Algorithmica (2016) 75:53–83
DOI 10.1007/s00453-015-0012-z

New Approximation Algorithms for the Unsplittable Capacitated Facility Location Problem

Babak Behsaz[1] · **Mohammad R. Salavatipour**[1] · **Zoya Svitkina**[2]

Received: 18 July 2014 / Accepted: 26 May 2015 / Published online: 5 June 2015

Abstract In this paper, we consider the Unsplittable (hard) Capacitated Facility Location Problem (UCFLP) with uniform capacities and present new approximation algorithms for it. This problem is a generalization of the classical facility location problem where each facility can serve at most u units of demand and each client must be served by *exactly one* facility. This problem is motivated by its applications in many practical problems including supply chain problems of indivisible goods (Verter in Foundations of location analysis, chapter 2. International series in operations research and management science. Springer, Berlin, 2011) and the assignment problem in the content distribution networks (Bateni and Hajiaghayi in Proceedings of the nineteenth annual ACM-SIAM symposium on discrete algorithms, pp 805–814, 2009). While there are several approximation algorithms for the *soft* capacitated version of this problem (in which one can open multiple copies of each facility) or the splittable version (in which the demand of each client can be divided to be served

A preliminary version of this paper has appeared in the Proceedings of 13th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT), Pages 237–248, 2012.
Babak Behsaz: Supported in part by Alberta Innovates Graduate Student Scholarship.
Mohammad R. Salavatipour: Supported by NSERC and an Alberta Ingenuity New Faculty Award.
Zoya Svitkina: Supported in part by Alberta Ingenuity.

✉ Mohammad R. Salavatipour
mreza@cs.ualberta.ca; mrs@ualberta.ca

Babak Behsaz
behsaz@ualberta.ca

Zoya Svitkina
zoya@cs.cornell.edu

[1] Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada

[2] Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA

by multiple open facilities), there are very few results for the UCFLP. It is known that it is NP-hard to approximate this problem within any factor without violating the capacities. So we consider bicriteria (α, β)-approximations where the algorithm returns a solution whose cost is within factor α of the optimum and violates the capacity constraints within factor β. Shmoys et al. (Proceedings of the twenty-ninth annual ACM symposium on theory of computing, pp 265–274, 1997) were the first to consider this problem and gave a $(9, 4)$-approximation. Later results imply $(O(1), 2)$-approximations, however, no constant factor approximation is known with capacity violation of less than 2. We present a framework for designing bicriteria approximation algorithms for this problem and show two new approximation algorithms with factors $(9, 3/2)$ and $(29.315, 4/3)$. These are the first algorithms with constant approximation in which the violation of capacities is below 2. The heart of our algorithm is a reduction from the UCFLP to a restricted version of the problem. One feature of this reduction is that any $(O(1), 1+\epsilon)$-approximation for the restricted version implies an $(O(1), 1+\epsilon)$-approximation for the UCFLP and we believe our techniques might be useful towards finding such approximations or perhaps $(f(\epsilon), 1+\epsilon)$-approximation for the UCFLP for some function f. In addition, we present a quasi-polynomial time $(1+\epsilon, 1+\epsilon)$-approximation for the (uniform) UCFLP in Euclidean metrics, for any constant $\epsilon > 0$.

Keywords Approximation algorithms · Capacitated facility location · Hard capacities

1 Introduction

We consider the Unsplittable Capacitated Facility Location Problem (UCFLP) with uniform capacities. In this problem, we are given a set of clients C and a set of facilities F where each client j has demand d_j and each facility i has opening cost f_i and capacity u. There is a metric cost c_{ij} which denotes the cost of serving one unit of demand of client j at facility i. The goal is to open a subset of facilities $I \subseteq F$ and assign each client j to *exactly one* open facility $\phi(j)$ to serve its entire demand d_j so that the total amount of demand assigned to each open facility is no more than u, while minimizing the total cost of opening facilities and connecting (serving) clients, i.e., minimizing $\sum_{i \in I} f_i + \sum_{j \in C} d_j c_{\phi(j)j}$. This problem generalizes the bin packing, the minimum makespan, and some facility location problems. If the demands of clients can be served by multiple open facilities, then we have the *splittable* capacitated version of the problem. If each facility can be opened multiple times then we have the so-called *soft* capacitated version. Each of these relaxations (i.e., allowing splitting the demands of clients and/or having multiple copies of each facility) makes the problem significantly easier as discussed below. When each facility i has a given capacity u_i, the problem is called the *non-uniform* version.

Facility location problems have been studied extensively in operations research and management sciences and even a few books are devoted to these problems (e.g., see [17,29]). They are also well studied in theoretical computer science and various approximation algorithms are designed. While these problems arise in a wide range of

practical applications, the most common context of their employment has been supply chain. In a supply chain that consists of suppliers, distribution centres, or warehouses and customers, these problems emerge in making location decisions [36]. When we deal with indivisible goods, the unsplittable demand assumption is a necessity. In particular, the UCFLP has been studied in operations research literature from the eighties, where it is called the capacitated facility location problem with single sourcing or the capacitated concentrator location problem [17]. The latter name comes from the problem of assigning a set of terminals or workstations to some concentrator devices in telecommunications networks. Here, each terminal has a demand that must be served by exactly one concentrator and each concentrator has a capacity that shows the amount of traffic that it can manage.

As Bateni and Hajiaghayi [9] pointed out, solving the UCFLP without relaxation of capacities is NP-hard even in very special cases. This is done via a simple reduction from the special case of minimum makespan when the size of each client j is in a set $\{p_j, \infty\}$. In fact, it can be shown that unless **P**= **NP**, any approximation algorithm with a bounded approximation ratio for the UCFLP violates the capacities of at least $\lfloor \frac{|F|}{2} \rfloor$ facilities in some instances [10]. Thus, research has focused on the design of bicriteria approximation algorithms. An (α, β)-*approximation* algorithm for the UCFLP returns a solution whose cost is within factor α of the optimum and violates the capacity constraints within factor β. It should be noted that if we violate capacity of a facility within factor β, we must pay β times its opening cost. In the context of approximation algorithms, Shmoys et al. [35] were the first to consider this problem and presented a $(9, 4)$-approximation algorithm. They used a filtering and rounding technique to get an approximation algorithm for the splittable version and used a rounding for the generalized assignment problem (GAP) [34] to obtain their algorithm for the unsplittable version. This technique of reducing the unsplittable version using the rounding for the GAP to the splittable version was a cornerstone of the subsequent approximation algorithms. In addition, in the same place, by a randomized version of the filtering technique, they got a new algorithm with the ratio (7.62, 4.29). Korupolu et al. [25] gave the first constant factor approximation algorithm for the splittable hard capacitated version, and applied the GAP rounding technique of [35] to get a $(O(1), 2)$-approximation algorithm for the UCFLP. Applying the current best approximation algorithms for the splittable capacitated version with non-uniform capacities [37] and uniform capacities [1], it is straightforward to get factor $(9, 2)$ and $(5, 2)$ approximation algorithms for the UCFLP with non-uniform and uniform capacities, respectively [10].

Bateni and Hajiaghayi [9] designed the first approximation algorithms for the UCFLP having violation ratio less than 2. They modelled an assignment problem in content distribution networks by the UCFLP. This assignment problem has been first considered by Alzoubi et al. [2] and is basically the assignment of downloadable objects, such as media files or softwares, to some servers. We cannot split a downloadable object and we need to store it in a single server. As Alzoubi et al. mention, the server capacities is very crucial in practice and a high overload amount on a server can disrupt a large numbers of connections. Motivated by this strict requirement on capacities, the authors of [9] designed a $(1+\epsilon, 1+\epsilon)$-approximation algorithm for *tree metrics* (for any constant $\epsilon > 0$) using a dynamic programming approach. They also presented a

quasi-polynomial time $(1+\epsilon, 1+\epsilon)$-approximation algorithm (again for trees) for the non-uniform capacity case. Using Fakcharoenphol et al.'s [18] improvement of Bartal's machinery [8], this implies a polynomial time $(O(\log n), 1+\epsilon)$-approximation algorithm for almost uniform capacities and a quasi-polynomial time $(O(\log n), 1+\epsilon)$-approximation algorithm for non-uniform version for an arbitrary constant $\epsilon > 0$.

1.1 Related Work

Perhaps the most well-studied facility location problem is the *uncapacitated* facility location problem (UFLP). In this problem, we do not have the capacity constraints and we only need to decide which facilities to open, as each client will be assigned to its closest open facility. The first constant approximation for the UFLP was a 3.16-approximation algorithm by Shmoys et al. [35]. This algorithm was based on a filtering method due to Lin and Vitter [28] and rounding a linear programming (LP) formulation of the problem.

There is a long series of works that improve this constant approximation ratio for the UFLP. Guha and Khuller [20] improved the ratio to 2.41 by combining a simple greedy heuristic with the algorithm of [35]. This greedy heuristic adds unopened facilities one by one greedily based on some measure of effectiveness for each facility. Later, the factor improved to $1 + 2/e \approx 1.74$ by Chudak [13] using generalized techniques of [35] for the LP rounding. A key element to this improvement is the use of randomized rounding of some variables in conjunction with the approach of Shmoys et al. Meanwhile, Jain and Vazirani [22] gave a 3-approximation primal-dual algorithm with a better running time and Korupolu et al. [25] gave a surprisingly simple local search algorithm with factor $5 + \epsilon$ for any $\epsilon > 0$.

Charikar and Guha [12] slightly improved the ratio to 1.73 by combining primal dual algorithm of [22] with cost scaling and greedy augmentation. The scaling technique exploits the difference between approximation guarantees for the facility cost and the service cost. This can be used by producing a new instance where the facility costs are multiplied by δ, then apply the algorithm to the scaled instance, and then scale back to get a solution for the original instance. Mahdian et al. [21] used a variant of the primal-dual method, called dual fitting, and a new analysis technique, called factor revealing LP, to bring down the factor to 1.61. Later, Mahdian et al. [32] combined this algorithm with greedy augmentation of [12] to decrease the factor to 1.52. Afterwards, Byrka [11] combined this new algorithm of [32] with the algorithm of Chudak [13] to get a 1.5-approximation. Finally, Li [27] showed that by choosing a parameter of Byrka's algorithm from a specific distribution, one can get a factor 1.488 approximation algorithm, which is the current best known factor.

On the negative side, a result of Guha and Khuller [20], combined with an observation of Sviridenko (personal communication cited in [15]), implies 1.463-hardness for the Uncapacitated Facility Location Problem (UFLP). As a result, unless $\mathbf{P} = \mathbf{NP}$, there is very little room to improve the best known approximation algorithm for the UFLP.

The (soft and hard) capacitated facility location problems have also received a lot of attention. In the soft capacitated facility location problem, despite having capacities,

the fact that we can open a facility multiple times makes the problem easier in comparison to the case of hard capacities. Shmoys et al. [35] designed the first constant factor, namely a 5.69-approximation algorithm for this problem by a similar LP rounding technique they used for the first constant factor approximation for the UFLP. Then, Chudak and Shmoys [14] used the same LP and the randomized LP-rounding technique to get a 3-approximation algorithm for this problem. For non-uniform capacities, Jain and Vazirani [22] reduced this problem to the UFLP, and by solving the UFLP, they obtained a 4-approximation algorithm. Arya et al. [6] proposed a simple local search algorithm with an approximation ratio of 3.72 for the non-uniform version. Following the reduction of Jain and Vazirani [22] to the UFLP, Jain et al. [21] showed that the soft Capacitated Facility Location (CFLP) with non-uniform capacities (where one can open $\lceil x/u_i \rceil$ copies of a facility with capacity u_i to serve x demands) can be solved within a factor 3 of optimum. This result was improved to a 2.89-approximation algorithm for the non-uniform soft CFLP in [32]. Finally, Mahdian et al. [31] improved this factor to 2, achieving the integrality gap of the natural LP relaxation of the problem. To the best of our knowledge, this is the current best ratio for this problem.

We should point out that all of the above algorithms except the local search algorithm of Arya et al. [6] use the optimal value of a natural LP relaxation of the soft capacitated facility location problem as a lower bound in their analysis. Therefore, they cannot obtain a better ratio than the integrality gap of this relaxation. Mahdian et al. [31] also showed that the integrability gap of this LP is 2 and hence, their analysis is tight.

If we add the constraint that the demand of a client cannot be split, it does not make the problem much more difficult than its splittable counterpart in the soft setting. One can show that any α-approximation algorithm for the unsplittable version of the soft CFLP yields a 2α-approximation algorithm for the splittable version and vice versa [10]. As a result, the 2-approximation algorithm of Mahdian et al. [31] yields a 4-approximation for unsplittable version of the soft CFLP. In fact, it is not difficult to observe that this algorithm is a 2-approximation for this version. Their algorithm assigns each client by utilizing a UFLP algorithm and hence, to a *single* facility, and its cost is within factor 2 of the optimum value of soft CFLP. Clearly, the optimum value of the unsplittable version is not less than the optimum value of the splittable version, because all feasible solutions for the unsplittable version are feasible solutions with the same cost for the splittable version, too. Thus, their algorithm gives a feasible solution for the unsplittable version of the soft CFLP which is within factor 2 of the optimum value for the unsplittable version.

The (splittable) hard capacitated facility location problem has also received a lot of attention. In contrast to the UFLP and soft capacitated facility location problem, there is an important distinction between the splittable and unsplittable case in the presence of hard capacities, because in the unsplittable case, even checking whether there exists a feasible solution becomes **NP**-hard and we can only hope for a bicriteria algorithm. In contrast, in the splittale case, if we decide on the set of open facilities, the best way of serving the clients can be determined by building a flow network and using a minimum cost flow algorithm.

For the splittable case, Korupolu et al. [25] gave a simple factor $8 + \epsilon$ local search algorithm. This was the first constant factor approximation algorithm for the hard capacitated facility location problem.

Later, Chudak and Williamson [15] simplified their analysis and showed the actual ratio of algorithm of Korupolu et al. [25] is at most $6 + \epsilon$. Pal et al. [33] gave a more powerful local search with factor $8.54+\epsilon$ for the case of non-uniform capacities. Later, with a series of more powerful local search algorithms, the ratio for the case of non-uniform capacities decreased to $7.46 + \epsilon$ [30], and $5.83 + \epsilon$ [37]. Later, Aggarwal et al. [1] showed that the algorithm of Korupolu et al. [25] is actually a 3-approximation for the uniform splittable CFLP and this ratio is tight. Recently, Bansal et al. [7] showed that slightly more powerful versions of algorithm of Zhang et al. [37] give a 5-approximation for the non-uniform splittable CFLP and this ratio is tight.

It should be noted that in contrast to the UFLP and soft CFLP, all the known LP relaxations for this problem have super-constant integrality gap in the general case. The only LP-based result is a 5-approximation algorithm by Levi et al. [26] for the non-uniform version in the special case that all facility opening costs are equal.

1.2 Our Results

Recall that given an instance (F, C) of the UCFLP with facility opening costs f_i, demands d_j, and connection costs c_{ij}, a solution is a subset I of facilities to open along with assignment function $\phi : C \to I$. Since all capacities are uniform, by a simple scaling, we can assume that all of them are 1 and all the client demands are at most 1.

All the known constant factor algorithms for the UCFLP violate the capacity constraints by a factor of at least 2 which is mainly due to using the rounding algorithm for GAP [34]; and the algorithm of [9] (although it has $1 + \epsilon$ violation) is not a constant factor approximation. We are interested in $(O(1), \beta)$-approximation algorithms for a $\beta < 2$. We define a restricted version of the problem and show that finding a good approximation algorithm for this restricted version would imply a good approximation for the general version. The definition of similar restricted versions has been a common practice in solving bin packing type problems (e.g., see [16,23]).

Definition 1 An ϵ-restricted UCFLP instance, denoted by RUCFLP(ϵ), is an instance of the UCFLP in which each demand has size more than ϵ, i.e., $\epsilon < d_j \leq 1$ for all $j \in C$.

The following theorem establishes the reduction from the general instances of the UCFLP to the restricted version. Here, the general idea is that if we assign the large clients oblivious to small clients, we can fractionally assign the small clients without paying too much in cost. We use the maximum-flow minimum-cut theorem to show this. Then we can round this fractional assignment of small clients with the GAP rounding technique [34].

Theorem 1 *If $\mathcal{A}$ is an $(\alpha(\epsilon), \beta(\epsilon))$-approximation algorithm for the RUCFLP(ϵ) with running time $\tau(\mathcal{A})$ then there is an algorithm $\mathcal{A}_C$ which is an $(\eta(1 + \epsilon)\alpha(\epsilon), \max\{\beta(\epsilon), 1 + \epsilon\})$-approximation algorithm for the UCFLP, for some constant η, whose running time is polynomial in $\tau(\mathcal{A})$ and the instance size.*

Corollary 1 *For any constant $\epsilon > 0$, an $(\alpha(\epsilon), 1+\epsilon)$-approximation algorithm for the RUCFLP(ϵ) yields an $(O(\alpha(\epsilon)), 1 + \epsilon)$-approximation for the UCFLP. Particularly,*

when $\alpha(\epsilon)$ is a constant, we have a constant approximation for the UCFLP with a $(1+\epsilon)$ factor violation of capacities in polynomial time.

This reduction shows that to get a $(O(1), 1+\epsilon)$-approximation, it is sufficient to consider large clients only, which may open the possibility of designing algorithms using some of the techniques used in the bin packing type problems. If one finds such an algorithm for large clients, the above corollary shows that we have an $(O(1), (1+\epsilon))$-approximation for the UCFLP. As an evidence for this, we find approximation algorithms for the RUCFLP($\frac{1}{2}$) and the RUCFLP($\frac{1}{3}$). For the RUCFLP($\frac{1}{2}$), we present an exact algorithm and for the RUCFLP($\frac{1}{3}$), we present a (21, 1)-approximation algorithm. These, together with Theorem 1, yield:

Theorem 2 *There is a polynomial time $(9, \frac{3}{2})$-approximation algorithm for the UCFLP.*

Theorem 3 *There is a polynomial time $(29.315, \frac{4}{3})$-approximation algorithm for the UCFLP.*

Finally, we give a quasi polynomial time approximation scheme (QPTAS) for the UCFLP restricted to Euclidean metrics. Here, we employ a dynamic programming based algorithm and combine the shifted quad-tree dissection of Arora [3], some ideas from [9], and some new ideas.

Theorem 4 *There exists a $(1+\epsilon', 1+\epsilon')$-approximation algorithm for the Euclidean UCFLP in $\mathbb{R}^2$ with quasi-polynomial running time for any constant $\epsilon' > 0$.*

Although this theorem is presented for $\mathbb{R}^2$, it can be generalized to $\mathbb{R}^d$ for any constant $d > 2$.

In the following discussions, for a solution (I, ϕ), where I is the set of open facilities and $\phi : C \to I$ is the assignment of clients to open facilities, we use $c_f(\phi)$ to denote the total facility opening cost and $c_s(\phi)$ to denote the total service cost, and $c(\phi)$ to denote the total cost of the solution. Thus, we have $c(\phi) = c_f(\phi) + c_s(\phi)$. In the splittable versions, the assignment function is $\phi : C \times F \to \mathbb{R}_{\geq 0}$, where $\phi(i, j)$ shows the amount of demand of client j served by facility i.

The rest of this paper is organized as follows. In Sect. 2, we prove Theorem 1. In the next section, we present approximation algorithms for the RUCFLP(1/2) and RUCFLP(1/3), which also prove Theorems 2 and 3. In Sect. 4, we give a QPTAS for Euclidean metrics. Finally, in Sect. 5, we conclude the paper with a discussion of results, future works and open problems.

2 Reduction to the Restricted UCFLP

Let $L = \{j \in C : d_j > \epsilon\}$ be the set of large clients and $S = C \backslash L$ be the set of small clients[1]. We call two assignments $\phi_1 : C_1 \to F_1$ and $\phi_2 : C_2 \to F_2$ consistent if

[1] We should point out that the definitions of L and S are with respect to a given parameter ϵ. Since throughout the following sections, this parameter is the same for all statements, in the interest of brevity, we use this notation instead of $L(\epsilon)$ and $S(\epsilon)$.

$\phi_1(j) = \phi_2(j)$ for all $j \in C_1 \cap C_2$. The high level idea of the algorithm is as follows. We first ignore the small clients and solve the problem restricted to only the large clients by running algorithm $\mathcal{A}$ of Theorem 1. We can show that given a *good* assignment of large clients, there exists a *good* assignment of all the clients (large and small) that is consistent with this assignment of large clients, i.e., a solution which assigns the large clients the same way that $\mathcal{A}$ does, whose cost is not far from the optimum cost. More specifically, we show there exists a *fractional* (i.e., splittable) assignment of small clients that together with the assignment of large clients obtained from $\mathcal{A}$ gives an approximately good solution. Then, we try to find a fractional assignment of small clients. To do this, we update the capacities and the opening costs of facilities with respect to the assignment of large clients (according to the solution of $\mathcal{A}$). Then, we fractionally assign small clients and round this fractional assignment at the cost of violating the capacities with additive factor ϵ by using a rounding algorithm for the Generalized Assignment Problem (GAP).

The GAP is a generalization of the matching problem that can be described as a scheduling problem which has similarities to the UCFLP. In the GAP, we have a collection of jobs J and a set M of machines. Each job must be assigned to exactly one machine in M. If job $j \in J$ is assigned to machine $i \in M$, then it requires p_{ij} units of processing time and incurs a cost r_{ij}. Each machine $i \in M$ can be assigned jobs of total processing time at most P_i. We want to find an assignment of jobs to machines to minimize the total assignment cost. We should point out that r_{ij} values do not necessarily satisfy the triangle inequality. Shmoys and Tardos [34] considered an LP relaxation and showed that a feasible solution of this LP can be rounded, in polynomial time, to an integral solution with the same cost that violates processing time limit P_i within additive factor $\max_{j \in J} p_{ij}$.

Our algorithm is presented below. Here, $\phi^{-1}(i)$ is the set of clients assigned to facility i by the assignment ϕ and for a set $F' \subseteq F$, $\phi^{-1}(F') = \cup_{i \in F'} \phi^{-1}(i)$.

Algorithm 1 Algorithm for the UCFLP by reduction to the RUCFLP(ϵ)

Input: An instance of the UCFLP, a parameter $\epsilon > 0$, and an algorithm $\mathcal{A}$ for the RUCFLP(ϵ)
Output: A subset $I \subseteq F$ to open and an assignment of clients $\phi : C \to I$

1: Let $L = \{j \in C : d_j > \epsilon\}$ and $S = C \backslash L$. Assign the clients in L by running $\mathcal{A}$. Let I_L be the opened facilities and $\phi_L : L \to I_L$ be the assignment found by $\mathcal{A}$.
2: For $i \in I_L$, set $f_i = 0$, and set $u'_i = \max\{0, 1 - \sum_{j \in \phi_L^{-1}(i)} d_j\}$ as the new capacity of facility i. Assign the clients in S with respect to updated opening costs and capacities by an approximation algorithm for the splittable CFLP with *non-uniform* capacities. Let I_S be the new set of opened facilities and $\phi'_S : S \to I'_S$ be the assignment function, where $I'_S \subseteq I_S \cup I_L$.
3: Round the splittable assignment ϕ'_S using algorithm of [34] for GAP to find an unsplittable assignment $\phi_S : S \to I'_S$.
4: Let $I = I'_S \cup I_L$ and define $\phi : C \to I$ as $\phi(j) = \phi_S(j)$ if $j \in S$ and $\phi(j) = \phi_L(j)$, otherwise. Return ϕ and I.

First, we formally prove the property that given assignment of large clients, there is a feasible *fractional* assignment of small clients with an acceptable cost. A feasible fractional assignment is an assignment of demands of clients to open facilities where each client's demand might be served by multiple facilities (instead of just one),

i.e. we split their demands between multiple open facilities while satisfying capacity constraints. Note that we do not open facilities fractionally (they are open integraly). We should point out that the proof of this property is only an existential result and we do not actually find the assignment in the proof. We only use this lemma to bound the cost of our solution. Let *OPT* be an optimum solution which opens set I^* of facilities and with assignment of clients $\phi^* : C \to I^*$. We use $\phi^*_L : L \to I^*$ and $\phi^*_S : S \to I^*$ to denote the restriction of ϕ^* to large and small clients, respectively.

Lemma 1 *Suppose I_L is a set of open facilities and $\phi_L : L \to I_L$ is an arbitrary (not necessarily capacity respecting) assignment of large clients. Given the assignment ϕ_L, there exists a feasible fractional assignment of small clients, $\phi''_S : S \to I''_S$ such that $c_s(\phi''_S) \leq c_s(\phi^*) + c_s(\phi_L)$ and $c_f(\phi''_S) \leq c_f(\phi^*)$.*

Proof Recall u'_i is equal to $\max\{0, 1 - \sum_{j \in \phi_L^{-1}(i)} d_j\}$, i.e., the amount of capacity left for facility i after the assignment of large clients based on ϕ_L. We open all the open facilities in the optimum solution, i.e., all facilities in I^* (if not already open in I_L). Let $I''_S = I_L \cup I^*$. To show the existence of ϕ''_S, first we move the demands of small clients to the facilities in I^* based on ϕ^*_S and we pay $c_s(\phi^*_S)$ for this. So now the demands of small clients are located at facilities in I^*. However, a facility $i \in I''_S$ has only u'_i residual capacity left (after committing parts of its capacity for the large clients assigned to it by ϕ_L) and this capacity may not be enough to serve the demands of small clients moved to that location. In order to rectify this, we will fractionally redistribute the demands of these small clients between facilities (in I''_S) in such a way that we do not violate capacities u'_i. In this redistribution, we only use the edges used in ϕ_L or ϕ^*_L and if an edge is used to assign large client j to facility i (in ϕ_L or ϕ^*_L), we move at most d_j units of demands of small clients along this edge. Therefore, we pay at most $c_s(\phi_L) + c_s(\phi^*_L)$ in this redistribution. Thus, by the triangle inequality, the connection cost of the fractional assignment of small clients obtained at the end is bounded by $c_s(\phi^*_S) + c_s(\phi^*_L) + c_s(\phi_L) = c_s(\phi^*) + c_s(\phi_L)$. Since we only open facilities in the optimum solution (on top of what is already open in I_L) the extra facility cost (for assignment ϕ''_S) is bounded by the facility cost of the optimum.

This process of moving the small client demands can be alternatively thought of in the following way. We start from the optimum assignment ϕ^* and change the assignment of large clients to get an assignment identical to ϕ_L for those in L. Specifically, we change the assignment of a large client j from $i' = \phi^*(j)$ to $i = \phi_L(j)$. This switch increases the amount of demands served at i by d_j and decreases the amount of demand served at i' by d_j. After doing all these switches, we might have more demand at some facilities than their capacities. To resolve this problem, we try to redistribute (fractionally) the demands of small clients so that there is no capacity violation due to these clients and we use the max-flow min-cut theorem to show that this redistribution is possible.

Let s^*_i be the total demand of small clients served by facility i in ϕ^*. Define a flow network H as follows. H has set of vertices $X \cup Y \cup \{s, t\}$ where $X = \{x_i | i \in F\}$ and $Y = \{y_i | i \in F\}$, and s is called the source and t is the sink. We add an edge from s to all the vertices in X and set the capacity of edge sx_i to s^*_i (this represents the total demand of small clients that can be moved *from* facility i). We connect each

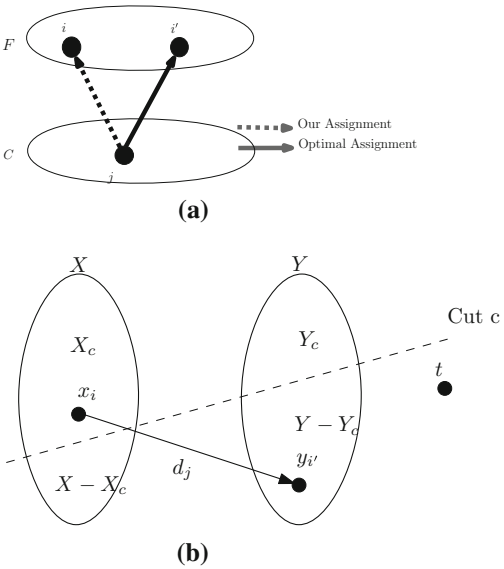

Fig. 1 The flow network used in the proof of Lemma 1. **a** Assignment of large client j in ϕ_l^* and ϕ_L. **b** The edge $x_i\, y_{i'}$ corresponding to the large client j shown in part (**a**). Also, an arbitrary $s - t$ cut of bounded value in H shown with a *dashed line*. The portion above the *dashed line* shows the partition $s \cup X_c \cup Y_c$

$y_i \in Y$ to t and set the capacity of edge $y_i t$ to u'_i (this represents the residual capacity of facility i after the assignment of large clients according to ϕ_L). We connect x_i and y_i bidirectionally with edges of unlimited capacity (because x_i and y_i represent the same facility). Finally, for each large client j with $\phi_L(j) = i$ and $\phi^*(j) = i'$ we add an edge from x_i to $y_{i'}$ with capacity d_j (see Fig. 1). This means that we can transport d_j units of (small clients) demands from facility i to facility i' since we switch the demand of large client j from being served at i' in the optimum to be served at i.

A flow of value $\sum_i s_i^*$ in H represents a fractional redistribution of demands of small clients between facilities in such a way that we do not violate capacities u'_i. It follows that if there is a flow of value $\sum_i s_i^*$ in H then we can redistribute the demands of small clients among facilities so that large clients are served according to ϕ_L and no facility capacity is violated because of small clients. We show that a maximum s–t flow in H has value at least $\sum_i s_i^*$.

We show that the capacity of any s–t cut in H is at least $\sum_i s_i^*$. Therefore, the capacity of minimum cut is at least $\sum_i s_i^*$ and by the max-flow min-cut theorem, the value of max flow is at least $\sum_i s_i^*$. If for any $i \in F$, the vertices x_i and y_i are in different partitions of a cut, the cut has unbounded capacity (because of the unlimited capacity of bidirectional edge $x_i\, y_i$) and clearly has capacity at least $\sum_i s_i^*$.

Now, consider an arbitrary cut in which for all $i \in F$, the vertices x_i and y_i are in the same partition. Consider the partition containing s in this cut and let $X_c \subseteq X$ and $Y_c \subseteq Y$ be the rest of the vertices in this partition, i.e., the partition is $s \cup X_c \cup Y_c$. Let F' be the facilities corresponding to the vertices in X_c (or equivalently, in Y_c). Let C_l^* be the large clients assigned to the facilities in F' by ϕ^* and C_l be the large clients assigned to the facilities in F' by ϕ_L. Since ϕ^* does not violate capacities, the total demand of clients assigned to facilities in F' is at most their total capacity, i.e., $|F'|$.

In other words:

$$\sum_{i\in F'} s_i^* + \sum_{j\in C_l^*} d_j \le |F'| \le \sum_{i\in F'} u_i' + \sum_{j\in C_l} d_j,$$

where the second inequality follows from the way that we defined u_i' and C_l. Adding the term $\sum_{i\in F\setminus F'} s_i^* - \sum_{j\in C_l^*} d_j$ to both sides:

$$\sum_{i\in F'} s_i^* + \sum_{i\in F\setminus F'} s_i^* \le \sum_{i\in F\setminus F'} s_i^* + \sum_{i\in F'} u_i' + \sum_{j\in C_l} d_j - \sum_{j\in C_l^*} d_j,$$

and by the simple fact that $\sum_{j\in C_l} d_j - \sum_{j\in C_l^*} d_j \le \sum_{j\in C_l\setminus C_l^*} d_j$, the above inequality implies

$$\sum_{i\in F} s_i^* \le \sum_{i\in F\setminus F'} s_i^* + \sum_{i\in F'} u_i' + \sum_{j\in C_l\setminus C_l^*} d_j.$$

Notice that the right hand side of the above inequality is exactly the capacity of the cut (see Fig. 1). The first term of the right hand side is the capacity of the edges leaving the cut from s to $X\setminus X_c$. The second term is the capacity of the edges leaving the cut from the vertices in Y_c to t. The third term is the capacity of the edges leaving the cut from X_c to $Y\setminus Y_c$, because a client j is in $C_l\setminus C_l^*$ if and only if it is assigned to a facility in F' by ϕ_L and is assigned to a facility outside of F' by ϕ^* if and only if we have an edge of capacity d_j from $\phi_L(j)\in X_c$ to $\phi^*(j)\in Y\setminus Y_c$. □

Remark 1 The above lemma can be generalized in the following way. Assume facility i has capacity u_i, i.e., we are in the non-uniform case. Let $\phi_{C'} : C' \to I_{C'}$ and $\phi_C : C \to I_C$ be two arbitrary capacity respecting assignments, where C' can be any subset of C, and $I_{C'}$ and I_C are subsets of F. Let $C'' = C\setminus C'$. Almost the same proof as above shows that given the assignment $\phi_{C'}$, there exists a feasible fractional assignment $\phi_{C''} : C'' \to I_{C''}$ such that $c_s(\phi_{C''}) \le c_s(\phi_C) + c_s(\phi_{C'})$ and $c_f(\phi_{C''}) \le c_f(\phi_C)$, where $I_{C''} \subseteq I_C \cup I_{C'}$. If we set $C' = L$ and $\phi_C = \phi^*$, then $C'' = S$ and we get the above lemma.

We point out that the above lemma is tight. Consider a path $P = v_1v_2v_3$ where the edges have unit cost. Assume $1/2\epsilon$ is an integer and denote this integer by q. There are q small clients of demand ϵ on v_1, there is a facility with opening cost 0 on v_1, there is a large client with demand $1/2$ on v_2, there is another facility with opening cost 0 on v_3 and there are $2q$ small clients of demand ϵ on v_3. In the optimal solution, we open both facilities and assign the large client on v_2 to the facility on v_1. Thus, we have $c_s(\phi^*) = 1/2$ and $c_f(\phi^*) = 0$. An algorithm which decides the assignment of large clients oblivious to small clients (including Algorithm 1) may assign the large client on v_2 to the facility on v_3. After this assignment, any feasible fractional assignment of small clients, ϕ_S'', must assign at least $1/2$ units of demands of small clients on v_3 to the facility on v_1. Thus, we have $c_s(\phi_S'') \ge 1/2 + 1/2 = 1 = c_s(\phi^*) + c_s(\phi_L)$, because $c_s(\phi_L) = 1/2$ and $c_f(\phi_S'') = c_f(\phi^*) = 0$. This shows that Lemma 1 is tight. Note that

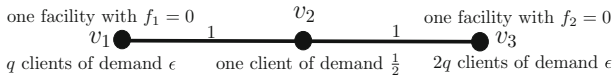

Fig. 2 A tight example for Lemma 1

this example shows that any algorithm which decides the assignment of large clients oblivious to small clients is at least a factor 3 away from the optimum even for simple metrics such as Euclidean, planar, and tree metrics (Fig. 2).

One should note that when all facility costs and at least one of $c_s(\phi_L^*)$ or $c_s(\phi_L)$ are non-zero, Lemma 1 is not tight: remember that I^* is the set of facilities opened by ϕ^*. Assume the facility cost inequality of Lemma 1 is tight, i.e., for any feasible fractional assignment of small clients ϕ_S'', we have $c_f(\phi_S'') = c_f(\phi^*)$. This means that none of the large clients is assigned to I^* by ϕ_L or equivalently, no small client is assigned to a facility opened by ϕ_L. Therefore, when we switch the assignment of large clients from the one in the optimal solution to the one in ϕ_L, we do not need to send back anything and change the assignment of small clients. Since at least one of $c_s(\phi_L^*)$ or $c_s(\phi_L)$ is non-zero, we have $c_s(\phi_S'') < c_s(\phi_S^*) + c_s(\phi_L^*) + c_s(\phi_L)$ for a ϕ_S'' that assigns small clients the same way as ϕ_S^*.

Now, we prove our main theorem using Lemma 1:

Proof of Theorem 1 Since the cost of the optimum solution for the instance consisting of only the large clients is clearly no more than that of the original instance, after Step 1 of Algorithm 1, we have an assignment ϕ_L such that $c(\phi_L) \leq \alpha(\epsilon)c(\phi_L^*)$ and it violates the capacities by a factor of at most $\beta(\epsilon)$. By Lemma 1, given ϕ_L, there is a feasible fractional assignment ϕ_S'' for small clients such that $c_s(\phi_S'') \leq c_s(\phi^*) + c_s(\phi_L)$ and $c_f(\phi_S'') \leq c_f(\phi^*)$.

In Step 1, consider the instance of the splittable CFLP consisting of the small clients and the residual facility opening costs and capacities as defined. We use an approximation algorithm for the splittable CFLP to find an approximate splittable (i.e., fractional) assignment ϕ_S' for small clients. Suppose that the approximation algorithm used for the splittable CFLP has separate factors $\lambda_{ss}, \lambda_{sf}, \lambda_{fs}, \lambda_{ff}$ such that it returns an assignment with service cost at most $\lambda_{ss}c_s(\tilde{\phi}_S) + \lambda_{sf}c_f(\tilde{\phi}_S)$ and with opening cost $\lambda_{fs}c_s(\tilde{\phi}_S) + \lambda_{ff}c_f(\tilde{\phi}_S)$ for any feasible solution $\tilde{\phi}_S$. Therefore, if we use the fractional assignment ϕ_S'' guaranteed by Lemma 1:

$$c_s\left(\phi_S'\right) \leq \lambda_{ss}c_s\left(\phi_S''\right) + \lambda_{sf}c_f\left(\phi_S''\right), \tag{1}$$

and

$$c_f\left(\phi_S'\right) \leq \lambda_{fs}c_s\left(\phi_S''\right) + \lambda_{ff}c_f\left(\phi_S''\right). \tag{2}$$

The current best approximation for the non-uniform splittable CFLP is due to Bansal et al. [7] with parameters $\lambda_{ss} = 1$, $\lambda_{sf} = 1$, $\lambda_{fs} = 4$, and $\lambda_{ff} = 4$.

In Step 1, we round the splittable assignment ϕ_S' using the algorithm of Shmoys and Tardos [34] for the Generalized Assignment Problem (GAP) to find an integer assignment ϕ_S. Given the fractional assignment of clients to facilities ϕ_S', using the rounding algorithm of [34], we can round ϕ_S' to ϕ_S without increasing the connection

cost, i.e., $c_s(\phi_S) = c_s(\phi'_S)$, such that the capacity constraints are violated by at most an additive factor of $\max_{j \in S} d_j$. Since all the jobs in S have demand at most ϵ, the capacity constraints are violated by at most a factor of $1 + \epsilon$.

After combining ϕ_S and ϕ_L in Step 1, the violation of capacities is within a factor of at most $\max\{\beta(\epsilon), (1+\epsilon)\}$, because the facilities with violated capacities in Step 1 will be removed in Step 1 and will not be used in Step 1. So it only remains to bound the cost of this assignment:

$$\begin{aligned}
c_s(\phi_S) &= c_s\left(\phi'_S\right) && \text{by rounding of the GAP [34]}\\
&\le \lambda_{ss} c_s\left(\phi''_S\right) + \lambda_{sf} c_f\left(\phi''_S\right) && \text{by Eq. (1)}\\
&\le \lambda_{ss}\left(c_s\left(\phi^*\right) + c_s\left(\phi_L\right)\right) + \lambda_{sf} c_f\left(\phi^*\right), && \text{by Lemma 1}\\
c_f(\phi_S) &\le (1+\epsilon) c_f\left(\phi'_S\right) && \text{by rounding of the GAP [34]}\\
&\le (1+\epsilon)\lambda_{fs} c_s\left(\phi''_S\right) + (1+\epsilon)\lambda_{ff} c_f\left(\phi''_S\right) && \text{by Eq. (2)}\\
&\le (1+\epsilon)\lambda_{fs}(c_s\left(\phi^*\right)+c_s(\phi_L))+(1+\epsilon)\lambda_{ff} c_f\left(\phi^*\right). && \text{by Lemma 1}
\end{aligned}$$

Therefore:

$$\begin{aligned}
c(\phi) &= c(\phi_S) + c(\phi_L)\\
&= c_s(\phi_S) + c_f(\phi_S) + c_s(\phi_L) + c_f(\phi_L)\\
&\le h_1(\epsilon) c_s\left(\phi^*\right) + h_2(\epsilon) c_f\left(\phi^*\right) + (h_1(\epsilon)+1) c_s(\phi_L) + c_f(\phi_L), \qquad (3)
\end{aligned}$$

where $h_1(\epsilon) = \lambda_{ss} + (1+\epsilon)\lambda_{fs}$ and $h_2(\epsilon) = \lambda_{sf} + (1+\epsilon)\lambda_{ff}$. Since $h_1(\epsilon) \ge 0$ for any $\epsilon > 0$: $(h_1(\epsilon)+1) c_s(\phi_L) + c_f(\phi_L) \le (h_1(\epsilon)+1) c(\phi_L) \le \alpha(\epsilon)(h_1(\epsilon)+1) c(\phi^*_L) \le \alpha(\epsilon)(h_1(\epsilon)+1) c(\phi^*)$. Combining this with Inequality (3), we obtain that the cost of ϕ is within factor:

$$g(\epsilon, \alpha(\epsilon)) = \max(h_1(\epsilon), h_2(\epsilon)) + \alpha(\epsilon)(h_1(\epsilon)+1) \qquad (4)$$

of the optimum, where $h_1(\epsilon) = h_2(\epsilon) = 5 + 4\epsilon$ using the current best ratio in [7]. □

3 The RUCFLP$\left(\frac{1}{2}\right)$ and RUCFLP$\left(\frac{1}{3}\right)$

In this section, we give two algorithms for the RUCFLP($\frac{1}{2}$) and RUCFLP($\frac{1}{3}$). Combined with Theorem 1 (and using Algorithm 1) these imply two approximation algorithms for the UCFLP. We start with the simpler of the two, namely the RUCFLP($\frac{1}{2}$).

Theorem 5 *There is a polynomial time exact algorithm for the RUCFLP($\frac{1}{2}$).*

Proof Consider an optimal solution for a given instance of this problem with value OPT_L. Because $d_j > \frac{1}{2}$ for all $j \in C$, each facility can serve at most one client in the optimal solution. Therefore, the optimal assignment function, ϕ^*_L, induces a matching $M = \{j\phi^*_L(j) : j \in C\}$. Let $w_{ij} = c_{ij}.d_j + f_i$ and let $w(H) = \sum_{e \in H} w_e$ for any subset of edges $H \subseteq E$. It follows that $w(M) = OPT_L$.

Let M^* be a minimum weight perfect matching with respect to weights w_{ij}. Clearly, $w(M^*) \le w(M) = OPT_L$. In addition, M^* induces a feasible assignment of clients to

facilities with cost $w(M^*)$. Thus, M^* induces an optimal solution for the RUCFLP($\frac{1}{2}$). Since we can find a minimum weight perfect matching in polynomial time, there is an exact algorithm for the RUCFLP($\frac{1}{2}$). □

Corollary 2 *There is a polynomial time* $(15, 3/2)$*-approximation algorithm for the UCFLP.*

Proof We run Algorithm 1, where we use the algorithm of Theorem 5 in the first step. Substituting $\alpha(\epsilon) = 1$ and $\epsilon = 1/2$, we have $h_1(\frac{1}{2}) = 7$, $h_2(\frac{1}{2}) = 7$, and $g(\epsilon, \alpha(\epsilon)) = 15$. Since $\beta(\epsilon) = 1$, the overall ratio is $(15, 3/2)$. □

Remark 2 We can generalize the above Theorem. Let ϕ_L be the assignment induced by M^* and $\tilde{\phi_L}$ be any feasible assignment for the RUCFLP($\frac{1}{2}$) instance. The reader may verify that a similar proof shows that $c_s(\phi_L) + c_f(\phi_L) \leq c_s(\tilde{\phi_L}) + c_f(\tilde{\phi_L})$.

The algorithm for the RUCFLP($\frac{1}{3}$) is more involved. First, we show how finding an approximation algorithm for the RUCFLP(ϵ) with zero facility opening costs leads to an approximation algorithm for the general RUCFLP(ϵ). Then, we give an approximation algorithm for the RUCFLP($\frac{1}{3}$) with zero opening costs.

Lemma 2 *Given an* $(\alpha'(\epsilon), \beta(\epsilon))$*-approximation algorithm* $\mathcal{A}'$ *for the RUCFLP*(ϵ) *with zero facility opening costs, we can find a* $(\alpha'(\epsilon)\frac{1}{\epsilon}, \beta(\epsilon))$*-approximation* $\mathcal{A}$ *for the general RUCFLP*(ϵ)*.*

Proof Define a new connection cost $c'_{ij} = c_{ij} + f_i$ and opening cost $f'_i = 0$ for all $i \in F$ and $j \in C$. Note that the new cost function is still metric. Then, we run $\mathcal{A}'$ on this new modified instance and let the solution returned by the algorithm be assignment ϕ_L. We use ϕ_L to assign the clients for the original instance and we claim this is a $(\alpha'(\epsilon)\frac{1}{\epsilon}, \beta(\epsilon))$-approximation. In the following, the costs of all assignments are based on c_{ij} and f_i values.

Let ϕ_L^* be an optimal assignment for the original instance of the RUCFLP(ϵ) and OPT_L be the cost of ϕ_L^* (including opening costs). Let $C_i = \phi_L^{*-1}(i)$. The cost of ϕ_L^* in the new instance will be

$$\sum_i \sum_{j \in C_i} d_j c'_{ij} = \sum_i \left(\sum_{j \in C_i} d_j c_{ij} + \left(\sum_{j \in C_i} d_j \right) . f_i \right) \leq \sum_i \left(\sum_{j \in C_i} d_j c_{ij} + f_i \right) \leq OPT_L,$$

where we used the fact that $\sum_{j \in C_i} d_j \leq 1$. Thus, there is a solution in the modified instance with cost at most OPT_L. Therefore, the value of the assignment found by $\mathcal{A}'$ in the new graph is at most $\alpha'(\epsilon) \cdot OPT_L$. Let γ be the portion of this value that comes from the facility costs in c' costs, i.e., the cost of solution ϕ_L in the new graph is $c_s(\phi_L) + \gamma \leq \alpha'(\epsilon) OPT_L$. Since $d_j > \epsilon$ for all $j \in C$, each client j pays at least ϵ fraction of the opening cost of $\phi_L(j)$ embedded in costs c' and hence, $c_f(\phi_L) \leq \frac{1}{\epsilon}\gamma$. Therefore, we have

$$c(\phi_L) = c_s(\phi_L) + c_f(\phi_L) \leq c_s(\phi_L) + \frac{1}{\epsilon}\gamma \leq \frac{1}{\epsilon} c_s(\phi_L) + \frac{1}{\epsilon}\gamma \leq \frac{1}{\epsilon}\alpha'(\epsilon) OPT_L$$

for $\epsilon \leq 1$. □

Remark 3 Let $\tilde{\phi_L}$ be any feasible assignment for the original instance of the RUCFLP(ϵ). A similar proof shows that $c_s(\phi_L) + \gamma \leq \alpha'(\epsilon)(c_s(\tilde{\phi_L}) + c_f(\tilde{\phi_L}))$. We use this fact in Sect. 3.1 to get a better ratio for the RUCFLP($\frac{1}{3}$).

In the following, we present a (7, 1)-approximation algorithm for the RUCFLP($\frac{1}{3}$) with zero opening costs (see Algorithm 2), which coupled with Lemma 2 yields a (21, 1)-approximation algorithm for the RUCFLP($\frac{1}{3}$).

Algorithm 2 Algorithm for solving the RUCFLP($\frac{1}{3}$) with zero opening costs

Input: An instance of the RUCFLP($\frac{1}{3}$) with zero opening costs
Output: A subset $I \subseteq F$ and a function $\phi : C \to I$
1: Let $L' = \{j \in C : d_j > \frac{1}{2}\}$ and $L'' = C \backslash L'$. Assign the clients in L' by running a minimum weight perfect matching algorithm with edge weights $w_{ij} = d_j c_{ij}$. Let $I_{L'}$ be the opened facilities and $\phi_{L'} : L' \to I_{L'}$ be the assignment function.
2: Build the flow network H as described in the proof of Theorem 6.
3: Find a minimum cost maximum flow in H. If the value of the flow is smaller than $|L''|$ then return "Infeasible". Else, let $I_{L''}$ be the subset of facilities in $F \backslash I_{L'}$ whose corresponding nodes in Y (in H) have non-zero flow through them and $\phi_{L''}$ be the assignment function defined as: if there is a unit flow from x_j to y_i in H then $\phi_{L''}(j) = i$.
4: Let $I = I_{L''} \cup I_{L'}$. Combine $\phi_{L''}$ and $\phi_{L'}$ to form assignment function $\phi_L : C \to I$ where $\phi(j) = \phi_{L''}(j)$ if $j \in L''$, otherwise $\phi(j) = \phi_{L'}(j)$. Return ϕ and I.

Theorem 6 *There is a* (7, 1)*-approximation algorithm for the RUCFLP($\frac{1}{3}$) with zero opening costs.*

Proof Note that all the clients in the given instance have size $> \frac{1}{3}$. We break them into two groups: $L' = \{j \in C : d_j > \frac{1}{2}\}$ and $L'' = C \backslash L'$ are those which have size in $(\frac{1}{3}, \frac{1}{2}]$. In this proof (and that of Lemma 3), we call clients in L', *huge* clients and those in L'', medium clients. The algorithm assigns the huge clients by running a minimum weight perfect matching algorithm with edge weights $w_{ij} = d_j c_{ij}$. Let $I_{L'}$ be the opened facilities and $\phi_{L'} : L' \to I_{L'}$ be the assignment function. For medium clients (i.e., those in L''), we define a flow-network H and show that minimum cost maximum flows in H corresponds to minimum cost feasible assignment of clients in L'' to facilities (given the assignment $\phi_{L'}$).

Directed network H has node set $X \cup Y \cup \{s, t\}$ where there is a node $x_j \in X$ for every client $j \in L''$ and a node $y_i \in Y$ for every facility $i \in F$; s is the source and t is the sink. The source is connected to each node $x_j \in X$, and all $y_i \in Y$ are connected to the sink. Each $x_j \in X$ is connected to a node $y_i \in Y$ if either the corresponding facility i is in $F \backslash I_{L'}$, i.e., unopened yet, or i is in $I_{L'}$ and the remaining capacity of i is enough to serve the demand of client j. Set the capacity of the edges between the source and the nodes in X to 1, set the capacity of the edges between X and Y to 1, set the capacity of the edges between the nodes $y_i \in Y$ whose corresponding facility i is unopened (i.e., not in $I_{L'}$) and the sink to 2, and set the capacity of the edges between the nodes $y_i \in Y$ whose corresponding facility is in $I_{L'}$ and the sink to 1. The

cost of an edge connecting $x_j y_i$ is $d_j \cdot c_{ij}$ and all the other costs are 0. Algorithm 2 summarizes the algorithm for the RUCFLP($\frac{1}{3}$) with zero opening costs.

Let ϕ_L^* be an optimal assignment for the given instance of the RUCFLP($\frac{1}{3}$) with cost OPT_L. In Lemma 3, we will prove that there exists an assignment ϕ' of clients consistent with assignment $\phi_{L'}$ found in Step 2, with cost at most $7OPT_L$. Below we prove that in Steps 2 and 2 the algorithm finds the best possible feasible assignment of clients in L'' (given $\phi_{L'}$). Therefore, the cost of ϕ formed in Step 2 is at most $c(\phi')$ and hence, is at most $7OPT_L$.

Since for any $j \in L''$: $\frac{1}{3} < d_j \leq \frac{1}{2}$, each unopened facility after Step 2 can serve any two clients of L'' (and no more than two). This fact is reflected in that we connect all the nodes in X (corresponding to medium clients) to the nodes in Y corresponding to unopened facilities $F \backslash I_{L'}$ and we set the capacity of the edges between those nodes in Y and the sink to 2. In addition, each facility in $I_{L'}$ can serve at most one medium client, because more than $\frac{1}{2}$ of its capacity is already used by a huge client; accordingly we set the capacity of the edges from those nodes in Y to the sink to 1 and we only connect to them the nodes of X whose corresponding client can be served by them. Considering these two simple facts: □

Claim 1 *The maximum flow in H has value $|L''|$ if and only if the given instance is feasible and there is a one to one correspondence between maximum flows in H and feasible assignment of medium clients (i.e., in L'') given $\phi_{L'}$. Furthermore, a maximum flow in H and its corresponding assignment of clients of L'' to F have the same cost.*

Proof Since all the edges of H have integer capacities we may consider an integral maximum flow. If a node $x_j \in X$ has a flow of one to a node $y_i \in Y$ we assume client j is assigned to facility i, and vice-versa. First, suppose that the instance is feasible and let ϕ be an arbitrary feasible assignment. We show that there is a feasible assignment consistent with $\phi_{L'}$. Let $I_{L'}^{(\phi)}$ be the set of open facilities in ϕ to which a client of L' is assigned. Clearly, $|I_{L'}^{(\phi)}| = |L'|$. Since opening costs are zero and all facilities have the same capacity, we can easily swap the facilities in $I_{L'}^{(\phi)}$ with ones to which a client of L' is assigned to in $\phi_{L'}$ so that we get a feasible assignment consistent with $\phi_{L'}$. Then it is easy to see that H has a flow of value $|L''|$ (basically the edges between X and Y in H with non-zero flow correspond to the assignment of clients of L'' in the feasible solution consistent with $\phi_{L'}$). Conversely, if H has a flow of $|L''|$ then that corresponds to a feasible assignment of medium clients to facilities (consistent with $\phi_{L'}$). The correspondence between cost of a maximum flow and an assignment of clients (of L'') is immediate from the definition of costs of edges. □

Therefore, the assignment $\phi_{L''}$ obtained from a minimum cost maximum flow in H has the minimum cost among the assignments consistent with $\phi_{L'}$. This together with Lemma 3 implies that ϕ as defined has cost at most $7OPT_L$. □

Lemma 3 *There exists an assignment ϕ'_L of clients consistent with $\phi_{L'}$ with cost at most $7OPT_L$, where OPT_L is the cost of an optimum assignment ϕ_L^* for the given instance of the RUCFLP($\frac{1}{3}$) with zero opening costs.*

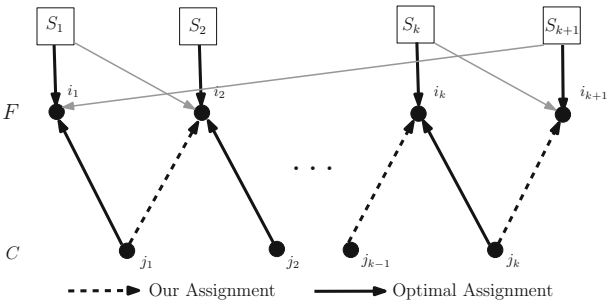

Fig. 3 An arbitrary path in $G_{L'}$. In ϕ', the clients in S_l are assigned to the facility pointed to by the *gray arrow* adjacent to S_l

Proof Let $\phi^*_{L'}$ be the restriction of ϕ^*_L to clients in L', and $G_{L'}$ be the graph induced by the edges used in $\phi_{L'}$ or $\phi^*_{L'}$. Since at most one client in L' is assigned to each facility by any feasible solution, the degree of vertices in $G_{L'}$ is at most 2; so $G_{L'}$ is a collection of cycles and paths. For an arbitrary path, we show how the assignment of huge clients in ϕ^*_L (i.e., those in L') can be changed to the ones in $\phi_{L'}$ without violating any capacity, while the cost increases by at most a factor of 7. Similarly, each cycle can be fixed with at most a factor of 3 increase in its cost, which completes the proof.

Let $P = i_1 j_1 i_2 j_2 \dots i_k j_k i_{k+1}$ be an arbitrary path in $G_{L'}$ where $\phi^*_{L'}(j_l) = i_l$ and $\phi_{L'}(j_l) = i_{l+1}$ for $1 \le l \le k$ (see Fig. 3). Let S_l be the set of medium clients (i.e., in L'') assigned to facility i_l by ϕ^*_L. In ϕ', we assign j_l and the clients in S_l to i_{l+1} for $1 \le l \le k$ and we assign the clients in S_{k+1} to i_1. Since we used a min-cost maximum matching algorithm to find $\phi_{L'}$, we have $c(\phi_{L'}) \le c(\phi^*_{L'})$, and changing the assignment of clients in L' does not increase the cost (compared to that of ϕ^*_L). Therefore, we only need to analyse the increase in cost because of the change in assignment of medium clients. Note that due to uniformity, the capacity constraints remain satisfied.

Consider an arbitrary l where $1 \le l \le k$. Since $d_{j_l} > \frac{1}{2}$, the total demand of clients in S_l is less than d_{j_l} (because $1 - d_{j_l} < 1/2 < d_{j_l}$). Therefore, if we send the clients in S_l to i_{l+1} over the edges $i_l j_l$ and $j_l i_{l+1}$, we increase the cost by at most $d_{j_l}(c_{i_l j_l} + c_{j_l i_{l+1}})$. Thus, in total over all paths in $G_{L'}$, changing the assignment of clients in $\cup_{1 \le l \le k} S_l$ increases the cost by at most an additive $c(\phi_{L'}) + c(\phi^*_{L'}) \le 2OPT_L$. Finally, the total demand of clients in S_{k+1} is at most $1 < 2d_{j_l}$ for any $1 \le l \le k$. Therefore, sending back these clients to i_1 over P increases the cost in total by at most an additive $2c(\phi_{L'}) + 2c(\phi^*_{L'}) \le 4OPT_L$. Cycles in $G_{L'}$ are handled the same way as paths, but with nodes i_1 and i_{k+1} identified. We conclude that the reassignment of clients increases cost by at most an additive $6OPT_L$, making the overall cost of ϕ' at most $7OPT_L$. □

The above lemma is essentially tight: there are instances of the RUCFLP($\frac{1}{3}$) with zero costs that after deciding the assignment of huge clients by a minimum weight perfect matching algorithm, any feasible solution consistent with this assignment has cost at least $7OPT_L$ (see Fig. 4). Consider a path $P = v_1 v_2 v_3$ where the edges have unit cost. There are two facilities with zero opening cost on v_1 and v_3. In addition, there is a medium client of demand $1/2 - \delta$ on v_1, a huge client of demand $1/2 + \delta$ on v_2, and two medium clients of demand $1/2$ on v_3, where δ is a small positive number. In the optimal solution, the client on v_2 is assigned to the facility on v_1 and

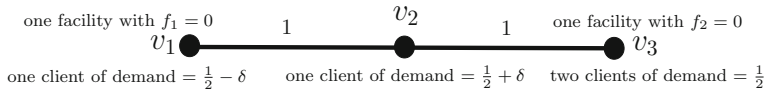

Fig. 4 A tight example for Lemma 3

$OPT_L = 1/2 + \delta$. In our solution, the minimum weight perfect matching algorithm may assign the huge client on v_2 to the facility on v_3. Then, in any feasible solution, we must move the medium client on v_1 to v_3 and move the two medium clients on v_3 to v_1, so the total cost of any feasible solution given the assignment of the huge client, is at least $(1/2 - \delta) \times 2 + 1 \times 2 + (1/2 + \delta) \times 1 = 7/2 - \delta$. This shows any feasible solution is essentially a factor of 7 away from the optimum.

Combining Lemma 2 and Theorem 6:

Corollary 3 *There is a polynomial time* $(21, 1)$*-approximation algorithm for the RUCFLP*$(\frac{1}{3})$.

Corollary 4 *There is a (160.334,4/3)-approximation algorithm for the UCFLP.*

Proof We run Algorithm 1, where we use the algorithm of Corollary 3 for $\mathcal{A}$. That is, we first run the $(7, 1)$-approximation algorithm of Theorem 6 as algorithm $\mathcal{A}'$ in Lemma 2 to obtain $\mathcal{A}$ with $\alpha(\epsilon) = 21$ and $\epsilon = 1/3$. Thus $h_1(\frac{1}{3}) = 19/3$, $h_2(\frac{1}{3}) = 19/3$, and $g(\epsilon, \alpha(\epsilon)) = 19/3 + 21(22/3) < 160.334$. Since $\beta(\epsilon) = 1$, the overall ratio is $(160.334, 4/3)$. □

Notice that we solved the RUCFLP$(\frac{1}{2})$ and $(\frac{1}{3})$ without violation of capacities, but this is not possible for smaller values of ϵ as shown below.

Theorem 7 *The RUCFLP*(ϵ) *does not admit any* $(\alpha(\epsilon), 1)$*-approximation algorithm for* $\epsilon < \frac{1}{3}$ *unless* $\boldsymbol{P = NP}$.

Proof This can be shown by a simple reduction from the 3-partition problem (which is **NP**-hard). In the 3-partition problem, we are given a set of $3m$ integers $a_1, \ldots, a_{3m}$, a positive integer bound B where $\frac{B}{4} < a_j < \frac{B}{2}$ for all $1 \leq j \leq 3m$ and $mB = \sum_{1 \leq j \leq 3m} a_j$. The question is if there is a way to partition these numbers into m sets of size 3 each such that the sum of the numbers in each set is exactly B. This problem is **NP**-hard [19].

Starting from a given instance $\mathcal{I}_p$ of the 3-partition problem, we build an instance $\mathcal{I}_R$ of the RUCFLP(ϵ) in the following way. Let c be a constant dependent on ϵ, which we define soon. For each $1 \leq j \leq 3m$, let $d_j = \frac{a_j + cB}{B(3c+1)}$ and create a client j with demand d_j. Also, create m facilities with zero opening cost and capacity 1. We set all the connection costs $c_{ij} = 0$. We choose constant c large enough such that $\epsilon < \frac{1}{3} - \frac{1}{12(3c+1)}$. Since $\frac{B}{4} < a_j < \frac{B}{2}$ for all $1 \leq j \leq 3m$, the value of the demands are between $\frac{1}{3} - \frac{1}{12(3c+1)}$ and $\frac{1}{3} + \frac{1}{6(3c+1)}$ and clearly, are greater than ϵ by the choice of c. This completes the description of instance $\mathcal{I}_R$.

First, note that if we define $a'_j = \delta_1 a_j + \delta_2$ for all $1 \leq j \leq 3m$ and $B' = \delta_1 B + 3\delta_2$ for two positive constants δ_1 and δ_2, then this new instance is a yes instance of the

3-partition problem if and only if $\mathcal{I}_p$ is a yes instance. In the above reduction, we used $\delta_1 = \frac{1}{B(3c+1)}$ and $\delta_2 = \frac{c}{3c+1}$ to define d_j values. Thus, $\mathcal{I}_p$ is a yes instance if and only if we can partition d_j values to m sets of size 3 each such that the sum of numbers in each set is exactly $B' = 1$. Since any solution for $\mathcal{I}_R$, that does not violate the capacity constraints, provides such a partition of d_j values, an $(\alpha(\epsilon), 1)$-approximation algorithm for the RUCFLP(ϵ) can be used to solve the 3-partition problem in polynomial time. Therefore, unless $\mathbf{P} = \mathbf{NP}$, there is no such approximation algorithm. □

It should be noted that to find an algorithm for the UCFLP that violates capacities within factor $1+\epsilon$, we do not need to find an algorithm that does not violate capacities in the RUCFLP(ϵ). Even if we violate the capacities within factor $1+\epsilon$ in the RUCFLP(ϵ), using Theorem 1 we can get an algorithm for the UCFLP that violates the capacities within factor $1+\epsilon$. We think it is possible to find an $(\alpha(\epsilon), 1+\epsilon)$-approximation for the RUCFLP(ϵ) for any constant $\epsilon > 0$. This, together with Theorem 1 would imply an $(f(\epsilon), 1+\epsilon)$-approximation for the UCFLP, for any constant $\epsilon > 0$.

3.1 Improving the Ratios

With a more careful analysis and a simple scaling to balance the bi-factors of connection and facility costs, we can bring down the factors of our algorithms. A similar scaling has been used to obtain better ratios for several variations of facility location problems (for example see [12]). For certain parameters δ_1 and δ_2 to be defined, we change Algorithm 1 as follows:

1. We multiply the original connection costs by δ_1 to get a new cost function $c^{(1)}$, i.e., $c^{(1)}_{ij} = \delta_1 c_{ij}$ for all $i \in F$ and $j \in C$. Then, we perform Step 1 with cost $c^{(1)}$ to find ϕ_L.
2. After step 1, we multiply the *original* connection costs by δ_2 to get a new cost function $c^{(2)}$ i.e., $c^{(2)}_{ij} = \delta_2 c_{ij}$ for all $i \in F$ and $j \in C$. Then, we do Steps 1 and 1 with cost $c^{(2)}$ to find ϕ_S.

In the following, for an assignment ϕ, we use $c_s(\phi)$, $c^{(1)}_s(\phi)$, and $c^{(2)}_s(\phi)$ to indicate the service cost of this assignment with respect to cost functions c, $c^{(1)}$, and $c^{(2)}$, respectively. In addition, assignments ϕ^* and ϕ^*_L have the same definition as before and are defined with respect to the original costs. As a result, they are not necessarily optimal with respect to cost functions $c^{(1)}$ and $c^{(2)}$.

By Remark 1 (applied to ϕ^*) there exists an assignment ϕ''_S such that $c^{(1)}_s(\phi''_S) \leq c^{(1)}_s(\phi^*) + c^{(1)}_s(\phi_L)$. Thus:

$$\begin{aligned} c^{(2)}_s\left(\phi''_S\right) &= \frac{\delta_2}{\delta_1} c^{(1)}_s\left(\phi''_S\right) \\ &\leq \frac{\delta_2}{\delta_1} c^{(1)}_s\left(\phi^*\right) + \frac{\delta_2}{\delta_1} c^{(1)}_s(\phi_L) \\ &= c^{(2)}_s\left(\phi^*\right) + c^{(2)}_s(\phi_L). \end{aligned} \tag{5}$$

In addition, the inequalities for $c_s(\phi_S)$ and $c_f(\phi_S)$ at the end of proof of Theorem 1 change to:

$$\begin{aligned} c_s^{(2)}(\phi_S) &= c_s^{(2)}\left(\phi'_S\right) \\ &\le \lambda_{ss} c_s^{(2)}\left(\phi''_S\right) + \lambda_{sf} c_f(\phi''_s) \\ &\le \lambda_{ss}(c_s^{(2)}\left(\phi^*\right) + c_s^{(2)}(\phi_L)) + \lambda_{sf} c_f\left(\phi^*\right), \quad \text{by Eq. (5)}, \qquad (6) \\ c_f(\phi_S) &\le (1+\epsilon) c_f\left(\phi'_S\right) \\ &\le (1+\epsilon)\lambda_{fs} c_s^{(2)}\left(\phi''_S\right) + (1+\epsilon)\lambda_{ff} c_f\left(\phi''_S\right) \\ &\le (1+\epsilon)\lambda_{fs}(c_s^{(2)}\left(\phi^*\right) + c_s^{(2)}(\phi_L)) + (1+\epsilon)\lambda_{ff} c_f\left(\phi^*\right). \quad \text{by Eq. (5)} \\ & \qquad (7) \end{aligned}$$

Therefore, scaling down Eq. (6) by δ_2 and using definition of $c^{(2)}$:

$$c_s(\phi_S) \le \lambda_{ss}(c_s\left(\phi^*\right) + c_s(\phi_L)) + \frac{\lambda_{sf}}{\delta_2} c_f\left(\phi^*\right).$$

Also, using definition of $c^{(2)}$ and Eq. (7):

$$c_f(\phi_S) \le \delta_2(1+\epsilon)\lambda_{fs}(c_s\left(\phi^*\right) + c_s(\phi_L)) + (1+\epsilon)\lambda_{ff} c_f\left(\phi^*\right).$$

Adding these inequalities together and then, adding $c_s(\phi_L) + c_f(\phi_L)$ to the result, we obtain:

$$\begin{aligned} c(\phi) &= c_s(\phi_S) + c_f(\phi_S) + c_s(\phi_L) + c_f(\phi_L) \\ &\le \hat{h}_1(\epsilon) c_s\left(\phi^*\right) + \hat{h}_2(\epsilon) c_f\left(\phi^*\right) + (\hat{h}_1(\epsilon)+1) c_s(\phi_L) + c_f(\phi_L) \qquad (8) \end{aligned}$$

where $\hat{h}_1(\epsilon) = \lambda_{ss} + \delta_2(1+\epsilon)\lambda_{fs}$ and $\hat{h}_2(\epsilon) = \frac{\lambda_{sf}}{\delta_2} + (1+\epsilon)\lambda_{ff}$. Using algorithm of Aggarwal et al. [1] for splittable CFLP, we have $\lambda_{ss} = \lambda_{sf} = 1$ and $\lambda_{fs} = \lambda_{ff} = 4$. Therefore, $\hat{h}_1(\epsilon) = 1 + 4\delta_2(1+\epsilon)$ and $\hat{h}_2(\epsilon) = \frac{1}{\delta_2} + 4(1+\epsilon)$. We use these equalities to get the improved results.

Proof of Theorem 2 We run the modified Algorithm 1 with $\delta_1 = \hat{h}_1(\frac{1}{2}) + 1$ and value of δ_2 to be defined soon. We have:

$$\begin{aligned} \left(\hat{h}_1\left(\frac{1}{2}\right) + 1\right) c_s(\phi_L) + c_f(\phi_L) &= c_s^{(1)}(\phi_L) + c_f(\phi_L) \\ &\le c_s^{(1)}\left(\phi_L^*\right) + c_f\left(\phi_L^*\right) \\ &= \left(\hat{h}_1\left(\frac{1}{2}\right) + 1\right) c_s\left(\phi_L^*\right) + c_f\left(\phi_L^*\right) \\ &\le \left(\hat{h}_1\left(\frac{1}{2}\right) + 1\right) c_s\left(\phi^*\right) + c_f\left(\phi^*\right), \end{aligned}$$

where the first inequality follows from Remark 2. Combining this with Inequality (8):

$$\begin{aligned} c(\phi) &\leq (2\hat{h}_1\left(\frac{1}{2}\right)+1)c_s\left(\phi^*\right)+(\hat{h}_2\left(\frac{1}{2}\right)+1)c_f\left(\phi^*\right) \\ &= (12\delta_2+3)c_s\left(\phi^*\right)+(1/\delta_2+7)c_f\left(\phi^*\right). \end{aligned}$$

Solving the equation $12\delta_2+3=1/\delta_2+7$ for δ_2, we find $\delta_2=(4+\sqrt{64})/24=\frac{1}{2}$, which gives the claimed ratio. □

Proof of Theorem 3 Recall that we use Lemma 2 to obtain algorithm $\mathcal{A}$ used in Algorithm 1 for large clients. We first start with a more careful analysis in Lemma 2. By Remark 3, the proof of this lemma implies:

$$c_f(\phi_L)\leq\frac{1}{\epsilon}\gamma \text{ and } c_s^{(1)}(\phi_L)+\gamma\leq\alpha'(\epsilon)(c_s^{(1)}(\phi_L^*)+c_f(\phi_L^*)).$$

In our case, we have $\epsilon=1/3$ and $\alpha'(\epsilon)=7$. Thus: $c_f(\phi_L)\leq 3\gamma\leq 3(7(c_s^{(1)}(\phi_L^*)+c_f(\phi_L^*))-c_s^{(1)}(\phi_L))$ which implies $3c_s^{(1)}(\phi_L)+c_f(\phi_L)\leq 21(c_s^{(1)}(\phi_L^*)+c_f(\phi_L^*))$. If we set $\delta_1=(\hat{h}_1(\frac{1}{3})+1)/3$, we have

$$\begin{aligned} \left(\hat{h}_1\left(\frac{1}{3}\right)+1\right)c_s(\phi_L)+c_f(\phi_L) &\leq 7\left(\hat{h}_1\left(\frac{1}{3}\right)+1\right)c_s(\phi_L^*)+21c_f(\phi_L^*) \\ &\leq 7\left(\hat{h}_1\left(\frac{1}{3}\right)+1\right)c_s\left(\phi^*\right)+21c_f\left(\phi^*\right). \end{aligned}$$

Combining this with Inequality (8):

$$\begin{aligned} c(\phi) &\leq \left(\hat{h}_1(1/3)+7\left(\hat{h}_1(1/3)+1\right)\right)c_s\left(\phi^*\right)+(\hat{h}_2(1/3)+21)c_f\left(\phi^*\right) \\ &= (8(16\delta_2/3+1)+7)c_s\left(\phi^*\right)+(1/\delta_2+16/3+21)c_f\left(\phi^*\right) \\ &= (128\delta_2/3+15)c_s\left(\phi^*\right)+(1/\delta_2+79/3)c_f\left(\phi^*\right) \end{aligned}$$

By solving the equation $128\delta_2/3+15=1/\delta_2+79/3$ for δ_2, we find $\delta_2=(17+\sqrt{673})/128$, which gives the claimed ratio. □

4 The Euclidean UCFLP

In this section, we present a quasi-polynomial time $(1+\epsilon,1+\epsilon)$-approximation algorithm for the UCFLP in Euclidean metrics. For these instances, we do not reduce our problem to the RUCFLP(ϵ) as we did in the previous sections. The reason is that one cannot get a ratio better than $(3,1+\epsilon)$ by applying this reduction as shown in the tight example of Lemma 1 (note that the cost function in that example is Euclidean).

Our algorithm for Euclidean metrics assigns the large clients integrally using a dynamic programming approach, while it assigns the small clients fractionally at the

same time at low cost. Then we can round the fractional assignment of the small clients by the algorithm of Shmoys and Tardos [34] for the GAP, as we did in Sect. 2. This rounding will not increase the cost and yields a blowup of factor at most $1+\epsilon$ in facility capacities. We should note that Bateni and Hajiaghayi [9] used this approach in the design of their $(1+\epsilon, 1+\epsilon)$-approximation algorithm for the UCFLP in the metrics induced by trees. We combine some of their ideas with techniques developed for designing PTASs for some Euclidean optimization problems; most notably the algorithms of [3,5]. While our algorithm has some similarities to that of [9] (such as the way we define client types), there are some technical differences. In particular, our table entries in the dynamic program are defined based on a shifted quad-tree that is obtained from a dissection of the plane and the way we combine solutions for the subproblems is more complicated.

We assume that the input points are on a unit grid, the minimum inter-node distance is at least 4 (which implies no two nodes are located on the same grid point), and the maximum inter-node distance is at most $O(n^4)$. We can enforce all these assumptions by a preprocessing step similar to the one used by Arora [4] to make the instances *well-rounded* for k-TSP and k-MST. For completeness of exposition, we describe this perturbation step in Sect. 4.4.

We use the randomly shifted quad-tree dissection method due to Arora to partition the Euclidean plane [3]. We briefly explain this method here (a reader familiar with this can skip this paragraph). We start with the bounding square of the input points and recursively partition it into smaller sub-squares. Suppose the length of this bounding square (box) is L and without loss of generality, assume L is a power of 2. Each square is partitioned into four equal sub-squares and we recursively proceed for each sub-square and stop partitioning a sub-square if it has unit length (and therefore has at most one input point in it). There is an immediate 4-ary tree structure corresponding to this dissection where each square in the dissection is a node of the tree. For two integers $a, b \in [0, L)$, the (a, b)-shift of dissections is obtained by shifting the x- and y- coordinates of all the dissecting lines by a and b modulo L, respectively. We obtain an (a, b)-shifted quad-tree from the 4-ary tree representing the corresponding shifted dissection by simply removing (from the tree) the partitioning of squares without any input point.

For each square in a shifted dissection, we place a portal at each of its four corners and place m evenly spaced points on each edge, where m is a power of 2 (to be defined later). In other words, we put $4(m+1)$ portals on each square and the portals of higher level squares are co-located on some portals of lower level squares. Note however that each portal is owned by a distinct square (so even if several portals are co-located at the same point each belongs to one unique square). Let a portal-respecting path between two points be a path that crosses the squares only at portals. Let S be a collection of pairs of input points in the plane and let $c(S)$ be the total Euclidean distance of pairs. Arora, Raghavan, and Rao [5] show that in presence of our assumptions, if we pick $0 \le a, b < L$ uniformly at random and use $m = O(\log n/\epsilon)$ portals on each side of each square, then with probability at least $\frac{1}{2}$, there exist some portal-respecting paths between the pairs in S with total length (cost) at most $(1+\epsilon)c(S)$. We present a dynamic programming algorithm to find a $(1+\epsilon, 1+\epsilon)$-approximate portal-respecting solution for the UCFLP instances. Then, we run this algorithm for all possible values

of a and b and return the best solution of these runs. Based on the above result regarding portal respecting paths, this solution is a $(1+\epsilon', 1+\epsilon')$-approximate solution for the Euclidean UCFLP for some ϵ' depending on ϵ.

4.1 Grouping Clients and Rounding Demand Sizes

For the simplicity of description, we assume $\frac{1}{\epsilon}$ is an integer and denote it by p. First, we apply a grouping and rounding step which is essentially the same as the one in [9]. We partition the clients into three groups of *large*, *small*, and *tiny* clients according to their demand sizes. We round down the demands of large and small clients and show that the precision error (i.e., violation of facility capacities) resulted from this rounding does not exceed 2ϵ. Also, this rounding does not increase the cost by more than a $(1+\epsilon)$ factor when we restore the original demands. As before, the clients with demands greater than ϵ are called *large* but the definition of small clients are different from before. We round down the demands of large clients to the nearest multiple of ϵ^2. This yields $q = \frac{1-\epsilon}{\epsilon^2} + 1 = p^2 - p + 1$ distinct demand values. Since each facility can serve at most p large clients, this rounding yields a blowup of at most $p\epsilon^2 = \epsilon$ of facility capacities. Therefore, if we work with these rounded down demand values from now on, we violate the facility capacities by at most an additive value of ϵ. We define a client type corresponding to each of these q distinct demand value; so far we have defined q client types.

A client j with demand $\epsilon/n < d_j \leq \epsilon$ is called *small*. We round down the demands of small clients to the nearest multiple of ϵ^2/n. Similar to large clients, this yields an additive blowup of at most ϵ of facility capacities (as there can be at most n/ϵ small clients adding a total of ϵ to the demand of a facility). We intend to assign these clients fractionally. Therefore, we just need to keep track of their total demand which is a multiple of ϵ^2/n. For this purpose, we divide each small client with rounded demand d into dn/ϵ^2 clients each with demand ϵ^2/n. Since $d \leq \epsilon$ and is a multiple of ϵ^2/n, we break each small client into at most n/ϵ clients. These clients of demand ϵ^2/n form type $q+1$.

Finally, we call all the other clients *tiny*. The demand of these clients are negligible with respect to capacities, i.e., even if we assign all of them to an already full facility, it yields an additive blowup of at most ϵ. We leave their demands intact. It should be noted that their assignment cost may still be significant and we must be careful in their assignment. Now, we are ready to define the dynamic programming tables with respect to the $q+1$ client types and tiny clients. Note that $q+1$ is a constant for a fixed $\epsilon > 0$ and this is crucial in our algorithm.

4.2 The Dynamic Programming Tables

We define a set of subproblems for each square in the shifted quad-tree and solve it by combining the solutions of its children. We treat the clients of type 1 through $q+1$ separately from tiny clients. First, we explain how we handle the clients of type 1 through $q+1$. Consider a shortest portal-respecting path from a client to a facility. It

passes through some nodes of the quad-tree (i.e., squares in the dissection): The client moves up to its least common ancestor in the quad-tree and moves downward to the node (i.e., square) containing the facility. Equivalently, we can think of moving up the facility to meet the client in their least common ancestor node in the tree and each one pays its own movement cost. Looking at the problem this way, the portion of a facility capacity serving a client must move to their least common ancestor. In fact, we are breaking each facility i into some virtual facilities, one for each client j it must serve, and move that virtual facility (corresponding to the portion of capacity for serving j) to meet j at the least common ancestor of the two squares containing i and j.

Assume we have decided (in the dynamic programming) that a facility must serve n_ℓ clients of demand type ℓ for each $1 \leq \ell \leq q+1$ (we make this decision by enumerating all possible ways of choosing n_ℓ values when we solve the base case corresponding to the leaf of quad-tree containing this facility). We break this facility into n_ℓ virtual facilities of type ℓ for all $1 \leq \ell \leq q+1$. A virtual facility of type ℓ can only serve a client of type ℓ. Considering the problem this way, in each square some clients will be served by facilities inside, some clients will be shipped outside through the portals of the square, some virtual facilities inside will serve the clients inside and some will be sent outside through the portals to serve clients from outside. Thus, for each portal of a square, we just need to keep track of how many of each demand type is sent to this portal to be served outside and how many of each virtual facility type is sent to this portal to service clients from outside. One nice feature of breaking the facilities into virtual facilities is that once we have decided how a facility is going to be broken into virtual facilities (i.e., each n_ℓ is guessed for $1 \leq \ell \leq q+1$), we can consider the subproblem of each demand type independent of other types since no client or virtual facility of a type can interfere with other types.

To handle tiny clients, we first need to point out a simple observation. Assume some tiny clients are sent to a portal of a square they belong to, to be sent to their facility. As we stated before, tiny clients can be assigned to any open facility without a significant blowup in capacities. Thus, we can send all the tiny clients sent to a portal to the nearest open facility, without increasing the connection cost. In other words, there is a solution with cost no more than optimal cost in which all the tiny clients sent to the same portal, head to the same facility (i.e., nearest open one) and we seek such a solution in our algorithm. To find such a solution, we fix (by enumerating all possibilities) the facility that should be used by tiny clients sent to each portal (of each square) in our dynamic programming. Then, instead of assigning each tiny client to a facility, we assign it to one of the portals around the square containing it and automatically all the tiny clients assigned to that portal will be served at the "guessed" nearest open facility for that portal. After choosing a portal for a tiny client, the client pays the cost of connection to the facility of that portal with a portal-respecting path fully upfront.

Formally, an instance of a subproblem is defined as a tuple (s, D, F) where s is a square in the quad-tree (i.e., the dissection) and D and F are two matrices containing information about the demands and (virtual) facilities moved to portals of s: D is a $4(m+1) \times (q+2)$ matrix (for demands) where the ith row keeps track of the information regarding the ith portal of S. The first $q+1$ elements of this row show the number of clients of each client type (from 1 to $q+1$) moved to this portal. The

$(q + 2)$th element shows the index of the facility that the tiny clients (inside of s) that are moved to this portal plan to use. F is defined similarly for facilities, namely it is a $4(m + 1) \times (q + 2)$ matrix where the ith row keeps track of the information regarding the ith portal for facilities. The first $q + 1$ elements of this row show the number of virtual facilities of each of $q + 1$ virtual facility types moved to this portal. The $(q + 2)$th element shows the index of the nearest open facility inside s to this portal (to be used for tiny clients moved to this portal). A value of 0 for facility index indicates that no tiny client can use this portal to reach an open facility. For example, when there is no open facility available inside, this value is 0. We store in $A[s, D, F]$ the value of optimal solution of the subproblem (s, D, F), i.e., the minimum total cost to service the clients in square s by opening facilities inside and sending clients and virtual facilities inside s to outside of s through its portals according to matrices D and F. The tiny clients inside s pay their connection cost fully upfront i.e., if they are to be served at a facility outside of s they contribute their connection cost to those facilities to the total cost, while the other clients (large and small) just pay the cost to move to their portal or to be served inside. The solution that we are seeking to find is $A[s_r, \mathbf{0}, \mathbf{0}]$ with s_r being the bounding box of all the input points. In the last step, we should round the "fractional" assignment of small clients to an integer one, again using the algorithm of Shmoys and Tardos [34] for the GAP. This rounding does not increase the cost but may violate the capacity constraints by the maximum demand value. Since each small client has demand at most ϵ, this will incur another additive of at most ϵ blow up in capacity constraints.

4.3 Computing the Table

We now show how we can compute $A[s, D, F]$ recursively. The leaves of the quad-tree have either one client or one facility in them. We have three cases:

1. There is one client of type j $(1 \leq j \leq q + 1)$ inside s: We must ship this client to one of the portals. For the entries $A[s, D, F]$ where F is the zero matrix and D has one non-zero value equal to 1 in the jth column of exactly one of its rows, say row ℓ, we set $A[s, D, F]$ to the cost of shipping the client to the ℓth portal. For all other entries, we set $A[s, D, F]$ to undefined (or infinity).
2. There is one tiny client inside s: For the entries $A[s, D, F]$ where F is zero and D has exactly one non-zero value, say i (corresponding to facility i), in the $(p+2)$th column of exactly one of its rows, say ℓ, we set $A[s, D, F]$ to the cost of moving the client to the ℓth portal and then moving it through the shortest portal-respecting path between the portal and facility i. For all other entries, we set $A[s, D, F]$ to undefined (or infinity).
3. There is a facility, say i, in the square. We set $A[s, \mathbf{0}, \mathbf{0}] = 0$. For the entries $A[s, D, F]$ in which D is the zero matrix, and the total capacity (i.e., virtual facilities) sent to the portals of s according to F is not more than the capacity of facility i, and the $(p+2)$th column of F has value i in its entries, we set $A[s, D, F]$ to the opening cost of i plus the cost of moving virtual facilities according to F to the portals. For all other entries, we set $A[s, D, F]$ to undefined (or infinity).

We update each non-leaf node as follows. Let s be a non-leaf node and s_1, s_2, s_3, and s_4 be its four children. We enumerate all the combinations of matrices D and F for s and matrices D_i and F_i for s_i for all $1 \leq i \leq 4$ where the subproblems are consistent (to be explained below). We update $A[s, D, F]$ with respect to subproblems (s_i, D_i, F_i). As mentioned before, we can solve the problem for each demand type and the corresponding virtual facility type separately, because there is no dependencies between two different types. Let Ψ_ℓ be the extra cost for demand type ℓ in sub-problem (s, D, F) in addition to the cost we paid in the subproblems (s_i, D_i, F_i) $(1 \leq i \leq 4)$. We show how to compute Ψ_ℓ below.

Let $m^{(\ell)}$ and $m_i^{(\ell)}$ (for $1 \leq i \leq 4$) be the number of virtual facilities of type ℓ in F and F_i, respectively. We define $n^{(\ell)}$ and $n_i^{(\ell)}$ (for $1 \leq i \leq 4$) for the number of clients of type ℓ in D and D_i, respectively. Note that $n_1^{(\ell)} + n_2^{(\ell)} + n_3^{(\ell)} + n_4^{(\ell)} - n^{(\ell)}$ is the total number of clients of type ℓ inside s that are to be serviced inside s since only $n^{(\ell)}$ many demands of type ℓ are shipped outside from a total of $n_1^{(\ell)} + n_2^{(\ell)} + n_3^{(\ell)} + n_4^{(\ell)}$. Similarly, of a total of $m_1^{(\ell)} + m_2^{(\ell)} + m_3^{(\ell)} + m_4^{(\ell)}$ virtual facility of type ℓ inside s, only $m^{(\ell)}$ is sent to portals and therefore $m_1^{(\ell)} + m_2^{(\ell)} + m_3^{(\ell)} + m_4^{(\ell)} - m^{(\ell)}$ virtual facilities of type ℓ must be used inside s. Therefore, we must have

$$n_1^{(\ell)} + n_2^{(\ell)} + n_3^{(\ell)} + n_4^{(\ell)} - n^{(\ell)} = m_1^{(\ell)} + m_2^{(\ell)} + m_3^{(\ell)} + m_4^{(\ell)} - m^{(\ell)}, \tag{9}$$

or else the subproblems considered are inconsistent. Denote this quantity by $r^{(\ell)}$; so we must assign exactly $r^{(\ell)}$ clients of type ℓ to $r^{(\ell)}$ facilities of this type inside s; otherwise this combination of sub-problems is impossible and the solutions to the sub-problems are inconsistent. We can assign these $r^{(\ell)}$ clients and facilities of type ℓ optimally in polynomial time by running a minimum cost perfect matching algorithm as described below. Note that by Eq. (9), we must have $n_1^{(\ell)} + n_2^{(\ell)} + n_3^{(\ell)} + n_4^{(\ell)} + m^{(\ell)} = m_1^{(\ell)} + m_2^{(\ell)} + m_3^{(\ell)} + m_4^{(\ell)} + n^{(\ell)}$. Since out of $n_1^{(\ell)} + n_2^{(\ell)} + n_3^{(\ell)} + n_4^{(\ell)}$ clients of type ℓ, $n^{(\ell)}$ are shipped outside, we must match each of $n_1^{(\ell)} + n_2^{(\ell)} + n_3^{(\ell)} + n_4^{(\ell)}$ to one of $n^{(\ell)}$ or one of $m_1^{(\ell)} + m_2^{(\ell)} + m_3^{(\ell)} + m_4^{(\ell)}$ virtual facilities of type ℓ. Similarly, out of $m_1^{(\ell)} + m_2^{(\ell)} + m_3^{(\ell)} + m_4^{(\ell)}$ facilities of type ℓ, only $m^{(\ell)}$ are sent to portals of s, thus each is either matched to one of $m^{(\ell)}$ facilities or is used to serve one of $n_1^{(\ell)} + n_2^{(\ell)} + n_3^{(\ell)} + n_4^{(\ell)}$ clients. This suggests to form a bipartite graph H with $n_1^{(\ell)} + n_2^{(\ell)} + n_3^{(\ell)} + n_4^{(\ell)}$ clients of the sub-squares $s_1, \ldots, s_4$ and $m^{(\ell)}$ facilities of s on one side and $m_1^{(\ell)} + m_2^{(\ell)} + m_3^{(\ell)} + m_4^{(\ell)}$ facilities of the sub-squares and $n^{(\ell)}$ clients of s on the other side. Put an edge between two vertices of the two partitions of H unless one of them is from the $m^{(\ell)}$ facilities and the other is from the $n^{(\ell)}$ clients. In other words, H has $(n_1^{(\ell)} + n_2^{(\ell)} + n_3^{(\ell)} + n_4^{(\ell)} + m^{(\ell)})(m_1^{(\ell)} + m_2^{(\ell)} + m_3^{(\ell)} + m_4^{(\ell)} + n^{(\ell)}) - m^{(\ell)} n^{(\ell)}$ edges. The cost of an edge of H is the cost of the shortest portal-respecting path between its endpoints times d_ℓ (demand of type ℓ). It is not hard to see that this gives the best possible assignment for the demand type ℓ given the sub-problems, i.e., Ψ_ℓ is the weight of the minimum cost perfect matching of H. We update $A[s, D, F]$ to:

$$\min \Bigg(A[s, D, F], A[s_1, D_1, F_1] + A[s_2, D_2, F_2] + A[s_3, D_3, F_3] + A[s_4, D_4, F_4] + \sum_{\ell=1}^{p+1} \Psi_\ell \Bigg).$$

If for one of the types $1 \leq \ell \leq p+1$, one of the sub-problems does not have a solution, then the combinations of matrices that we have guessed are inconsistent (and we do not update $A[s, D, F]$).

One other consistency condition of the matrices D, F, and D_i, F_i (for $1 \leq i \leq 4$) that we need to check is that for co-located portal points of s, s_1, s_2, s_3, and s_4, we must have consistency of facilities devoted to tiny clients in those portal points. For instance, if t_1 is a portal of s_1 and t is a portal point of s and the tiny clients of s_1 shipped to t_1 are to be served at facility i_1 and the tiny clients of s shipped to t are to be served at i and the shortest portal-respecting path from t_1 to i_1 goes through t then we must have $i_1 = i$, or else the sub-problems are not consistent. Here is how we check for this type of consistency. Let T be the set of all portals in s, s_1, s_2, s_3, s_4. Recall that each portal point is "owned" by a square, so there might be several portal points in T that are co-located. For each portal $t \in T$ (where t is the portal of exactly one of s, s_1, s_2, s_3, or s_4) let $f(t)$ be the index of the facility that the tiny clients sent to t are assigned to; this index can be found in row t and column $(q + 2)$ of D or D_1, D_2, D_3, or D_4 (whichever sub-problem the portal t belongs to). For each portal $t \in T$, let $f'(t)$ be the index of the nearest facility to this portal inside the square of this portal, which can be found in column $(q + 2)$ of F or F_1, F_2, F_3, or F_4 (again depending on which square owns portal t). For each portal $t \in T$, if $f(t)$ is non-zero, we check a shortest portal-respecting path from t to $f(t)$ and consider two cases:

> Case 1: if $f(t)$ is outside s then let $t_s \in T$ be the the last portal of T on this shortest path from t to $f(t)$ (note that t_s must be a portal of s). Then we must have that $f(t_s) = f(t)$.
> Case 2: if $f(t)$ is inside s and belongs to one of the 4 sub-squares of it, say square s_i, then let t_i be the last portal of T on this shortest path from t to $f(t)$ (note that t_i must belong to s_i). Then we must have that $f(t_i) = f(t)$.

If either of these conditions are not satisfied we have an inconsistency of facilities of tiny clients and we skip the guessed matrices for the sub-problems.

4.4 Preprocessing Step

In this subsection we describe a perturbation step which enforces the assumptions we made about the size of the bounding box containing the points. Recall that we want: all the points are on a unit grid, the minimum inter-node distance is at least 4, and the maximum inter-node distance is at most $O(n^4)$. Observe that if we scale the connection and opening costs, then the optimum solution of the problem remains the same. Thus,

we scale the costs by $4/c_{min}$, where c_{min} is the minimum non-zero inter-node distance. In the new instance with scaled costs, c_{min} is 4. From now on, we work with this instance. Let OPT be the value of an optimal solution. We first compute a crude approximation, A, of OPT. For instance, this can be done by using the $O(\log(n), 1+\epsilon)$-approximation algorithm of [9]. Therefore, we have $A = O(\log n) \cdot OPT$. We pick random a and b and construct a shifted quad-tree. This time, we stop as soon as the size of squares become less than nA. We claim that the shifted quad-tree does not cut any edge of an optimal solution with probability at least $1 - 4/n$. Hence, we can treat each leaf of this quad-tree as an independent instance where the length of the bounding box of each instance is at most nA.

To prove the claim, consider the case that the optimal solution consists of a collection of line segments with length at least 4. By Lemma 4 of [4], each line segment of the optimal solution with length s crosses at most $2s$ lines of unit grid. Therefore, at most $2OPT \leq 2A$ lines of gird are crossed by any segment in the solution. As a result, the probability that the shifted quad-tree cuts any of these segments is at most $2A$ over the length of the leaf squares. Since the length of a leaf square is at least $nA/2$, the probability is at most $4/n$.

Let L be the size of the bounding box of the points. We overlay a grid of granularity $\epsilon A/(4n^2 \log(n))$. We move the input nodes one by one to its nearest grid point whose neighbourhood of gird-distance 4 is empty of any other node. It is not hard to see that each original node can be assigned to a grid point of grid-distance at most $4n$. Therefore, we move each point by at most $4n\epsilon A/(4n^2 \log(n))$, and by the triangle equality, the cost of any solution increases by at most $n\epsilon A/(n \log(n)) \leq \epsilon OPT$. Since $L < nA$, the size of the bounding box after scaling is at most $2(4n) + L/(\epsilon A/(4n^2 \log(n))) < 8n + 4\frac{1}{\epsilon}n^3 \log(n) = O(n^4)$.

4.5 Wrap-up

Proof of Theorem 4 Recall that in the preprocessing step, the cost of the solution increases by a factor of at most $(1 + \epsilon)$. When we restrict our solution to be portal-respecting, we lose another $1 + \epsilon$ factor. In the dynamic programming, we find the best solution under these restrictions with respect to rounded demand values. When we restore the demands to their original values, we increase them by a factor of at most $1 + \epsilon$, which increases the cost by at most another $1 + \epsilon$ factor. As a result, we find a solution whose cost is at most $(1 + \epsilon)^3$ factor away from the optimal value. As we stated before, the total blow-up in facility capacities incurred due to rounding of large and small clients is at most 2ϵ. In addition, the assignment of tiny clients may result in another ϵ blowup of capacity for each facility. Finally, when we round the fractional assignment of small clients using the Shmoys-Tardos algorithm, we may have another blowup of at most ϵ. Therefore, in total, we may have a violation of capacities by at most 4ϵ. This shows that the bicriteria factor of our algorithm is $((1 + \epsilon)^3, 1 + 4\epsilon)$. By setting $\epsilon < \epsilon'/4$, we have the claimed performance, because $(1 + \epsilon'/4)^3 \leq 1 + \epsilon'$ for $0 < \epsilon' \leq 1$.

Now, we analyse the time complexity. The preprocessing step and construction of the shifted quad-tree can be done in time $O(n \log n)$ [4]. Let T be the number of

table entries for each square s. It can be seen that $T = O(n^{8(m+1)(q+2)})$. When s corresponds to a leaf of the quad-tree all the entries of the table can be computed in constant time. We compute the table entries corresponding to each non-leaf square s in time $O(T^5)$ which is to enumerate all combinations of possible entries for s and its four children. Therefore, for any fixed $\epsilon > 0$ the overall running time of the dynamic programming algorithm is in $n^{O(m)} = n^{O(\log n/\epsilon)}$. □

Remark 4 We presented our algorithm and the proof for $\mathbb{R}^2$. One can generalize these to d dimensions where the performance ratio remains the same and the running time increases to $n^{O((\log n/\epsilon)^{d-1})}$ which is still quasi-polynomial time for constant $d > 0$.

5 Discussion

We presented a reduction from the UCFLP to a restricted version in which all demand values are large (i.e., larger than ϵ) and presented two algorithms for the case of $\epsilon = \frac{1}{2}$ and $\frac{1}{3}$. These implied two constant factor approximation algorithms for the UCFLP with capacity bounds within factor $3/2$ and $4/3$. We suspect similar results can be found with capacity violations bounded within factor $1 + \epsilon$ for any $\epsilon > 0$. We also showed that at a loss of factor $1/\epsilon$, we can ignore the opening cost of facilities, and that if there is an $(\alpha(\epsilon), 1 + \epsilon)$-approximation for these instances then there is an $(\alpha'(\epsilon), 1 + \epsilon)$-approximation for the general case. We guess that it should be possible to design constant factor (perhaps depending on ϵ) approximation for RCFLP(ϵ) with a violation of at most $1 + \epsilon$ on capacities. The elimination of small clients (and facility costs) makes the problem a generalization of the bin packing, where there is an associated cost function for assigning each item (client j) to each bin (facility i) and it seems one might be able to use the scaling/grouping techniques used for bin-packing here. For instance, with a simple grouping technique and rounding down each demand value to its closest power of $1 + \epsilon$, we can reduce the number of distinct demand types to a function of ϵ. Using this, one can guess (enumerate) the number of different facility "types" similar to what Fernandez de la Vega and Lueker [16] did for the bin packing problem. Here, a facility type shows how many of each client type must be present in a facility. However, it is not clear what the next step should be. Perhaps a suitable configuration LP at this step could be useful.

The reader might wonder if local search algorithms analogous to the ones used in the splittable CFLP might work here. The main issue in the splittable CFLP is to decide the subset of the facilities that should be opened. As stated above, as a consequence of our reduction, we can assume there is no opening costs and we can open all the facilities. Thus, the local search operations defined in the splittable CFLP are ineffective here as the main difficulty is how to assign the clients to open facilities. Another possibility is that after guessing the facility types as discussed above, we run a local search to assign these facility types to the facilities. For each assignment of the facility types to the facilities, we can run a min-cost max-flow algorithm for each type to decide where each client should go. Unfortunately, the most natural local search operations we tried (for example swapping facility types assigned to two facilities) have bad locality gaps.

For the case of Euclidean metrics, one might ask if the stronger structural theorem of Kolliopoulos and Rao [24] for the standard facility location problem which only needs $O(1/\epsilon)$ portals instead of $O(\log n/\epsilon)$ portals for each square could be used to improve the running time of our algorithm from quasi-polynomial to a true polynomial. The difficulty is that Kolliopoulos and Rao critically use the fact that we can assign a client to any open facility in the solution of the UFLP in the proof of their structural theorem, while in our case, this is not true. It is an interesting question whether it is possible to find a similar structural theorem for the UCFLP with $O(1/\epsilon)$ portal points for each square; that would imply a PTAS for the Euclidean UCFLP.

References

1. Aggarwal, A., Anand, L., Bansal, M., Garg, N., Gupta, N., Gupta, S., Jain, S.: A 3-approximation for facility location with uniform capacities. In: Integer Programming and Combinatorial Optimization, vol 6080 of Lecture Notes in Computer Science, pp. 149–162. Springer, Berlin (2010)
2. Alzoubi, H.A., Lee, S., Rabinovich, M., Spatscheck, O., der Merwe, J.: Anycast CDNS revisited. In: WWW '08: Proceeding of the 17th International Conference on World Wide Web, pp. 277–286. ACM, New York, NY (2008)
3. Arora, S.: Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In: Proceedings of the 37th Annual Symposium on Foundations of Computer Science, pp. 2–12. IEEE Computer Society, Washington, DC (1996)
4. Arora, S.: Nearly linear time approximation schemes for Euclidean TSP and other geometric problems. In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science, pp. 554–563. IEEE Computer Society, Washington, DC (1997)
5. Arora, S., Raghavan, P., Rao, S.: Approximation schemes for Euclidean k-medians and related problems. In: Proceedings of the Thirteeth Annual ACM Symposium on Theory of Computing, STOC '98, pp. 106–113. ACM, New York, NY (1998)
6. Arya, V., Garg, N., K., Rohit, M., Adam, M., Kamesh, P., Vinayaka: Local search heuristic for k-median and facility location problems. In: STOC '01: Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing, pp. 21–29. ACM, New York, NY (2001)
7. Bansal, M., Garg, N., Gupta, N.: A 5-approximation for capacitated facility location. In: 20th European Symposium on Algorithms, pp. 133–144 (2012)
8. Bartal, Y.: On approximating arbitrary metrices by tree metrics. In: STOC '98: Proceedings of the Thirteeth Annual ACM symposium on Theory of computing, pp. 161–168. ACM, New York, NY (1998)
9. Bateni, M.H., Hajiaghayi, M.T.: Assignment problem in content distribution networks: unsplittable hard-capacitated facility location. In: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 805–814 (2009)
10. Behsaz, B.: Approximation Algorithms for Clustering Problems. PhD thesis, University of Alberta, Department of Computing Science (2012)
11. Byrka, J.: An Optimal Bifactor Approximation Algorithm for the Metric Uncapacitated Facility Location Problem. In: APPROX '07/RANDOM '07: Proceedings of the 10th International Workshop on Approximation and the 11th International Workshop on Randomization, and Combinatorial Optimization. Algorithms and Techniques, pp. 29–43. Springer, Berlin (2007)
12. Charikar, M., Guha, S.: Improved combinatorial algorithms for the facility location and k-median problems. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, FOCS '99, pp. 378–388. IEEE Computer Society, Washington, DC (1999)
13. Chudak, F.: Improved approximation algorithms for uncapacitated facility location. In: Integer Programming and Combinatorial Optimization, volume 1412 of Lecture Notes in Computer Science, pp. 180–194. Springer, Berlin (1998)
14. Chudak, F.A., Shmoys, D.B.: Improved approximation algorithms for a capacitated facility location problem. In: SODA '99: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 875–876. Society for Industrial and Applied Mathematics, Philadelphia, PA (1999)

15. Chudak, F.A., Williamson, D.P.: Improved approximation algorithms for capacitated facility location problems. In: Proceedings of the 7th International IPCO Conference on Integer Programming and Combinatorial Optimization, pp. 99–113. Springer, London (1999)
16. de la Vega, W., Lueker, G.S.: Bin packing can be solved within $1 + \epsilon$ in linear time. Combinatorica **1**(4), 349–355 (1981)
17. Drezner, Zvi, Hamacher, Horst W.: Facility Location: Applications and Theory. Springer, Berlin (2004)
18. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. In: STOC '03: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, pp. 448–455. ACM, New York, NY (2003)
19. Garey, Michael R., Johnson, David S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Co., New York, NY (1979)
20. Guha, S., Khuller, S.: Greedy strikes back: improved facility location algorithms. In: SODA '98: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 649–657. Society for Industrial and Applied Mathematics, Philadelphia, PA (1998)
21. Jain, K., Mahdian, M., Saberi, A.: A new greedy approach for facility location problems. In: STOC '02: Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing, pp. 731–740. ACM, New York, NY (2002)
22. Jain, K., Vazirani, V.V.: Primal-dual approximation algorithms for metric facility location and k-median problems. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, FOCS '99, pp. 2–13. IEEE Computer Society, Washington, DC (1999)
23. Karmarkar, N., Karp, R.M.: An efficient approximation scheme for the one-dimensional bin-packing problem. In: IEEE Symposium on Foundations of Computer Science, pp. 312–320 (1982)
24. Kolliopoulos, S.G., Rao, S.: A nearly linear-time approximation scheme for the Euclidean kappa-median problem. In: Proceedings of the 7th Annual European Symposium on Algorithms, ESA '99, pp. 378–389. Springer, London (1999)
25. Korupolu, M.R., Plaxton, C.G., Rajaraman, R.: Analysis of a local search heuristic for facility location problems. In: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete algorithms, SODA '98, pp. 1–10. Society for Industrial and Applied Mathematics , Philadelphia, PA (1998)
26. Levi, R., Shmoys, D., Swamy, C.: LP-based Approximation Algorithms for Capacitated Facility Location. In: Bienstock, D., Nemhauser, G. (eds.) Integer Programming and Combinatorial Optimization. Lecture Notes in Computer Science, vol. 3064, pp. 21–27. Springer, Berlin (2004)
27. Li, S.: A 1.488 approximation algorithm for the uncapacitated facility location problem. In: Proceedings of the 38th International Conference on Automata, Languages and Programming, Vol. Part II, ICALP'11, pp. 77–88. Springer, Berlin (2011)
28. Lin, J.-H., Vitter, J.S.: e-approximations with minimum packing constraint violation (extended abstract). In: Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing, STOC '92, pp. 771–782. ACM, New York, NY (1992)
29. Love, Robert F., Morris, James G., Wesolowsky, George O.: Facilities Location: Models and Methods. North-Holland, Amsterdam (1988)
30. Mahdian, M., Pál, M.: Universal facility location. In: Di Battista, G., Zwick, U. (eds.) Algorithms—ESA 2003. Lecture Notes in Computer Science, vol. 2832, pp. 409–421. Springer, Berlin (2003)
31. Mahdian, M., Ye, Y., Zhang, J.: A 2-approximation algorithm for the soft-capacitated facility location problem. In: RANDOM-APPROX, pp. 129–140 (2003)
32. Mahdian, M., Ye, Y., Zhang, J.: Improved approximation algorithms for metric facility location problems. In: APPROX, pp. 229–242 (2002)
33. Pál, M., Tardos, É., Wexler, T: Facility location with nonuniform hard capacities. In: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, FOCS '01, pp. 329–338. IEEE Computer Society, Washington, DC (2001)
34. Shmoys, D.B., Tardos, E.: An approximation algorithm for the generalized assignment problem. Math. Progr. **62**(3), 461–474 (1993)
35. Shmoys, D.B., Tardos, E., Aardal, K.: Approximation algorithms for facility location problems. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, pp. 265–274 (1997)
36. Verter, V.: Foundations of location analysis, chapter 2. International Series in Operations Research and Management Science. Springer, Berlin (2011)
37. Zhang, J., Chen, B., Ye, Y.: A Multi-exchange local search algorithm for the capacitated facility location problem. In: Integer Programming and Combinatorial Optimization, pp. 1–4. Springer, Berlin (2004)