}_1(2)$, facility $\dot{1}$ will get split into $\dot{1}$ and $\ddot{1}$ with $\bar{y}_{\dot{1}} = 1$ and $\bar{y}_{\ddot{1}} = 1/3$. (This will happen in Line 4–7 of Pseudocode 2.) Then, in Line 9 we will pick client $p = 1$ and create a demand denoted as $1'$ (see Table 1b). Since there are no primary demands yet, we make $1'$ a primary demand with $\overline{N}(1') = \widetilde{N}_1(1) = \{\dot{2}, \dot{3}, \dot{4}\}$. Notice that client 1 is exhausted after this iteration and U becomes $\{2, 3, 4\}$.

In the second iteration we compute $\mathrm{tcc}(j)$ for $j = 2, 3, 4$ and pick client $p = 2$, from which we create a new demand $2'$. We have $\widetilde{N}_1(2) = \{\dot{1}\}$, which is disjoint from $\overline{N}(1')$. So we create a demand $2'$ and make it primary, and set $\overline{N}(2') = \{\dot{1}\}$. In the third iteration we compute $\mathrm{tcc}(j)$ for $j = 2, 3, 4$ and again we pick client $p = 2$. Since $\widetilde{N}_1(2) = \{\ddot{1}, \dot{3}, \dot{4}\}$ overlaps with

$\overline{N}(1')$, we create a demand $2''$ and assign it to $1'$. We also set $\overline{N}(2'') = \overline{N}(1') \cap \widetilde{N}(2) = \{\dot{3}, \dot{4}\}$. After this iteration client 2 is exhausted and we have $U = \{3, 4\}$.

In the fourth iteration we compute $\mathrm{tcc}(j)$ for client $j = 3, 4$. We pick $p = 3$ and create demand $3'$. Since $\widetilde{N}_1(3) = \{\dot{1}\}$ overlaps $\overline{N}(2')$, we assign $3'$ to $2'$ and set $\overline{N}(3') = \{\dot{1}\}$. In the fifth iteration we compute $\mathrm{tcc}(j)$ for client $j = 3, 4$ and pick $p = 3$ again. At this time $\widetilde{N}_1(3) = \{\ddot{1}, \dot{2}, \dot{4}\}$, which overlaps with $\overline{N}(1')$. So we create a demand $3''$ and assign it to $1'$, as well as set $\overline{N}(3'') = \{\dot{2}, \dot{4}\}$.

In the last two iterations we will pick client $p = 4$ twice and create demands $4'$ and $4''$. For $4'$ we have $\widetilde{N}_1(4) = \{\dot{1}\}$ so we assign $4'$ to $2'$ and set $\overline{N}(4') = \{\dot{1}\}$. For $4''$ we have $\widetilde{N}_1(4) = \{\ddot{1}, \dot{2}, \dot{3}\}$ and we assign it to $1'$, as well as set $\overline{N}(4'') = \{\dot{2}, \dot{3}\}$.

Now that all clients are exhausted we perform Phase 2, the augmenting phase, to construct a fractional solution in which all demands have total connection value equal to 1. We iterate through each of the seven demands created, that is $1', 2', 2'', 3', 3'', 4', 4''$. $1'$ and $2'$ already have neighborhoods with total connection value of 1, so nothing will change in the first two iterations. $2''$ has $\dot{3}, \dot{4}$ in its neighborhood, with total connection value of $2/3$, and $\widetilde{N}(2) = \{\ddot{1}\}$ at this time, so we add $\ddot{1}$ into $\overline{N}(2'')$ to make $\overline{N}(2'') = \{\ddot{1}, \dot{3}, \dot{4}\}$ and now $2''$ has total connection value of 1. Similarly, $3''$ and $4''$ each get $\ddot{1}$ added to their neighborhood and end up with total connection value of 1. The other two demands, namely $3'$ and $4'$, each have $\dot{1}$ in its neighborhood so each of them has already its total connection value equal 1. This completes Phase 2.

The final partitioned fractional solution is given in Table 1b. We have created a total of five facilities $\dot{1}, \ddot{1}, \dot{2}, \dot{3}, \dot{4}$, and seven demands, $1', 2', 2'', 3', 3'', 4', 4''$. It can be verified that all the stated properties are satisfied.

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. Condition (PS.1) is a result of calling Procedure AugmentToUnit() in Line 21. 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 (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 (Lines 14–16), 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 $\sum_{\mu\in i}\sum_{\kappa\in P}\bar{x}_{\mu\kappa} = \sum_{\mu\in K}\bar{x}_{\mu\kappa_\mu} = \sum_{\mu\in K}\bar{y}_\mu \le \sum_{\mu\in i}\bar{y}_\mu = y_i^*$.

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 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 for (PD.3(b)) to Lemma 5.

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 Lines 11–13 of the algorithm. 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 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 $\nu_1,\dots,\nu_{r_j}$ be the demands of a client $j\in\mathbb{C}$, listed in the order of creation, and, for each $q=1,2,\dots,r_j$, denote by κ_q the primary demand that ν_q is assigned to. After the completion of Phase 1 of Pseudocode 1 (Lines 5–18), we have $\overline{N}(\nu_s)\subseteq\overline{N}(\kappa_s)$ for $s=1,\dots,r_j$. 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.

After Phase 1 all neighborhoods $\overline{N}(\kappa_s)$, $s=1,\dots,r_j$ 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 $s=1,\dots,r_j$, by the way we allocate facilities in Lines 13 and 16. 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 19–21 of Pseudocode 1), all the required disjointness conditions will be preserved. □

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 4. *Let η and ν be two demands, with η created no later than ν, and let $j\in\mathbb{C}$ be a client that is not exhausted when ν is created. Then we have*

(a) $\mathrm{tcc}^\eta(j)\le\mathrm{tcc}^\nu(j)$, *and*

(b) if $\nu\in j$ *then* $\mathrm{tcc}^\eta(j)\le C^{\mathrm{avg}}_\nu$.

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 7. Let $\widetilde{N}(j)=\{\mu_1,\dots,\mu_q\}$ with all facilities μ_s ordered according to nondecreasing distance from j. Consider the following linear program:

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

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^{avg}_{ν} 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 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

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

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 $C^{\mathrm{avg}}_{\kappa}$ 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 9 in Pseudocode 1. The last inequality holds because $\alpha^*_j = \alpha^*_{\nu}$ (due to $\nu \in j$), and because $\mathrm{tcc}^{\kappa}(j) \leq C^{\mathrm{avg}}_{\nu}$, which follows from Lemma 4. □

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. Although this is still quite far from our best ratio 1.575 that we derive later, we include this algorithm in the paper to illustrate, in a relatively simple setting, how the properties of our partitioned fractional solution are used in rounding it to an integral solution with cost not too far away from an optimal solution. The rounding approach we use here is an extension of the corresponding method for UFL described in [19].

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 de-randomized 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 [2], 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. For a primary demand κ, its expected connection cost is C_κ^{avg} 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

$$d_{\phi(\kappa)\nu} \leq d_{\phi(\kappa)\kappa} + d_{\mu\kappa} + d_{\mu\nu} \leq d_{\phi(\kappa)\kappa} + \alpha^*_\kappa + \alpha^*_\nu.$$

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

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

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

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

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

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

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 $F^* + C^* + 2\cdot\mathrm{LP}^* = 3\cdot\mathrm{LP}^* \leq 3\cdot\mathrm{OPT}$. □

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 could be far from the optimum, since we connect a non-primary demand to a facility in the neighborhood of its assigned primary demand and then estimate the distance using the triangle inequality. The basic idea to improve the estimate of the connection cost, following the approach of Chudak and Shmoys [2], is to connect each non-primary demand to its nearest neighbor when one is available, and to only use the facility opened by its assigned primary demand when none of its neighbors 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).

We start, as before, by opening exactly one facility $\phi(\kappa)$ in the neighborhood of each primary demand κ (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 ECHS we may be able to find an open facility in ν's neighborhood and connect ν to this facility. Specifically, the two changes in the algorithm are as follows:

(1) Each facility μ that is not in the neighborhood of any primary demand is opened, independently, with probability $\bar{y}_\mu$ (Lines 4–5). Notice that if $\bar{y}_\mu > 0$ then, due to completeness of the partitioned fractional solution, we have $\bar{y}_\mu = \bar{x}_{\mu\nu}$ for some demand ν. This implies that $\bar{y}_\mu \le 1$, because $\bar{x}_{\mu\nu} \le 1$, by (PS.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 $\overline{N}(\nu)$ has an open facility, we connect ν to the closest open facility in $\overline{N}(\nu)$ (Line 8). Otherwise, we connect ν to its target facility (Line 10).

Pseudocode 3 Algorithm ECHS: Constructing Integral Solution

```
1:  for each $\kappa \in P$ do
2:      choose one $\phi(\kappa) \in \overline{N}(\kappa)$, with each $\mu \in \overline{N}(\kappa)$ chosen as $\phi(\kappa)$ with probability $\bar{y}_\mu$
3:      open $\phi(\kappa)$ and connect $\kappa$ to $\phi(\kappa)$
4:  for each $\mu \in \overline{\mathbb{F}} - \bigcup_{\kappa \in P} \overline{N}(\kappa)$ do
5:      open $\mu$ with probability $\bar{y}_\mu$ (independently)
6:  for each non-primary demand $\nu \in \overline{\mathbb{C}}$ do
7:      if any facility in $\overline{N}(\nu)$ is open then
8:          connect $\nu$ to the nearest open facility in $\overline{N}(\nu)$
9:      else
10:         connect $\nu$ to $\phi(\kappa)$ where $\kappa$ is $\nu$'s assigned 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. 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 $\kappa \neq \kappa'$. 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.

We now show that the expected cost of the computed solution 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 [2]. 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 first estimate the expected cost $d_{\phi(\kappa)\nu}$ of the indirect connection. Let Λ^ν denote the event that some facility in $\overline{N}(\nu)$ is opened. Then

$$\mathrm{Exp}[d_{\phi(\kappa)\nu} \mid \neg\Lambda^\nu] = D(\overline{N}(\kappa)\setminus\overline{N}(\nu),\nu), \tag{3}$$

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

Lemma 9. *Let ν be a demand assigned to a primary demand κ, and assume that $\overline{N}(\kappa)\setminus\overline{N}(\nu)\neq\emptyset$. Then* $\mathrm{Exp}[d_{\phi(\kappa)\nu}\mid\neg\Lambda^\nu]\le C^{\mathrm{avg}}_\nu + 2\alpha^*_\nu$.

Proof. By (3), 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 $d_{\mu'\kappa}\le C^{\mathrm{avg}}_\kappa$. In this case, for every $\mu\in\overline{N}(\kappa)\setminus\overline{N}(\nu)$, we have

$$d_{\mu\nu}\le d_{\mu\kappa}+d_{\mu'\kappa}+d_{\mu'\nu}\le\alpha^*_\kappa+C^{\mathrm{avg}}_\kappa+\alpha^*_\nu\le C^{\mathrm{avg}}_\nu+2\alpha^*_\nu,$$

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 $D(\overline{N}(\kappa)\setminus\overline{N}(\nu),\nu)\le C^{\mathrm{avg}}_\nu+2\alpha^*_\nu$.

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

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

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. $\square$

We now continue our estimation of the connection cost. Denote by C_ν the random variable representing the connection cost for ν. The next step of our analysis is to show that

$$\mathrm{Exp}[C_\nu] \le C_\nu^{\mathrm{avg}} + \tfrac{2}{e}\alpha_\nu^*. \tag{4}$$

The argument is divided into three cases. The first, easy case is when ν is a primary demand κ. According to the algorithm (see Pseudocode 3, Line 2), we have $C_\kappa = d_{\mu\kappa}$ with probability $\bar{y}_\mu$, for $\mu \in \overline{N}(\kappa)$. Therefore $\mathrm{Exp}[C_\kappa] = C_\kappa^{\mathrm{avg}}$, so (4) holds.

Next, we consider a non-primary demand ν. Let κ be the primary demand that ν is assigned to. We first deal with the sub-case when $\overline{N}(\kappa) \setminus \overline{N}(\nu) = \emptyset$, which is the same as $\overline{N}(\kappa) \subseteq \overline{N}(\nu)$. Property (CO) implies that $\bar{x}_{\mu\nu} = \bar{y}_\mu = \bar{x}_{\mu\kappa}$ for every $\mu \in \overline{N}(\kappa)$, so we have $\sum_{\mu\in\overline{N}(\kappa)} \bar{x}_{\mu\nu} = \sum_{\mu\in\overline{N}(\kappa)} \bar{x}_{\mu\kappa} = 1$, due to (PS.1). On the other hand, we have $\sum_{\mu\in\overline{N}(\nu)} \bar{x}_{\mu\nu} = 1$, and $\bar{x}_{\mu\nu} > 0$ for all $\mu \in \overline{N}(\nu)$. Therefore $\overline{N}(\kappa) = \overline{N}(\nu)$ and C_ν has exactly the same distribution as C_κ. So this case reduces to the first case, namely we have $\mathrm{Exp}[C_\nu] = C_\nu^{\mathrm{avg}}$, and (4) holds.

The last, and only non-trivial case is 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 $d_1 \le d_2 \le \ldots \le d_l$.

Lemma 10. *Assume that $\overline{N}(\kappa) \setminus \overline{N}(\nu) \neq \emptyset$. Then, using the notation from the paragraph above, we have*

$$\mathrm{Exp}[C_\nu] \le \Big(1 - \textstyle\prod_{s=1}^{l}(1-y_s)\Big) \cdot \sum_{u=1}^{l} d_u y_u + \mathrm{Exp}[d_{\phi(\kappa)\nu} \mid \neg\Lambda^\nu] \cdot \prod_{s=1}^{l}(1-y_s)$$

Proof. The formal proof of Lemma 10 appears in [2]. The main idea of the proof is to consider a provably worse random process, which considers facilities in $\overline{N}(\nu)$ by grouping them according to the neighborhood of primary demands. That is, two facilities μ_1 and μ_2 belong to the same group if both are in $\overline{N}(\kappa)$ for some primary demand κ. The groups are then ordered by nondecreasing average distance to ν, where the average is $\sum_{\mu\in G} d_{\mu\nu}\bar{y}_\mu / \sum_{\mu\in G} \bar{y}_\mu$. The process first consider each group in this order and select a group G independently with probability $\sum_{\mu\in G} \bar{y}_\mu$. Stop as soon as one group is selected, or we end up with none selected. If one group G is selected, we open a facility in G by choosing each μ in G with probability $\bar{y}_\mu / \sum_{\mu\in G} \bar{y}_\mu$. We then connect demand ν to this facility μ. Otherwise no group is selected and we connect ν to $\phi(\kappa)$. Notice that the event occurred in each group are independent, which gives an easier estimate on the expected connection cost. More specifically, let the groups be $G_1, G_2, \ldots, G_k$ and $g_1 = \sum_{\mu\in G_1} \bar{y}_\mu$ and similarly we define $g_2, \ldots, g_k$. Also let $\bar{d}_1 = \sum_{\mu\in G_1} d_{\mu\nu}\bar{y}_\mu / \sum_{\mu\in G_1} \bar{y}_\mu$ and similarly we define $\bar{d}_2, \ldots, \bar{d}_k$. Then our expected connection cost, conditioned on the event that at least one facility in $\overline{N}(\nu)$ opens, is at most

$$\begin{aligned}
\mathrm{Exp}[C_\nu \mid \Lambda] &\le \bar{d}_1 g_1 + \bar{d}_2 g_2 (1-g_1) + \ldots + \bar{d}_k g_k (1-g_1)(1-g_2)\ldots(1-g_k) \\
&\le (\textstyle\sum_{s=1}^{k} \bar{d}_s g_s)(\sum_{t=1}^{k} g_t \prod_{z=1}^{t-1}(1-g_z)) \\
&= (\textstyle\sum_{i=1}^{l} d_i \bar{y}_i)(\sum_{t=1}^{k} g_t \prod_{z=1}^{t-1}(1-g_z)) \\
&\le (\textstyle\sum_{i=1}^{l} d_i \bar{y}_i)(1 - \prod_{i=1}^{l}(1-\bar{y}_i))
\end{aligned}$$

The first inequality follows from the use of a provably worse random process to estimate connection cost. The second inequality is from the inequality (B.1) in Appendix B. The third line, which is an equality, follows from the definition of $g_1, g_2, \ldots, g_k$ and $\bar{d}_1, \ldots, \bar{d}_k$. The last inequality follows from the fact that $(1 - g_1) \leq \prod_{\mu \in G_1}(1 - \bar{y}_\mu)$ since $g_1 = \sum_{\mu \in G_1} \bar{y}_\mu$. $\square$

Using Lemma 10, we estimate the (unconditional) expected connection cost of ν as follows:

$$\begin{aligned}
\mathrm{Exp}[C_\nu] &\leq \Big(1 - \prod_{s=1}^{l}(1 - y_s)\Big) \cdot \sum_{u=1}^{l} d_u y_u + \mathrm{Exp}[d_{\phi(\kappa)\nu} \mid \neg\Lambda^\nu] \cdot \prod_{s=1}^{l}(1 - y_s) && (5)\\
&\leq \Big(1 - \prod_{s=1}^{l}(1 - y_s)\Big) \cdot C_\nu^{\mathrm{avg}} + (C_\nu^{\mathrm{avg}} + 2\alpha_\nu^*) \cdot \prod_{s=1}^{l}(1 - y_s) && (6)\\
&\leq (1 - \tfrac{1}{e})C_\nu^{\mathrm{avg}} + \tfrac{1}{e}(C_\nu^{\mathrm{avg}} + 2\alpha_\nu^*) && (7)\\
&= C_\nu^{\mathrm{avg}} + \tfrac{2}{e}\alpha_\nu^*,
\end{aligned}$$

where inequality (5) is shown in the appendix (see also the proof in [2]), inequality (6) follows from Lemma 9, and inequality (7) follows from $\prod_{s=1}^{l}(1 - y_s) \leq 1/e$. This completes the proof of the bound (4) on the connection cost of ν.

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

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

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

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

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

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

Summarizing, we obtain the main result of this section.

Theorem 11. *Algorithm* ECHS *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, 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.* [4].

Recall that in Section 6 we showed how to compute an integral solution with facility cost bounded by F^* and connection cost bounded by $C^* + 2/e \cdot \mathrm{LP}^*$. Thus, while our facility cost does not exceed the optimal fractional facility cost, our connection cost is significantly larger than the connection cost in the optimal fractional solution. A natural idea is to balance these two ratios by reducing the connection cost at the expense of the facility cost. One way to do this would be to increase the probability of opening facilities, from $\bar{y}_\mu$ (used in

Algorithm ECHS) to, say, $\gamma\bar{y}_\mu$, for some $\gamma > 1$. This increases the expected facility cost by a factor of γ but, as it turns out, it also reduces the probability that an indirect connection occurs for a non-primary demand to $1/e^\gamma$ (from the previous value $1/e$). Moreover, for each primary demand κ, the new algorithm will select a facility from the nearest facilities μ in $\overline{N}(\kappa)$ such that the connection values $\bar{x}_{\mu\nu}$ sum up to $1/\gamma$, instead of 1. It is easily seen that this will improve the estimate on connection cost for primary demands. These two changes, along with a more refined analysis, are the essence of the approach in [4], expressed in our terminology.

Our approach can be thought of as a combination of the above ideas 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 disjoint parts, and with some restrictions on how parts from different demands can overlap.

We begin by describing properties that our partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ needs to satisfy. Assume that γ is some constant such that $1 < \gamma < 2$. As mentioned earlier, the neighborhood $\overline{N}(\nu)$ of each demand ν will be divided into two disjoint parts. The first part, called the *close neighborhood* and denoted $\overline{N}_{\text{cls}}(\nu)$, contains the facilities in $\overline{N}(\nu)$ nearest to ν with the total connection value equal $1/\gamma$, that is $\sum_{\mu\in\overline{N}_{\text{cls}}(\nu)} \bar{x}_{\mu\nu} = 1/\gamma$. The second part, called the *far neighborhood* and denoted $\overline{N}_{\text{far}}(\nu)$, contains the remaining facilities in $\overline{N}(\nu)$ (so $\sum_{\mu\in\overline{N}_{\text{far}}(\nu)} \bar{x}_{\mu\nu} = 1 - 1/\gamma$). We restate these definitions formally below in Property (NB). Recall that for any set A of facilities and a demand ν, by $D(A,\nu)$ we denote the average distance between ν and the facilities in A, that is $D(A,\nu) = \sum_{\mu\in A} d_{\mu\nu}\bar{y}_\mu / \sum_{\mu\in A}\bar{y}_\mu$. We will use notations $C^{\text{avg}}_{\text{cls}}(\nu) = D(\overline{N}_{\text{cls}}(\nu),\nu)$ and $C^{\text{avg}}_{\text{far}}(\nu) = D(\overline{N}_{\text{far}}(\nu),\nu)$ for the average distances from ν to its close and far neighborhoods, respectively. By the definition of these sets and the completeness property (CO), these distances can be expressed as

$$C^{\text{avg}}_{\text{cls}}(\nu) = \gamma \sum_{\mu\in\overline{N}_{\text{cls}}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu} \quad \text{and} \quad C^{\text{avg}}_{\text{far}}(\nu) = \frac{\gamma}{\gamma-1} \sum_{\mu\in\overline{N}_{\text{far}}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu}.$$

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.

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 a new neighborhood property (NB) and modified properties (PD') and (SI'), listed below.

(NB) *Neighborhoods.* For each demand $\nu \in \overline{\mathbb{C}}$, 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

- $\overline{N}_{\text{cls}}(\nu) \cap \overline{N}_{\text{far}}(\nu) = \emptyset$,
- $\sum_{\mu\in\overline{N}_{\text{cls}}(\nu)} \bar{x}_{\mu\nu} = 1/\gamma$, and
- if $\mu \in \overline{N}_{\text{cls}}(\nu)$ and $\mu' \in \overline{N}_{\text{far}}(\nu)$ then $d_{\mu\nu} \le d_{\mu'\nu}$.

Note that the first two conditions, together with (PS.1), imply 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 $i \in \mathbb{F}$, $\sum_{\kappa\in P}\sum_{\mu\in i\cap\overline{N}_{\mathrm{cls}}(\kappa)} \bar{x}_{\mu\kappa} \le y_i^*$. In the summation, as before, we overload notation i to stand for the set of facilities created on site 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) $C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) + C^{\mathrm{max}}_{\mathrm{cls}}(\nu) \ge C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) + C^{\mathrm{max}}_{\mathrm{cls}}(\kappa)$.

(SI') *Siblings.* For any pair $\nu, \nu' \in \overline{\mathbb{C}}$ 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 $\overline{N}(\nu') \cap \overline{N}_{\mathrm{cls}}(\kappa) = \emptyset$. In particular, by Property (PD'.3(a)), this implies that different sibling demands are assigned to different primary demands, since $\overline{N}_{\mathrm{cls}}(\nu')$ is a subset of $\overline{N}(\nu')$.

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 $\bar{x}_{\mu\nu}$ so that the total connection value of each demand ν is 1. As the partitioning algorithm proceeds, for any demand ν, $\overline{N}(\nu)$ denotes the set of facilities with $\bar{x}_{\mu\nu} > 0$; hence the notation $\overline{N}(\nu)$ actually represents a dynamic set which gets fixed once the partitioning algorithm concludes both Phase 2. On the other hand, $\overline{N}_{\mathrm{cls}}(\nu)$ and $\overline{N}_{\mathrm{far}}(\nu)$ refer to the close and far neighborhoods at the time when $\overline{N}(\nu)$ is fixed.

Similar to the algorithm in Section 4, Phase 1 runs in iterations. Fix some iteration and 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 $\widetilde{N}(j) = \{\mu_1, ..., \mu_q\}$ with non-decreasing distance from j, that is $d_{\mu_1 j} \le d_{\mu_2 j} \le \ldots \le d_{\mu_q j}$. Without loss of generality, there is an index l for which $\sum_{s=1}^{l} \widetilde{x}_{\mu_s j} = 1/\gamma$, since we can always split one facility to achieve this. Then we define $\widetilde{N}_{\mathrm{cls}}(j) = \{\mu_1, ..., \mu_l\}$. (Unlike close neighborhoods of demands, $\widetilde{N}_{\mathrm{cls}}(j)$ can vary over time.) In case of ties, which can occur when some facilities in $\widetilde{N}(j)$ are at the same distance from j, we use a tie-breaking rule that is explained in the proof of Lemma 12 (the only place where the rule is needed). We also use notation

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

When the iteration starts, we first find a not-yet-exhausted client p that minimizes the value of $\mathrm{tcc}_{\mathrm{cls}}(p) + \mathrm{dmax}_{\mathrm{cls}}(p)$ and create a new demand ν for p. Now we have two cases:

Case 1: $\widetilde{N}_{\mathrm{cls}}(p) \cap \overline{N}(\kappa) \neq \emptyset$ for some existing primary demand $\kappa \in P$. In this case we assign ν to κ. As before, if there are multiple such κ, we pick any of them. We also fix $\bar{x}_{\mu\nu} \leftarrow \widetilde{x}_{\mu p}$ and $\widetilde{x}_{\mu p} \leftarrow 0$ for each $\mu \in \widetilde{N}(p) \cap \overline{N}(\kappa)$. Note that although we check for overlap between $\widetilde{N}_{\mathrm{cls}}(p)$ and $\overline{N}(\kappa)$, the facilities we actually move into $\overline{N}(\nu)$ include all facilities in the intersection of $\widetilde{N}(p)$, a bigger set, with $\overline{N}(\kappa)$.

At this time, the total connection value between ν and $\mu \in \overline{N}(\nu)$ is at most $1/\gamma$, since $\sum_{\mu \in \overline{N}(\kappa)} \bar{y}_\mu = 1/\gamma$ (this follows from the definition of neighborhoods for new primary demands in Case 2 below) and we have $\overline{N}(\nu) \subseteq \overline{N}(\kappa)$ at this point. Later in Phase 2 we will add additional facilities from $\widetilde{N}(p)$ to $\overline{N}(\nu)$ to make ν's total connection value equal to 1.

Case 2: $\widetilde{N}_{\mathrm{cls}}(p) \cap \overline{N}(\kappa) = \emptyset$ for all existing primary demands $\kappa \in P$. In this case we make ν a primary demand (that is, add it to P) and assign it to itself. We then move the facilities from $\widetilde{N}_{\mathrm{cls}}(p)$ to $\overline{N}(\nu)$, that is for $\mu \in \widetilde{N}_{\mathrm{cls}}(p)$ we set $\bar{x}_{\mu\nu} \leftarrow \widetilde{x}_{\mu p}$ and $\widetilde{x}_{\mu p} \leftarrow 0$.

It is easy to see that the total connection value of ν to $\overline{N}(\nu)$ is now exactly $1/\gamma$, that is $\sum_{\mu \in \overline{N}(\nu)} \bar{y}_\mu = 1/\gamma$. Moreover, facilities remaining in $\widetilde{N}(p)$ are all farther away from ν than those in $\overline{N}(\nu)$. As we add only facilities from $\widetilde{N}(p)$ to $\overline{N}(\nu)$ in Phase 2, the final $\overline{N}_{\mathrm{cls}}(\nu)$ contains the same set of facilities as the current set $\overline{N}(\nu)$. (More precisely, $\overline{N}_{\mathrm{cls}}(\nu)$ consists of the facilities that either are currently in $\overline{N}(\nu)$ or were obtained from splitting the facilities currently in $\overline{N}(\nu)$.)

Once all clients are exhausted, that is, each client j has r_j demands created, Phase 1 concludes. We then run Phase 2, the augmenting phase, following the same steps as in Section 4. For each client j and each demand $\nu \in j$ with total connection value to $\overline{N}(\nu)$ less than 1 (that is, $\sum_{\mu \in \overline{N}(\nu)} \bar{x}_{\mu\nu} < 1$), we use our AugmentToUnit() procedure to add additional facilities (possibly split, if necessary) from $\widetilde{N}(j)$ to $\overline{N}(\nu)$ to make the total connection value between ν and $\overline{N}(\nu)$ equal 1.

This completes the description of the partitioning algorithm. Summarizing, for each client $j \in \mathbb{C}$ we created r_j demands on the same point as j, and we created a number of facilities at each site $i \in \mathbb{F}$. Thus computed sets of demands and facilities are denoted $\overline{\mathbb{C}}$ and $\overline{\mathbb{F}}$, respectively. For each facility $\mu \in i$ we defined its fractional opening value $\bar{y}_\mu$, $0 \leq \bar{y}_\mu \leq 1$, and for each demand $\nu \in j$ we defined its fractional connection value $\bar{x}_{\mu\nu} \in \{0, \bar{y}_\mu\}$. The connections with $\bar{x}_{\mu\nu} > 0$ define the neighborhood $\overline{N}(\nu)$. The facilities in $\overline{N}(\nu)$ that are closest to ν and have total connection value from ν equal $1/\gamma$ form the close neighborhood $\overline{N}_{\mathrm{cls}}(\nu)$, while the remaining facilities in $\overline{N}(\nu)$ form the far neighborhood $\overline{N}_{\mathrm{far}}(\nu)$. It remains to show that this partitioning satisfies all the desired properties.

Correctness of partitioning. We now argue that our partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ satisfies all the stated properties. Properties (PS), (CO) and (NB) are directly enforced by the algorithm.

(PD'.1) holds because for each primary demand $\kappa \in p$, $\overline{N}_{\mathrm{cls}}(\kappa)$ is the same set as $\widetilde{N}_{\mathrm{cls}}(p)$ at the time when κ was created, and $\widetilde{N}_{\mathrm{cls}}(p)$ is removed from $\widetilde{N}(p)$ right after this step. Further, the partitioning algorithm makes κ a primary demand only if $\widetilde{N}_{\text{cls}}(p)$ is disjoint from the set $\overline{N}(\kappa')$ of all existing primary demands κ' at that iteration, but these neighborhoods are the same as the final close neighborhoods $\overline{N}_{\text{cls}}(\kappa')$.

The justification of (PD'.2) is similar to that for (PD.2) from Section 4. All close neighborhoods of primary demands are disjoint, due to (PD'.1), so each facility $\mu \in i$ can appear in at most one $\overline{N}_{\text{cls}}(\kappa)$, for some $\kappa \in P$. Condition (CO) implies that $\bar{y}_\mu = \bar{x}_{\mu\kappa}$ for $\mu \in \overline{N}_{\text{cls}}(\kappa)$. As a result, the summation on the left-hand side is not larger than $\sum_{\mu\in i} \bar{y}_\mu = y_i^*$.

Regarding (PD'.3(a)), at first glance this property seems to follow directly from the algorithm, as we only assign a demand ν to a primary demand κ when $\overline{N}(\nu)$ at that iteration overlaps with $\overline{N}(\kappa)$ (which is equal to the final value of $\overline{N}_{\text{cls}}(\kappa)$). However, it is a little more subtle, as the final $\overline{N}_{\text{cls}}(\nu)$ may contain facilities added to $\overline{N}(\nu)$ in Phase 2. Those facilities may turn out to be closer to ν than some facilities in $\overline{N}(\kappa) \cap \widetilde{N}(j)$ (not $\widetilde{N}_{\text{cls}}(j)$) that we added to $\overline{N}(\nu)$ in Phase 1. If the final $\overline{N}_{\text{cls}}(\nu)$ consists only of facilities added in Phase 2, we no longer have the desired overlap of $\overline{N}_{\text{cls}}(\kappa)$ and $\overline{N}_{\text{cls}}(\nu)$. Luckily this bad scenario never occurs. We postpone the proof of this property to Lemma 12. The proof of (PD'.3(b)) is similar to that of Lemma 5, and we defer it to Lemma 13.

(SI'.1) follows directly from the algorithm because for each demand $\nu \in j$, all facilities added to $\overline{N}(\nu)$ are immediately removed from $\widetilde{N}(j)$ and each facility is added to $\overline{N}(\nu)$ of exactly one demand $\nu \in j$. Splitting facilities obviously preserves (SI'.1).

The proof of (SI'.2) is similar to that of Lemma 3. If $\kappa = \nu$ then (SI'.2) follows from (SI'.1), so we can assume that $\kappa \neq \nu$. Suppose that $\nu' \in j$ is assigned to $\kappa' \in P$ and consider the situation after Phase 1. By the way we reassign facilities in Case 1, at this time we have $\overline{N}(\nu) \subseteq \overline{N}(\kappa) = \overline{N}_{\text{cls}}(\kappa)$ and $\overline{N}(\nu') \subseteq \overline{N}(\kappa') = \overline{N}_{\text{cls}}(\kappa')$, so $\overline{N}(\nu') \cap \overline{N}_{\text{cls}}(\kappa) = \emptyset$, by (PD'.1). Moreover, we have $\widetilde{N}(j) \cap \overline{N}_{\text{cls}}(\kappa) = \emptyset$ after this iteration, because any facilities that were also in $\overline{N}_{\text{cls}}(\kappa)$ were removed from $\widetilde{N}(j)$ when ν was created. In Phase 2, augmentation does not change $\overline{N}_{\text{cls}}(\kappa)$ and all facilities added to $\overline{N}(\nu')$ are from the set $\widetilde{N}(j)$ at the end of Phase 1, which is a subset of the set $\widetilde{N}(j)$ after this iteration, since $\widetilde{N}(j)$ can only shrink. So the condition (SI'.2) will remain true.

Lemma 12. *Property (PD'.3(a)) holds.*

Proof. Let j be the client for which $\nu \in j$. We consider an iteration when we create ν from j and assign it to κ, and within this proof, notation $\widetilde{N}_{\text{cls}}(j)$ and $\widetilde{N}(j)$ will refer to the value of the sets at this particular time. At this time, $\overline{N}(\nu)$ is initialized to $\widetilde{N}(j) \cap \overline{N}(\kappa)$. Recall that $\overline{N}(\kappa)$ is now equal to the final $\overline{N}_{\text{cls}}(\kappa)$ (taking into account facility splitting). We would like to show that the set $\widetilde{N}_{\text{cls}}(j) \cap \overline{N}_{\text{cls}}(\kappa)$ (which is not empty) will be included in $\overline{N}_{\text{cls}}(\nu)$ at the end. Technically speaking, this will not be true due to facility splitting, so we need to rephrase this claim and the proof in terms of the set of facilities obtained after the algorithm completes.

We define the sets A, B, E^- and E^+ as the subsets of $\overline{\mathbb{F}}$ (the final set of facilities) that were obtained from splitting facilities in the sets $\widetilde{N}(j)$, $\widetilde{N}_{\text{cls}}(j) \cap \overline{N}_{\text{cls}}(\kappa)$, $\widetilde{N}_{\text{cls}}(j) - \overline{N}_{\text{cls}}(\kappa)$ and $\widetilde{N}(j) - \widetilde{N}_{\text{cls}}(j)$, respectively. (See Figure 1.) We claim that at the end $B \subseteq \overline{N}_{\text{cls}}(\nu)$, with the caveat that the ties in the definition of $\overline{N}_{\text{cls}}(\nu)$ are broken in favor of the facilities in B.

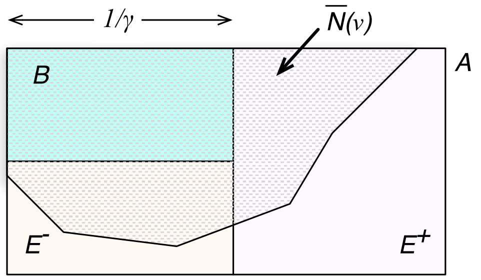

Figure 1: Illustration of the sets $\overline{N}(\nu)$, A, B, E^- and E^+ in the proof of Lemma 12. Let $X \Subset Y$ mean that the facility sets X is obtained from Y by splitting facilities. We then have $A \Subset \widetilde{N}(j)$, $B \Subset \widetilde{N}_{\rm cls}(j) \cap \overline{N}_{\rm cls}(\kappa)$, $E^- \Subset \widetilde{N}_{\rm cls}(j) - \overline{N}_{\rm cls}(\kappa)$, $E^+ \Subset \widetilde{N}(j) - \widetilde{N}_{\rm cls}(j)$.

(This is the tie-breaking rule that we mentioned in the definition of $\overline{N}_{\rm cls}(\nu)$.) This will be sufficient to prove the lemma because $B \neq \emptyset$, by the algorithm.

We now prove this claim. In this paragraph $\overline{N}(\nu)$ denotes the final set $\overline{N}(\nu)$ after both phases are completed. Thus the total connection value of $\overline{N}(\nu)$ to ν is 1. Note first that $B \subseteq \overline{N}(\nu) \subseteq A$, because we never remove facilities from $\overline{N}(\nu)$ and we only add facilities from $\widetilde{N}(j)$. Also, $B \cup E^-$ represents the facilities obtained from $\widetilde{N}_{\rm cls}(j)$, so $\sum_{\mu \in B \cup E^-} \bar{y}_\mu = 1/\gamma$. This and $B \subseteq \overline{N}(\nu)$ implies that the total connection value of $B \cup (\overline{N}(\nu) \cap E^-)$ to ν is at most $1/\gamma$. But all facilities in $B \cup (\overline{N}(\nu) \cap E^-)$ are closer to ν (taking into account our tie breaking) than those in $E^+ \cap \overline{N}(\nu)$. It follows that $B \subseteq \overline{N}_{\rm cls}(\nu)$, completing the proof. □

Lemma 13. *Property (PD'.3(b)) holds.*

Proof. This proof is similar to that for Lemma 5. For a client j and demand η, we will write $\mathrm{tcc}^{\eta}_{\rm cls}(j)$ and $\mathrm{dmax}^{\eta}_{\rm cls}(j)$ to denote the values of $\mathrm{tcc}_{\rm cls}(j)$ and $\mathrm{dmax}_{\rm cls}(j)$ at the time when η was created. (Here η may or may not be a demand of client j).

Suppose $\nu \in j$ is assigned to a primary demand $\kappa \in p$. By the way primary demands are constructed in the partitioning algorithm, $\widetilde{N}_{\rm cls}(p)$ becomes $\overline{N}(\kappa)$, which is equal to the final value of $\overline{N}_{\rm cls}(\kappa)$. So we have $C^{\rm avg}_{\rm cls}(\kappa) = \mathrm{tcc}^{\kappa}_{\rm cls}(p)$ and $C^{\max}_{\rm cls}(\kappa) = \mathrm{dmax}^{\kappa}_{\rm cls}(p)$. Further, since we choose p to minimize $\mathrm{tcc}_{\rm cls}(p) + \mathrm{dmax}_{\rm cls}(p)$, we have that $\mathrm{tcc}^{\kappa}_{\rm cls}(p) + \mathrm{dmax}^{\kappa}_{\rm cls}(p) \leq \mathrm{tcc}^{\kappa}_{\rm cls}(j) + \mathrm{dmax}^{\kappa}_{\rm cls}(j)$.

Using an argument analogous to that in the proof of Lemma 4, our modified partitioning algorithm guarantees that $\mathrm{tcc}^{\kappa}_{\rm cls}(j) \leq \mathrm{tcc}^{\nu}_{\rm cls}(j) \leq C^{\rm avg}_{\rm cls}(\nu)$ and $\mathrm{dmax}^{\kappa}_{\rm cls}(j) \leq \mathrm{dmax}^{\nu}_{\rm cls}(j) \leq C^{\max}_{\rm cls}(\nu)$ since ν was created later. Therefore, we have

$$\begin{aligned} C^{\rm avg}_{\rm cls}(\kappa) + C^{\max}_{\rm cls}(\kappa) &= \mathrm{tcc}^{\kappa}_{\rm cls}(p) + \mathrm{dmax}^{\kappa}_{\rm cls}(p) \\ &\leq \mathrm{tcc}^{\kappa}_{\rm cls}(j) + \mathrm{dmax}^{\kappa}_{\rm cls}(j) \leq \mathrm{tcc}^{\nu}_{\rm cls}(j) + \mathrm{dmax}^{\nu}_{\rm cls}(j) \leq C^{\rm avg}_{\rm cls}(\nu) + C^{\max}_{\rm cls}(\nu), \end{aligned}$$

completing the proof. □

Now we have completed the proof that the computed partitioning satisfies all the required properties.

Algorithm EBGS**.** The complete algorithm starts with solving the LP(1) 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 start the process of opening facilities and making connections to obtain an integral solution. To this end, for each primary demand $\kappa \in P$, we open exactly one facility $\phi(\kappa)$ in $\overline{N}_{\mathrm{cls}}(\kappa)$, where each $\mu \in \overline{N}_{\mathrm{cls}}(\kappa)$ is chosen as $\phi(\kappa)$ with probability $\gamma\bar{y}_\mu$. For all facilities $\mu \in \overline{\mathbb{F}} - \bigcup_{\kappa \in P} \overline{N}_{\mathrm{cls}}(\kappa)$, we open them independently, each with probability $\gamma\bar{y}_\mu$.

We claim that all probabilities are well-defined, that is $\gamma\bar{y}_\mu \le 1$ for all μ. Indeed, if $\bar{y}_\mu > 0$ then $\bar{y}_\mu = \bar{x}_{\mu\nu}$ for some ν, by Property (CO). If $\mu \in \overline{N}_{\mathrm{cls}}(\nu)$ then the definition of close neighborhoods implies that $\bar{x}_{\mu\nu} \le 1/\gamma$. If $\mu \in \overline{N}_{\mathrm{far}}(\nu)$ then $\bar{x}_{\mu\nu} \le 1 - 1/\gamma \le 1/\gamma$, because $\gamma < 2$. Thus $\gamma\bar{y}_\mu \le 1$, as claimed.

Next, we connect demands to facilities. Each primary demand $\kappa \in P$ will connect to the only open facility $\phi(\kappa)$ in $\overline{N}_{\mathrm{cls}}(\kappa)$. For each non-primary demand $\nu \in \overline{\mathbb{C}} - P$, 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. By the algorithm, for each client j, all its r_j demands are connected to open facilities. If two different siblings $\nu, \nu' \in j$ are assigned, respectively, to primary demands κ, κ' then, by Properties (SI'.1), (SI'.2), and (PD'.1) we have

$$(\overline{N}(\nu) \cup \overline{N}_{\mathrm{cls}}(\kappa)) \cap (\overline{N}(\nu') \cup \overline{N}_{\mathrm{cls}}(\kappa')) = \emptyset.$$

This condition guarantees that ν and ν' are assigned to different facilities, regardless whether they are connected to a neighbor facility or to its target facility. Therefore the computed solution is feasible.

We now estimate the cost of the solution computed by Algorithm EBGS. The lemma below bounds the expected facility cost.

Lemma 14. *The expectation of facility cost F_{EBGS} of Algorithm* EBGS *is at most γF^*.*

Proof. By the algorithm, each facility $\mu \in \overline{\mathbb{F}}$ is opened with probability $\gamma\bar{y}_\mu$, independently of whether it belongs to the close neighborhood of a primary demand or not. Therefore, by linearity of expectation, we have that the expected facility cost is

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

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

To bound the connection cost, we start with a lemma that estimates the cost of indirect connections. More precisely, if a non-primary demand ν is assigned to a primary demand κ, we want to estimate the expectation of the distance between ν and $\phi(\kappa)$, conditioned on the event that no facility in $\overline{N}(\nu)$ opens. The proof of this lemma relies on Properties (PD'.3(a)) and (PD'.3(b)) of modified partitioning and follows the reasoning in the proof of a similar lemma in [4, 5]. For the sake of completeness, we include a proof in Appendix A.

Lemma 15. *Let ν be a non-primary demand assigned to $\kappa \in P$ and let Λ^ν be the event that some facility in $\overline{N}(\nu)$ opens in Algorithm* EBGS*'s integral solution. Then*

$$\mathrm{Exp}[d_{\phi(\kappa)\nu} \mid \neg\Lambda^\nu] \leq C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) + C^{\max}_{\mathrm{cls}}(\nu) + C^{\mathrm{avg}}_{\mathrm{far}}(\nu).$$

A sketch of the proof is in Appendix A. We are now ready to bound the overall connection cost of Algorithm EBGS.

Lemma 16. *The expectation of connection cost C_{EBGS} of Algorithm* EBGS *is*

$$\mathrm{Exp}[C_{\mathrm{EBGS}}] \leq C^* \cdot \max\left\{\frac{1/e + 1/e^\gamma}{1 - 1/\gamma}, 1 + \frac{2}{e^\gamma}\right\}.$$

Proof. The argument is similar to that in [4]. Fix some non-primary demand ν. Let $\overline{N}(\nu) = \{\mu_1, \ldots, \mu_h\}$, where the facilities are ordered by non-decreasing distance to ν. Thus, denoting $d_s = d_{\mu_s\nu}$, for all s, we have $d_1 \leq d_2 \leq ... \leq d_h$. Choose l such that $\overline{N}_{\mathrm{cls}}(\nu) = \{\mu_1, \ldots, \mu_l\}$ and $\overline{N}_{\mathrm{far}}(\nu) = \{\mu_{l+1}, \ldots, \mu_h\}$.

Let C_ν be the random variable equal to the distance from ν to the facility that it connects to in the integral solution. As before, Λ^ν is the event that some facility in $\overline{N}(\nu)$ opens, and we also use notation $\Lambda^\nu_{\mathrm{cls}}$ for the event that some facility in $\overline{N}_{\mathrm{cls}}(\nu)$ opens. Using an argument similar to that in the analysis of Algorithm ECHS in Section 6, we can show that $\mathrm{Exp}[C_\nu]$ is upper bounded by the expected connection cost of ν obtained by the alternative random process that opens each facility $\mu \in \overline{N}(\nu)$ independently with probability $\gamma\bar{y}_\mu$, and connects to the nearest facility in $\overline{N}_{\mathrm{cls}}(\nu)$ if one opens, otherwise connect to the nearest facility open in $\overline{N}_{\mathrm{far}}(\nu)$, failing that, connect to its target facility $\phi(\kappa)$, the facility opened in the neighborhood of its assigned primary demand κ. Following this reasoning, we estimate $\mathrm{Exp}[C_\nu]$ as follows. Below we let $p_c \stackrel{\mathrm{def}}{=} 1 - \prod_{s=1}^{l}(1 - \gamma\bar{y}_{\mu_s})$ and $p_d \stackrel{\mathrm{def}}{=} 1 - \prod_{s=1}^{h}(1 - \gamma\bar{y}_{\mu_s})$.

$$\begin{aligned}
\mathrm{Exp}[C_\nu] &= \mathrm{Exp}[C_\nu \mid \Lambda^\nu_{\mathrm{cls}}] \cdot \mathrm{Prob}[\Lambda^\nu_{\mathrm{cls}}] + \mathrm{Exp}[C_\nu \mid \Lambda^\nu \ \wedge \neg\Lambda^\nu_{\mathrm{cls}}] \cdot \mathrm{Prob}[\Lambda^\nu \ \wedge \neg\Lambda^\nu_{\mathrm{cls}}] \\
&\quad + \mathrm{Exp}[C_\nu \mid \neg\Lambda^\nu] \cdot \mathrm{Prob}[\neg\Lambda^\nu] \\
&\leq C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) \cdot p_c + C^{\mathrm{avg}}_{\mathrm{far}}(\nu) \cdot (p_d - p_c) + \mathrm{Exp}[d_{\phi(\kappa)\nu} \mid \neg\Lambda^\nu] \cdot (1 - p_d) \\
&\leq C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) \cdot p_c + C^{\mathrm{avg}}_{\mathrm{far}}(\nu) \cdot (p_d - p_c) \\
&\quad + [C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) + C^{\max}_{\mathrm{cls}}(\nu) + C^{\mathrm{avg}}_{\mathrm{far}}(\nu)] \cdot (1 - p_d) \\
&= (C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) + C^{\mathrm{avg}}_{\mathrm{far}}(\nu)) \cdot (1 - p_d) - (C^{\mathrm{avg}}_{\mathrm{far}} - C^{\mathrm{avg}}_{\mathrm{cls}}) \cdot p_c + C^{\mathrm{avg}}_{\mathrm{far}}(\nu).
\end{aligned}$$

The first inequality follows from a similar argument in the proof of Lemma 10, and the second inequality is due to Lemma 15.

By definition $p_c = 1 - \prod_{s=1}^{l}(1 - \gamma\bar{y}_{\mu_s}) \geq 1 - 1/e$ since $\sum_{s=1}^{l}\bar{y}_{\mu_s} = 1/\gamma$. Similarly

$p_d = 1 - \prod_{s=1}^{h}(1-\gamma\bar{y}_{\mu_s}) \geq 1 - 1/e^\gamma$ since $\sum_{s=1}^{h}\bar{y}_{\mu_s} = 1$. Therefore,

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

where $\rho_\nu \overset{\mathrm{def}}{=} C^{\mathrm{avg}}_{\mathrm{cls}}(\nu)/C^{\mathrm{avg}}(\nu)$. It is easy to see that ρ_ν is between 0 and 1.

Since $\sum_{\nu\in j} C^{\mathrm{avg}}(\nu) = \sum_{\nu\in j}\sum_{\mu\in\overline{\mathbb{F}}} d_{\mu\nu}\bar{x}_{\mu\nu} = \sum_{i\in\mathbb{F}} d_{ij}x^*_{ij} = C^*_j$, summing over all clients j we have total connection cost bounded by $C^* \max\{\frac{1/e+1/e^\gamma}{1-1/\gamma}, 1+\frac{2}{e^\gamma}\}$, completing the proof. $\square$

Recall that the expected facility cost is bounded by γF^*, as argued earlier. Hence the total cost is bounded by $\max\{\gamma, \frac{1/e+1/e^\gamma}{1-1/\gamma}, 1 + \frac{2}{e^\gamma}\} \cdot \mathrm{LP}^*$. Picking $\gamma = 1.575$ we obtain the desired ratio.

Theorem 17. *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. As we mentioned earlier, we believe that our techniques of demand reduction and adaptive partitioning are very flexible and should be useful in extending other LP-rounding methods for UFL to obtain matching bounds for FTFP.

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] D. Shmoys, Éva Tardos, K. Aardal, Approximation algorithms for facility location problems (extended abstract), in: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, STOC '97, 1997, pp. 265–274.

[2] F. Chudak, D. Shmoys, Improved approximation algorithms for the uncapacitated facility location problem, SIAM J. Comput. 33 (1) (2004) 1–25.

[3] M. 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, 2002, pp. 240–257.

[4] J. Byrka, M. Ghodsi, A. Srinivasan, LP-rounding algorithms for facility-location problems, CoRR abs/1007.3611.

[5] J. Byrka, K. Aardal, An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem, SIAM J. Comput. 39 (6) (2010) 2212–2231.

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

[7] S. 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, Vol. 6756, 2011, pp. 77–88.

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

[9] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, V. Pandit, Local search heuristics for k-median and facility location problems, SIAM J. Comput. 33 (3) (2004) 544–562.

[10] S. Guha, S. Khuller, Greedy strikes back: improved facility location algorithms, in: Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, SODA '98, 1998, pp. 649–657.

[11] J. Vygen, Approximation Algorithms for Facility Location Problems, Forschungsinst. für Diskrete Mathematik, 2005.

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

[13] S. Guha, A. Meyerson, K. Munagala, Improved algorithms for fault tolerant facility location, in: Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms, SODA '01, 2001, pp. 636–641.

[14] C. Swamy, D. Shmoys, Fault-tolerant facility location, ACM Trans. Algorithms 4 (4) (2008) 1–27.

[15] J. Byrka, A. Srinivasan, C. Swamy, Fault-tolerant facility location, a randomized dependent LP-rounding algorithm, in: Proceedings of the 14th Integer Programming and Combinatorial Optimization, IPCO '10, 2010, pp. 244–257.

[16] S. Xu, H. Shen, The fault-tolerant facility allocation problem, in: Proceedings of the 20th International Symposium on Algorithms and Computation, ISAAC '09, 2009, pp. 689–698.

[17] L. Yan, M. Chrobak, Approximation algorithms for the fault-tolerant facility placement problem, Inf. Process. Lett. 111 (11) (2011) 545–549.

[18] K. Liao, H. Shen, Unconstrained and constrained fault-tolerant resource allocation, in: Proceedings of the 17th Annual International Conference on Computing and Combinatorics, COCOON'11, 2011, pp. 555–566.

[19] A. Gupta, Lecture notes: CMU 15-854b, spring 2008 (2008).

[20] M. Mahdian, Y. Ye, J. Zhang, Approximation algorithms for metric facility location problems, SIAM J. Comput. 36 (2) (2006) 411–432.

[21] G. H. Hardy, J. E. Littlewood, G. Pólya, Inequalities, Cambridge University Press, 1988.

Appendix A. Proof of Lemma 15

Lemma 15 provides a bound on the expected connection cost of a demand ν when none of facilities in $\overline{N}(\nu)$ opens by the rounding algorithm. The formal proof is very similar to that in [5], and we sketch the main steps here for completeness.

Let κ be the primary demand tht ν was assigned to. Also let $K_1 = \overline{N}_{\mathrm{cls}}(\kappa) \setminus \overline{N}(\nu)$, $K_2 = \overline{N}_{\mathrm{cls}}(\kappa) \cap \overline{N}_{\mathrm{cls}}(\nu)$ and $K_3 = \overline{N}_{\mathrm{cls}}(\kappa) \cap \overline{N}_{\mathrm{far}}(\nu)$. It is easy to see that K_1, K_2, K_3 are disjoint. Moreover, we have K_1 is not empty because the rounding algorithm opens some facility $\mu \in K_1$. K_2 is not empty because of (PD'.3(a)). We shall show that $D(K_1, \nu) \le C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) + C^{\max}_{\mathrm{cls}}(\kappa) + C^{\mathrm{avg}}_{\mathrm{far}}(\nu)$, which implies $D(K_1, \nu) \le C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) + C^{\max}_{\mathrm{cls}}(\nu) + C^{\mathrm{avg}}_{\mathrm{far}}(\nu)$ due to (PD'.3(b)). Because of the choice of the facility to open in $\overline{N}(\kappa)$ by the rounding algorithm, $D(K_1, \nu)$ is exactly the expected connection cost for demand ν conditioned on the event that none of ν's neighbors (both close and far) open. Recall that $D(A, \nu) \overset{\mathrm{def}}{=} \sum_{\mu \in A} d_{\mu\nu} \bar{y}_\mu$ is the average distance of facilities in set A to demand ν.

We have three cases:

- Case 1: $D(K_1, \kappa) \le C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa)$, then we can use any facility $\mu \in K_2$ to show that $D(K_1, \nu) \le D(K_1, \kappa) + d_{\mu\kappa} + d_{\mu\nu} \le C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) + C^{\max}_{\mathrm{cls}}(\kappa) + C^{\mathrm{avg}}_{\mathrm{far}}(\nu)$ because $d_{\mu\kappa} \le C^{\max}_{\mathrm{cls}}(\kappa)$ and $d_{\mu\nu} \le C^{\max}_{\mathrm{cls}}(\nu) \le C^{\mathrm{avg}}_{\mathrm{far}}(\nu)$, and we are done.
- Case 2: There exists a facility $\mu \in K_2$ such that $d_{\mu\kappa} \le C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa)$. Since $\mu \in K_2$, we infer that $d_{\mu\nu} \le C^{\max}_{\mathrm{cls}}(\nu) \le C^{\mathrm{avg}}_{\mathrm{far}}(\nu)$. Using $C^{\max}_{\mathrm{cls}}(\kappa)$ to bound $D(K_1, \kappa)$, we have $D(K_1, \nu) \le D(K_1, \kappa) + d_{\mu\kappa} + d_{\mu\nu} \le C^{\max}_{\mathrm{cls}}(\kappa) + C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) + C^{\mathrm{avg}}_{\mathrm{far}}(\nu)$.
- Case 3: In this case we can assume that $D(K_1, \kappa) > C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa)$, and every $\mu \in K_2$ has $d_{\mu\kappa} > C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa)$. As a result we can now assume that K_3 is not empty, because we have $\overline{N}_{\mathrm{cls}}(\kappa) = K_1 \cup K_2 \cup K_3$ and $D(K_1 \cup K_2, \kappa) > C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) \overset{\mathrm{def}}{=} D(\overline{N}_{\mathrm{cls}}(\kappa), \kappa)$. Moreover, there exists some finite number $\delta > 0$ such that $D(K_3, \kappa) = C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) - \delta$. We now have two subcases:
  - $D(K_3, \nu) \le C^{\mathrm{avg}}_{\mathrm{far}}(\nu) + \delta$.
    In this subcase there exists some $\mu \in K_3$ such that $d_{\mu\kappa} + d_{\mu\nu} \le C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) + C^{\mathrm{avg}}_{\mathrm{far}}(\nu)$ and we claim that $D(K_1, \nu) \le D(K_1, \kappa) + d_{\mu\kappa} + d_{\mu\nu} \le C^{\max}_{\mathrm{cls}}(\kappa) + C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) + C^{\mathrm{avg}}_{\mathrm{far}}(\nu)$.
  - $D(K_3, \nu) > C^{\mathrm{avg}}_{\mathrm{far}}(\nu) + \delta$.
    Because K_3 is a subset of $\overline{N}_{\mathrm{far}}(\nu)$, this gives us a bound on $D(\overline{N}_{\mathrm{far}}(\nu) \setminus K_3, \nu)$ (this set is not empty), which in turn, is an upper bound on $C^{\max}_{\mathrm{cls}}(\nu)$. Let $\hat{y} = \gamma \sum_{\mu \in K_3} \bar{y}_\mu$, using the fact that $C^{\mathrm{avg}}_{\mathrm{far}}(\nu) = D(K_3, \nu)\ \hat{y}/(\gamma - 1) + D(\overline{N}_{\mathrm{far}}(\nu) \setminus K_3)\ (\gamma - 1 - \hat{y})/(\gamma - 1)$ and in this subcase we have $D(K_3, \nu) > C^{\mathrm{avg}}_{\mathrm{far}}(\nu) + \delta$, we are able to show that
    $$C^{\max}_{\mathrm{cls}}(\nu) \le D(\overline{N}_{\mathrm{far}}(\nu) \setminus K_3, \nu) \le C^{\mathrm{avg}}_{\mathrm{far}}(\nu) - \hat{y}\delta/(\gamma - 1 - \hat{y}) \le C^{\mathrm{avg}}_{\mathrm{far}}(\nu) - \hat{y}\delta/(1 - \hat{y}) \qquad (*).$$
    The last step follows from $\gamma \le 2$ (we pick $\gamma = 1.575$ in the end). On the other hand, since $\overline{N}_{\mathrm{cls}}(\kappa) = K_1 \cup K_2 \cup K_3$, and K_1, K_2, K_3 are disjoint, we have $C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) =$

$(1-\hat{y})\ D(K_1 \cup K_2, \kappa) + \hat{y}\ D(K_3, \kappa)$, and by definition of δ we have $D(K_3, \kappa) = C^{\text{avg}}_{\text{cls}}(\kappa) - \delta$. Together they imply that

$$D(K_1 \cup K_2, \kappa) = C^{\text{avg}}_{\text{cls}}(\kappa) + \hat{y}\delta/(1-\hat{y}).$$

Now we are essentially done. If there exists some $\mu \in K_2$ such that $d_{\mu\kappa} \leq C^{\text{avg}}_{\text{cls}}(\kappa) + \hat{y}\delta/(1-\hat{y})$, then we have

$$\begin{aligned} D(K_1, \nu) &\leq D(K_1, \kappa) + d_{\mu\kappa} + d_{\mu\nu} \\ &\leq C^{\max}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{cls}}(\kappa) + \hat{y}\delta/(1-\hat{y}) + C^{\max}_{\text{cls}}(\nu) \\ &\leq C^{\max}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{far}}(\nu) \end{aligned}$$

Otherwise we must have $D(K_1, \kappa) \leq C^{\text{avg}}_{\text{cls}}(\kappa) + \hat{y}\delta/(1-\hat{y})$. It follows that

$$\begin{aligned} D(K_1, \nu) &\leq D(K_1, \kappa) + d_{\mu\kappa} + d_{\mu\nu} \\ &\leq C^{\text{avg}}_{\text{cls}}(\kappa) + \hat{y}\delta/(1-\hat{y}) + C^{\max}_{\text{cls}}(\kappa) + C^{\max}_{\text{cls}}(\nu) \\ &\leq C^{\max}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{far}}(\nu). \end{aligned}$$

In both derivations above we use the (*) inequality to bound $C^{\max}_{\text{cls}}(\nu)$ in the last step.

This concludes the proof of Lemma 15, and we have the desired bound on the expected distance when the algorithm connnects a demand ν indirectly to its target facility, that is, the facility opened by the primary demand that ν was assigned.

Appendix B. Proof of an Inequality for Analysis of Algorithm ECHS

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

$$\bar{d}_1 g_1 + \bar{d}_2 g_2 (1-g_1) + \ldots + \bar{d}_k g_k (1-g_1)(1-g_2)\ldots(1-g_k) \leq (\textstyle\sum_{s=1}^{k} \bar{d}_s g_s)(\sum_{t=1}^{k} g_t \prod_{z=1}^{t-1}(1-g_z)) \tag{B.1}$$

for $\bar{d}_1 \leq \bar{d}_2 \leq \ldots \leq \bar{d}_k$ and $\sum_{s=1}^{k} g_s = 1, y_s \geq 0$.

In this section we give a new proof of this inequality, much simpler than the existing proof in [2], and also simpler than the argument by Sviridenko [3]. We derive this inequality from the following generalized version of the Chebyshev Sum Inequality:

$$\textstyle\sum_i p_i \sum_j p_j a_j b_j \leq \sum_i p_i a_i \sum_j p_j b_j, \tag{B.2}$$

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 (B.2), we can obtain our inequality (B.1) by simple substitution

$$p_i \leftarrow g_i, a_i \leftarrow \bar{d}_i, b_i \leftarrow \Pi_{s=1}^{i-1}(1-g_s).$$

For the sake of completeness, we include the proof of inequality (B.2), due to Hardy, Littlewood and Polya [21]. The idea is to evaluate the following sum:

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

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.