Approximation Algorithms for Minimum-Load k-Facility Location*

Sara Ahmadian† Babak Behsaz‡ Zachary Friggstad§ Amin Jorati¶
Mohammad R. Salavatipour‖ Chaitanya Swamy†

Abstract

We consider a facility-location problem that abstracts settings where the cost of serving the clients assigned to a facility is incurred by the facility. Formally, we consider the *minimum-load k-facility location* (MLkFL) problem, which is defined as follows. We have a set $\mathcal{F}$ of facilities, a set $\mathcal{C}$ of clients, and an integer $k \geq 0$. Assigning client j to a facility f incurs a connection cost $d(f,j)$. The goal is to open a set $F \subseteq \mathcal{F}$ of k facilities, and assign each client j to a facility $f(j) \in F$ so as to minimize $\max_{f \in F} \sum_{j \in \mathcal{C}: f(j)=f} d(f,j)$; we call $\sum_{j \in \mathcal{C}: f(j)=f} d(f,j)$ the *load* of facility f. This problem was studied under the name of min-max star cover in [6, 2], who (among other results) gave bicriteria approximation algorithms for MLkFL for when $\mathcal{F} = \mathcal{C}$. ML$k$FL is rather poorly understood, and only an $O(k)$-approximation is currently known for MLkFL, *even for line metrics*.

Our main result is the *first polytime approximation scheme* (PTAS) for MLkFL on line metrics (note that no non-trivial true approximation of any kind was known for this metric). Complementing this, we prove that MLkFL is strongly *NP*-hard on line metrics. We also devise a quasi-PTAS for MLkFL on tree metrics. MLkFL turns out to be surprisingly challenging even on line metrics, and resilient to attack by a variety of techniques that have been successfully applied to facility-location problems. For instance, we show that: (a) even a configuration-style LP-relaxation has a bad integrality gap; and (b) a multi-swap k-median style local-search heuristic has a bad locality gap. Thus, we need to devise various novel techniques to attack MLkFL.

Our PTAS for line metrics consists of two main ingredients. First, we prove that there always exists a near-optimal solution possessing some nice structural properties. A novel aspect of this proof is that we first move to a mixed-integer LP (MILP) encoding of the problem, and argue that a MILP-solution minimizing a certain potential function possesses the desired structure, and then use a rounding algorithm for the generalized-assignment problem to "transfer" this structure to the rounded integer solution. Complementing this, we show that these structural properties enable one to find such a structured solution via dynamic programming.

---

*A preliminary version of this paper [**?**] appeared in the Proceedings of APPROX 2014.

†Dept. of Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1. Supported in part by the last author's NSERC grant 327620-09, an NSERC Discovery Accelerator Supplement Award, and an Ontario Early Researcher Award. Email: {sahmadian,cswamy}@uwaterloo.ca.

‡Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. Email: behsaz@ualberta.ca. Supported in part by NSERC.

§Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. Email: zacharyf@cs.ualberta.ca.

¶Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. Email: jorati@ualberta.ca.

‖Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. Email: mrs@ualberta.ca. Supported by NSERC.

1 Introduction

Facility-location (FL) problems have been widely studied in the Operations Research and Computer Science communities (see, e.g., [15] and the survey [18]), and have a wide range of applications. These problems are typically described in terms of an underlying set of clients that require service, and a candidate set of facilities that provide service to these clients. The goal is to determine which facilities to open, and decide how to assign clients to open facilities to minimize some combination of the facility-opening and client-connection (a.k.a service) costs. An oft-cited prototypical example is that of a company wanting to decide where to locate its warehouses/distribution centers so as to serve its customers in a cost-effective manner.

We consider settings where the cost of serving the clients assigned to a facility is incurred by the facility; for instance, in the above example, each warehouse may be responsible for supplying its clients and consequently bears a cost equal to the total cost of servicing its clients. In such settings, it is natural to consider the problem of minimizing the maximum cost borne by any facility. Formalizing this, we consider the following mathematical model: we have a set $\mathcal{F}$ of facilities and a set $\mathcal{C}$ of n clients. Assigning client j to a facility f incurs a *connection cost* or *service cost* $d(f,j)$. There are no facility-opening costs. The goal is to open k facilities from $\mathcal{F}$ and assign each client j to an open facility $f(j)$ so as to minimize the maximum *load* of an open facility, where the load of an open facility f is defined to be $\sum_{j\in\mathcal{C}:f(j)=f} d(f,j)$; that is, the load of f is the total connection cost incurred in serving the clients assigned to it. We call this the *minimum-load k-facility location* (MLkFL) problem. As is common in the study of facility-location problems, we assume that the clients and facilities lie in a common metric space, so the d is a metric on $\mathcal{F}\cup\mathcal{C}$.

Despite the extensive amount of literature on facility-location problems, there is surprisingly little amount of work on MLkFL and it remains a rather poorly understood problem (see [17]). One can infer that the problem is *NP*-hard, even when the set of open facilities is fixed, via a reduction from the makespan-minimization problem on parallel machines, and that an $O(k)$-approximation can be obtained by running any of the various $O(1)$-approximation algorithms for k-*median* [4, 11, 10, 3, 14] (where one seeks to minimize the *sum* of the facility loads). No better approximation algorithms are known for MLkFL even on line metrics, and this was mentioned as an open problem in [17]. The only works on approximation algorithms for this problem that we are aware of are due to Even et al. [6] and Arkin et al. [2], both of which refer to this problem as *min-max star cover* (where $\mathcal{F}=\mathcal{C}$).[1] Both works obtain *bicriteria* approximation algorithms for MLkFL in general metrics, which means that the algorithm returns a solution with near-optimal maximum load but may need to open more than k facilities.[2] For MLkFL on star metrics and when $\mathcal{F}=\mathcal{C}$, some $O(1)$-approximation algorithms follow from work on minimum-makespan scheduling and [6, 2] (see "Related work").

Our results. We resolve the status of min-load k-FL on line metrics. As we elaborate below (see "Our techniques"), MLkFL turns out to be surprisingly challenging even on line metrics, and seems resilient to attack by a variety of techniques that have been successfully applied to facility-location problems, including LP-rounding and primal-dual methods. Our main result is that despite these difficulties, one can devise a polynomial-time approximation scheme (PTAS) for MLkFL on line metrics (Theorem 3.1). As mentioned earlier, this is the *first* approximation algorithm for MLkFL on line metrics that achieves anything better than an $O(k)$-approximation.

We also consider MLkFL in tree metrics (Section 4). First, we observe that the quasi-PTAS obtained by

[1] Jorati [12], in his Master's thesis, obtained a preliminary version of some of our current results.

[2] When $\mathcal{F}=\mathcal{C}$, it is easy to obtain a bicriteria guarantee for MLkFL using an α-approximation algorithm $\mathcal{A}$ for k-median, as follows. Letting L^{opt} denote the optimal cost, by running $\mathcal{A}$ we obtain k facilities with total load at most $k\alpha L^{opt}$. Fix $\epsilon\geq 0$. For each facility i with load $L_i\geq\alpha(1+\epsilon)L^{opt}$, we can partition its clients into at most $\left\lceil\frac{2L_i}{\alpha(1+\epsilon)L^{opt}}\right\rceil$ groups, each creating a load of at most $\alpha(1+\epsilon)L^{opt}$ at i; we can open a facility at the location in a group closest to i to handle each group incurring a load of at most $2\alpha(1+\epsilon)L^{opt}$. This creates at most $k\left(1+\frac{2}{1+\epsilon}\right)$ facilities, each having load at most $2\alpha(1+\epsilon)L^{opt}$.

Jorati [12] for line metrics extends to yield a quasi-PTAS (QPTAS) for tree metrics (Theorem 4.3). Next, we consider the special case of star metrics, but in the more-general setting where clients may have non-uniform demands $\{D_j\}_{j\in\mathcal{C}}$ and the demand of a client may be split across several open facilities. We now define the load of a facility f to be $\sum_j x_{fj}d(f,j)$, where $x_{fj} \in \mathbb{R}_{\geq 0}$ is the amount of j's demand that is assigned to f. We devise a 12-approximation algorithm for MLkFL on star metrics with non-uniform demands (Theorem 5.1). Notice that when we restrict the metric to be a star metric, we cannot create colocated copies of a client (without destroying the star topology); thus, even the setting of non-uniform integer demands is strictly more general than the unit-demand setting.[3]

In Section 6, we obtain various computational-complexity and integrality-gap lower bounds for MLkFL. Complementing our PTAS, we show (Theorem 6.1) that MLkFL is *strongly* NP-*hard* on line metrics (and hence, a PTAS is the best approximation that one can hope to achieve in polytime unless *P =NP*). Finally, we justify our comment about the difficulty of tackling MLkFL via the various LP-based methods developed for facility-location problems by showing that even a configuration-style LP-relaxation for MLkFL—where we "guess" the optimum value B and have a variable $x_{f,S}$ for every facility f and every possible set S of clients such that $\sum_{j\in S} d(f,j) \leq B$—has an integrality gap of $\Omega(k/\log k)$ even for line metrics (Theorem 6.2). Note that the configuration LP is stronger than the natural LP-relaxation for MLkFL. Moreover, this holds even if the graph consisting of the edges (j,f) such that $d(j,f) \leq B$—call these feasible edges—is connected. This is in contrast with capacitated k-center [5, 1], where a large integrality gap for the natural LP arises due to the fact that the graph of feasible edges is disconnected.

Our techniques. Before detailing the techniques underlying our PTAS for line metrics, we describe some of the difficulties encountered in applying the machinery developed for (other) facility-location problems to MLkFL (even on line metrics). One prominent source of techniques for facility location are LP-based methods. However, our integrality-gap lower bound for line metrics points to the difficulty in leveraging such LP-based insights. In fact, we do not know of any LP-relaxation for MLkFL with a constant integrality gap even on line metrics. An approach that often comes to the rescue for FL problems when there is no known good LP-relaxation (e.g., capacitated FL) is local search, however the min-max nature of MLkFL makes it difficult to exploit this. In particular, one can come up with simple examples where a k-median style multi-swap local-search algorithm does not yield any bounded approximation ratio even for line metrics; see Section 7. (This is also true for the (uncapacitated) k-center problem, which is another min-max problem, and we are not aware of any local-search-based algorithms for k-center.)

It appears the standard methods that have been successfully applied to solve some classical clustering problems (such as variants of facility location, k-median, etc) do not work for MLkFL and one needs to find new venues of attack for min-load k-FL. Our PTAS for line metrics consists of two main ingredients. First, we prove that there always exists a near-optimal solution possessing some nice structural properties (Section 3.1). Second, we show in Section 3.2 that these structural properties enable one to find such a structured solution via dynamic programming (DP).

We now give a high-level overview of these two ingredients. First, we show that, for any $\epsilon > 0$, there is a $(1 + O(\epsilon))$-approximate solution, where the intervals corresponding to the client-facility assignments with "small" connection costs, which we call small arms, form a *laminar family*. It appears difficult to argue this directly. However, a key insight and a notable aspect of our proof, is that one can derive this structure by *moving to a mixed-integer LP* (MILP), arguing that an MILP solution satisfying this laminarity property must exist, and then utilizing a suitable rounding algorithm to "transfer" the laminarity property from the MILP solution to the rounded integral solution. More precisely, we show using uncrossing arguments that an

[3] In contrast, for line- and tree- metrics, integer (and hence, rational) demands can be handled by creating colocated copies of a client. Although, done naively, this creates a pseudpolynomial blow-up in the input size, this reduction can often be simulated without explicitly incurring this blowup. This is true of our PTAS and QPTAS for line- and tree- metrics; we therefore focus on unit demands in these settings for (mostly notational) simplicity.

MILP-solution minimizing a certain potential function (which depends on the optimal solution) must satisfy the above laminarity property. Further, we observe that the fractional assignment of clients to integrally-open facilities represented by the MILP solution can be converted to an integral one using the rounding algorithm of [19] for the *generalized assignment problem* (GAP), and this rounding procedure preserves the laminarity property.

Our next step is to exploit the above structure to efficiently find a solution satisfying these structural properties via DP. To gain some intuition, note that if *all* arms form a laminar family, then one could use DP to identify tuples (I, c, r), where I is a maximal interval such that all clients in I are either served by facilities opened from I or from c (which is a center outside I), and the load imposed by clients from I that are assigned to c is at most r. However, our setting is somewhat more complicated since only the small arms form a laminar family. Our definition of small arms however ensures that the number of "big", i.e., not small, arms incident to a facility is at most a constant (depending on ϵ). Exploiting this, we show that, despite this complication, a DP scheme is still possible if we maintain some extra information in the DP table corresponding to the big arms that "cross" each interval (see Section 3.2). This yields the desired PTAS.

Related work. There is a wealth of literature on facility-location problems (see, e.g., [15, 18]); we limit ourselves to the work that is relevant to MLkFL. As mentioned earlier, Even et al. [6] and Arkin et al. [2] are the only two previous works that study MLkFL (under the name min-max star cover). They view the problem as one where we seek to cover the nodes of a graph by stars (hence the name min-max star cover), and obtain bicriteria guarantees. Viewed from this perspective, MLkFL falls into the class of problems where we seek to cover nodes of an underlying graph using certain combinatorial objects. Even et al. and Arkin et al. consider various other min-max problems—where the number of covering objects is fixed and we seek to minimize the maximum cost of an object—in this genre. Both works devise a 4-approximation algorithm when the covering objects are trees (see also [16]), and Even et al. obtain the same approximation for the rooted problem where the roots of the trees are fixed. Arkin et al. obtain an $O(1)$-approximation when the covering objects are paths or walks. The approximation guarantees for min-max tree cover were improved by Khani and Salavatipour [13]. All of these works also consider the version of the problem where we fix the maximum cost of a covering object and seek to minimize the number of covering objects used. Frederickson et al. [7] obtain an $(e + 1)$-approximation when the covering objects are tours rooted at a given node.

For MLkFL on star metrics, when $\mathcal{F} = \mathcal{C}$, certain results follow from some known results and the above min-max results. For example, it is not hard to show that MLkFL, even with non-unit demands, can be reduced to the makespan-minimization problem on parallel machines while losing a factor of 2.[4] Since the latter problem admits a PTAS [9], this yields a $(2 + \epsilon)$-approximation algorithm for MLkFL on star metrics when $\mathcal{F} = \mathcal{C}$. When $\mathcal{F} = \mathcal{C}$ and with unit demands, one can also infer that (for star metrics) the objective value of any solution for min-max tree cover (viewed in terms of the node-sets of the trees) is within a constant factor of its objective value for min-max star cover. (This is simply because for any set S of nodes, the cost of the *best* star spanning S is at most twice the cost of the minimum spanning tree for S.) These correspondences however break down when $\mathcal{F} \neq \mathcal{C}$, even for unit demands. Our 12-approximation algorithm for star metrics works for arbitrary $\mathcal{F}, \mathcal{C}$ sets *and* non-unit (equivalently, non-uniform) demands.

As with the k-median and k-center problems, MLkFL can also be motivated and viewed as a clustering problem: we seek to cluster points in a metric space around k centers in a way that minimizes the maximum load (or "star cost") of a cluster. Whereas MLkFL and k-center are min-max clustering problems, where the quality is measured by the *maximum* cost (under some metric) of a cluster, k-median is a min-sum clustering problem, where the clustering quality is measured by summing the cost of each cluster.

[4] If we require that all k facilities lie at the root r of the star, then the resulting problem is precisely a makespan-minimization problem on k parallel machines. Given a partition $C_1, \ldots, C_k$ of the client-set obtained by solving this problem, we can simply open, for each C_i, a facility at the node in C_i that is closest to r. This increases the maximum load by a factor of at most 2.

Finally, observe that if we fix the set of k open facilities, then the problem of determining the client assignments is a special case of GAP. There is a well-known 2-approximation algorithm for GAP [19]. As noted earlier, this algorithm plays a role in the *analysis* of our PTAS for line metrics (but not the algorithm itself), when we reason about the existence of well-structured near-optimal solutions.

2 Problem definition

In the minimum-load k-facility location (MLkFL) problem, we are given a set of clients $\mathcal{C}$ and a set of facilities $\mathcal{F}$ in a given metric space d. The distance between any pair of points $i, j \in \mathcal{C} \cup \mathcal{F}$ is denoted by $d(i, j)$. Additionally we are given an integer $k \geq 1$. The goal is to select k facilities $f_1, \ldots, f_k$ to open and assign each client j to an open facility so as to minimize $\max_{i=1}^{k} \sum_{j \in \mathcal{C}} d(f(j), j)$, where $f(j)$ is the facility to which client j is assigned.

We use the terms facility and center interchangeably. We frequently use the term star to refer to a pair (f, S), where f is an open facility in the solution and $S \subseteq \mathcal{C}$ is the collection of clients assigned to f; we also refer to f as the center of this star. The cost of this star, which is the load of facility f, is $\sum_{j \in S} d(f, j)$. Thus, our goal is to find k stars, $(f_1, S_1), (f_2, S_2), \ldots, (f_k, S_k)$, centered at facilities so that they "cover" all the clients (i.e. $\mathcal{C} = \cup_{i=1}^{k} S_i$) and the maximum load of a facility (or cost of the star) is minimized. Throughout, we use OPT to denote an optimum solution and L^{opt} to denote its cost.

3 A PTAS for line metrics

In this section we focus on MLkFL on line metrics and present a PTAS for it. Here, each client/facility $i \in \mathcal{C} \cup \mathcal{F}$ is located at some rational point $v_i \in \mathbb{R}$. It may be that $v_i = v_j$ for $i \neq j$, for instance when we have collocated clients. (As noted earlier, we focus on unit demands for simplicity; but, with a little work, one can handle also integer demands by viewing these implicitly as colocated clients.) To simplify notation we use the term "point" to refer to a client or facility $i \in \mathcal{C} \cup \mathcal{F}$ as well as to its location v_i. The distance $d(i, j)$ between points $i, j \in \mathcal{C} \cup \mathcal{F}$ is simply $|v_i - v_j|$. We assume that $|\mathcal{C} \cup \mathcal{F}| = n$ and that $0 \leq v_1 \leq v_2 \leq \ldots \leq v_n$. For a star (f, S) in a MLkFL solution and for any $j \in S$, say that the open interval with endpoints v_f and v_j is an *arm* of the star (f, S) and we say that f covers j. For $S' \subseteq S$, we sometimes use the phrase "load of f by S'" to refer to the sum of the lengths of arms of f to the clients in S'. The main result of this section is the following theorem.

Theorem 3.1 *There is a $(1 + \varepsilon)$-approximation algorithm for MLkFL on line metrics for any constant $\varepsilon > 0$.*

Our high-level approach is similar to other min-max problems. Namely, we present an algorithm that, given a guess B on the optimum solution value, either certify that $B < L^{opt}$ or else find a solution with cost not much more than B. Our main technical result, which immediately yields Theorem 3.1 is the following.

Theorem 3.2 *Let $\Pi = (\mathcal{C} \cup \mathcal{F}, d, k)$ be a given MLkFL instance. For any constant $\epsilon > 0$ and any $B \geq 0$, there is a polynomial-time algorithm $\mathcal{A}$ that either finds a feasible solution with cost at most $(1 + 18\epsilon) \cdot B$ or declares that no feasible solution with cost at most B exists. If $B \geq L^{opt}$, then it always finds a feasible solution with cost at most $(1 + 18\epsilon) \cdot B$.*

In both Theorems 3.1 and 3.2 we assume $\varepsilon < 1$ and $\epsilon < 1$ without loss of generality.

Proof of Theorem 3.1 : Set $\epsilon := \varepsilon/18$. We use binary search to find a value $B \leq L^{opt}$ such that algorithm $\mathcal{A}$ from Theorem 3.2 finds a solution with cost $\leq (1 + 18\epsilon) \cdot B \leq (1 + 18\epsilon) \cdot L^{opt}$. Return this solution.

Since the points v_i are rational and since $n \cdot v_n$ is clearly an upper bound on the optimum solution, then we may perform the binary search over integers $\alpha \in [0, nv_n\Delta]$ where Δ is such that $v_i\Delta \in \mathbb{Z}$ for each point i. For each such value α in the binary search, we try algorithm $\mathcal{A}$ with value $B = \frac{\alpha}{\Delta}$. ■

In what follows, we describe algorithm $\mathcal{A}$. We will assume that $B \geq L^{opt}$ and show how to find a solution with cost at most $(1 + 18\epsilon) \cdot B$. Let $\mathcal{S}_B$ denote a collection of stars $\{(f_1, S_1), \ldots, (f_k, S_k)\}$ with cost at most B. In the remainder of this section, we will describe some preprocessing steps that simplify the structure of the problem. In Section 3.1 we prove that a well-structured near-optimum solution exists and in Section 3.2 we describe a dynamic programming algorithm that finds such a near-optimum well-structured solution.

Without loss of generality, we assume that $1/\epsilon$ is an integer. We start with some preprocessing steps. Note that $d(j, f) \leq B$ for any $j \in S$ of a star (f, S) in $\mathcal{S}_B$. So, if the distance of two consecutive points on the line is more than B then we can decompose the instance into some instances that the distance of any two consecutive points is at most B. For each of the resulting instances Π', we find the smallest k' such that running the subsequent algorithm on the instance with k' instead of k finds a solution with cost at most $(1 + 18\epsilon)B$. Since we are assuming $B \geq L^{opt}$, then the sum of these k' values over the subinstances is at most k. Note that in each subinstance Π' we can assume $0 \leq v_i \leq n \cdot B$ for each point v_i.

Next, we perform a standard scaling of distances. Move every point $i \in \mathcal{C} \cup \mathcal{F}$ left to its nearest integer multiple of $\frac{\epsilon B}{n}$ and then multiply this new point by $\frac{n}{\epsilon B}$. That is, move i from v_i to $\lfloor v_i \cdot n/\epsilon B \rfloor$. Denote the new position of client/facility i by v'_i. The following lemma describes how the optimum solutions to the original and new locations relate.

Lemma 3.3 *The optimum solution has cost at most $(1 + 1/\epsilon) \cdot n$ in the instance given by the new positions v'. Furthermore, any solution with cost at most $(1 + \alpha\epsilon) \cdot (1 + 1/\epsilon) \cdot n$ for the new positions has cost at most $(1 + (2 + 2\alpha)\epsilon) \cdot B$ in the original instance.*

Proof : After sliding each point v_i left to its nearest integer multiple of $\frac{\epsilon B}{n}$, the distance between any two points changes by at most $\frac{\epsilon B}{n}$. Therefore, the load of any star changes by at most ϵB so each star has load at most $(1 + \epsilon)B$. Finally, after multiplying all points by $\frac{n}{\epsilon B}$ we have that the maximum load of any star is at most $(1 + 1/\epsilon) \cdot n$.

Now consider any solution with cost at most $(1+\alpha\cdot\epsilon)\cdot(1+1/\epsilon)\cdot n$. Scaling the points v' back by $\epsilon B/n$ produces a solution with cost at most $(1+\alpha\cdot\epsilon)(1+\epsilon)B \leq (1+(1+2\alpha)\epsilon)\cdot B$. Then sliding, any two points i, j back to their original positions v_i, v_j changes their distance by at most $\epsilon B/n$, so doing this for all points changes the cost of any star by at most ϵB. The resulting stars then have cost at most $(1 + (2 + 2\alpha)\epsilon) \cdot B$. ■

In subsequent sections, we describe a $(1 + 8\epsilon)$-approximation for any one of the subinstances Π' of Π, except we use the new points v'_i. By Lemma 3.3, this gives us a solution to Π with cost at most $(1 + 18\epsilon)B$, proving Theorem 3.2

To simplify notation, we use v_i to refer to the *new* location of point $i \in \mathcal{C} \cup \mathcal{F}$ (i.e. rename v'_i to v_i). Similarly, the notation $d(i, j)$ for $i, j \in \mathcal{C} \cup \mathcal{F}$ refers to these new distances $|v'_i - v'_j|$ and B denotes the new budget $(1 + 1/\epsilon) \cdot n$. From now on, we assume our given instance Π of MLkFL satisfies the following properties:

- Each point v_i is an integer between 0 and $(1 + 1/\epsilon) \cdot n^2$ (which is n times the new value of B).
- There is a solution $\mathcal{S}_B$ with cost at most $B = (1 + \frac{1}{\epsilon})n$.

3.1 Structure of Near Optimum Solutions

In this section, we show that there is a near-optimum solution to the instance Π with clients and facilities $\mathcal{C} \cup \mathcal{F}$ that has some suitable structural properties. In Section 3.2, we will find such a solution using a

dynamic programming approach.

We denote the open interval between two points v_i and v_j on the line by $I_{i,j}$ and call this the *arm* between i and j (assuming that one of i, j is a client and the other is a facility). An arm $I_{i,j}$ is *large* if $d(i,j) > \epsilon B$ and is *small* otherwise. We say that two arms $I_{i,j}$ and $I_{i',j'}$ *cross* if $I_{i,j}$ is not contained in $I_{i',j'}$ or vice versa, and $I_{i,j} \cap I_{i',j'} \neq \emptyset$.

A *well-formed solution* for an MLkFL instance is a solution in which the small arms between clients and their assigned facilities (centers) do not cross. We show that there exists a low cost well-formed solution in two steps. First, we demonstrate the existence of a *fractional solution* where there are k (integral) facilities and the clients are assigned to these centers fractionally. This will be such that the fractional load of each facility is still at most B, all strictly fractional arms in the support have length at most $2\epsilon B$, and that all small arms in the support of the solution do not cross.

Second, we use a rounding algorithm for the Generalized Assignment Problem (GAP) by Shmoys and Tardos [19] to round such a fractional solution to an integral solution with cost at most $(1 + 2\epsilon)B$. We emphasize that this rounding algorithm is not a part of our algorithm, it is only used to demonstrate the existence of a well-structured solution.

For the first step, we will consider a fractional uncrossing argument to eliminate crossings. Instead of proving the fractional uncrossing process eventually terminates, we will instead provide a potential function that strictly decreases in a fractional uncrossing. This potential function is the objective function of a mixed integer-linear program below; thus an optimal solution will not contain any crossings between small arms in its support.

We let $C_B = \{f_1, \ldots, f_k\}$ denote the centers (facilities) of the stars in the solution $\mathcal{S}_B$ (recall that each star in C_B has cost/load at most B). The variable x_{ij} indicates that client j is assigned to facility $f_i \in C_B$. The first constraint ensures every client is assigned to some facility and the second ensures the cost of a star (i.e. load of a facility) does not exceed B.

$$\begin{aligned}
\text{minimize} \quad & \sum_{f_i \in C_B} \sum_{j \in \mathcal{C}} d(f_i, j) \cdot x_{ij} && && \text{(MIP)}\\
\text{subject to} \quad & \sum_{f_i \in C_B} x_{ij} = 1 && \forall j \in \mathcal{C} \\
& \sum_{j \in V} d(f_i, j) \cdot x_{ij} \leq B && \forall f_i \in C_B \\
& x_{ij} \in \{0, 1\} && \forall i, j : d(f_i, j) \geq 2\epsilon B \\
& 0 \leq x_{ij} \leq 1 && \forall i, j : d(f_i, j) < 2\epsilon B.
\end{aligned}$$

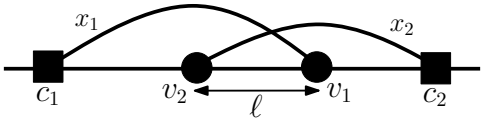
We stress that this is *not* a relaxation for MLkFL. The objective function is more similar to the objective function for the k-median problem. Rather, we will only be using this to help demonstrate the existence of a well-formed solution. The objective function acts as a potential function.

Lemma 3.4 *There is a feasible solution x to mixed integer-linear program* (MIP) *where the small arms in the support of x do not cross.*

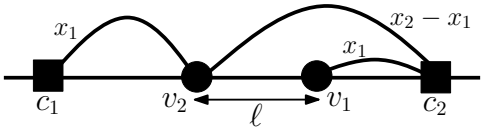
Proof : First observe that there is in fact a feasible solution x because the integer solution $\mathcal{S}_B$ is feasible for this ILP. By standard theory of mixed-integer programming and the fact that the set of feasible solutions is bounded, there is then an optimal solution x. The rest of this proof shows that an optimal solution to (MIP) cannot contain crossings between small arms in its support.

So, suppose x is a feasible solution such that two small arms $I_{i,j}$ and $I_{i',j'}$ in the support of x cross. To simplify notation, let $c_1 = v_{f_i}, c_2 = v_{f_{i'}}$ be the locations of the centers $f_i, f_{i'}$ and $v_1 = v_j$ and $v_2 = v_{j'}$ be the locations of the clients j, j'. Also let x_1 denote x_{ij} and x_2 denote $x_{i'j'}$. That is, x_1 is the extent to which

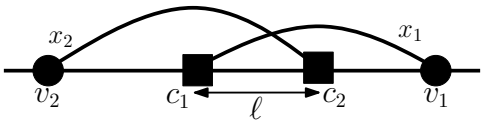
the client at location v_1 is assigned to the center at location c_1 and similarly for x_2. Finally, let $\ell_1 = |c_1 - v_1|$ and $\ell_2 = |c_2 - v_2|$ denote the lengths of the two crossing small arms. See Figure 1 for an illustration of how this notation is used; the notation ℓ will be defined separately for each case considered below.



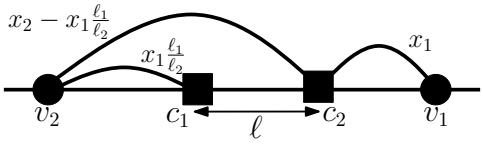
(a) v_1 and v_2 between c_1 and c_2.



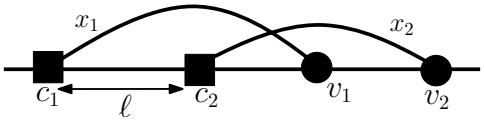
(b) Coverage after fixing the intersection in Case (1).



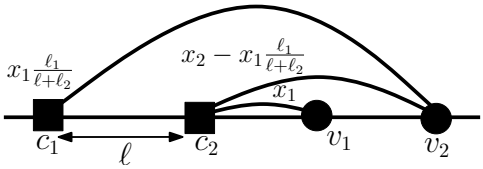
(c) v_1 and v_2 on different sides of c_1c_2.



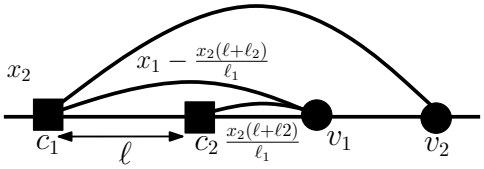
(d) Coverage after fixing the intersection in Case (2).



(e) v_1 and v_2 on the same side of c_1c_2.



(f) Coverage after fixing the intersection in Case (3-1).



(g) Coverage after fixing the intersection in Case (3-2).

Figure 1: Fixing the intersection of two small arms.

We check all possible ways that these two arms can cross. When we say that we shift some value α of coverage from one variable x' to another x'', we mean increase x'' by α and decrease x' by α. Note that we will always shift value between the $x_{ij}, x_{i'j}, x_{ij'}$ and $x_{i'j'}$ values. Since $\ell_1, \ell_2 \leq \epsilon B$ then $d(f_i, j')$ and $d(f_{i'}, j) \leq 2\epsilon B$ so such an uncrossing will maintain the constraint that only arms of length at most $2\epsilon B$ may be fractional.

(1) v_1 and v_2 lie between c_1 and c_2 (Figure 1a). Let $\ell > 0$ be the length of intersecting parts of these arms. Without loss of generality, assume that $x_1 \leq x_2$. Shift x_1 coverage from x_{ij} to $x_{ij'}$ and from $x_{i'j'}$ to $x_{i'j}$ and note that this preserves feasibility, since each client is still covered (fractionally) to the extent of 1. The coverage x after this shifting is depicted in Figure 1b. The total cost of the two stars (centered at c_1 and c_2) decreases by $2\ell x_1 > 0$, so the objective function strictly decreases and we are left with an even cheaper solution to (MIP).

(2) v_1 and v_2 are on different sides of the segment c_1c_2 (Figure 1c). Let $\ell > 0$ be the distance between c_1 and c_2. Without loss of generality, assume that $x_1\ell_1 \leq x_2\ell_2$. Shift $x_1\frac{\ell_1}{\ell_2}$ from $x_{i'j'}$ to $x_{ij'}$ and shift x_1 from x_{ij} to $x_{i'j}$ (Figure 1d).

We verify that the fractional load at each center c_1 and c_2 does not increase. That is, the load at c_1 changes by $x_1\frac{\ell_1}{\ell_2}(\ell_2 - \ell) - x_1\ell_1 = -x_1\frac{\ell_1}{\ell_2}\ell < 0$ and the load at c_2 changes by $x_1(\ell_1 - \ell) - x_1\frac{\ell_1}{\ell_2}\ell_2 = -x_1\ell < 0$. Since the load at both facilities strictly decreases then this is also yields a cheaper solution to (MIP).

(3) v_1 and v_2 are on the same side of the segment c_1c_2 (Figure 1e). Let $\ell > 0$ be the distance between c_1 and c_2. Without loss of generality, assume that v_1 and v_2 are on the right side of segment c_1 and c_2 and the left center is c_1. This means v_1 is between c_2 and v_2 and hence, $\ell_1 < \ell + \ell_2$. As a consequence: $(\ell + \ell_2)(\ell_1 - \ell) < \ell_1\ell_2$.

There are two sub-cases:

Case (3-1): $x_1\ell_1 \leq x_2(\ell + \ell_2)$.
This means $x_1\frac{\ell_1}{\ell+\ell_2} \leq x_2$. We shift x_1 from x^*_{ij} to $x^*_{i'j}$ and shift $x_1\frac{\ell_1}{\ell+\ell_2}$ from $x^*_{i'j'}$ to $x^*_{ij'}$ (Figure 1f). The fractional load at s_1 changes by $x_1\frac{\ell_1}{\ell+\ell_2}(\ell + \ell_2) - x_1\ell_1 = 0$ and the fractional load at s_2 changes by $x_1(\ell_1 - \ell) - x_1\frac{\ell_1}{\ell+\ell_2}\ell_2 = x_1(\ell_1 - \ell - \frac{\ell_1\ell_2}{\ell+\ell_2}) < 0$. Since the total load strictly decreases, then this is also yields a cheaper solution to (MIP).

Case (3-2): $x_1\ell_1 > x_2(\ell + \ell_2)$.
We shift x_2 from $x^*_{i'j'}$ to $x^*_{ij'}$ and shift $\frac{x_2(\ell+\ell_2)}{\ell_1}$ from x^*_{ij} to $x^*_{i'j}$ (Figure 1g). The fractional load at s_1 changes by $x_2(\ell + \ell_2) - x_2\frac{\ell+\ell_2}{\ell_1}\ell_1 = 0$ and the fractional load at s_2 changes by $x_2\frac{(\ell+\ell_2)(\ell_1-\ell)}{\ell_1} - x_2\ell_2 < 0$. Since the total load strictly decreases, then this is also yields a cheaper solution to (MIP).

In all these cases, the new solution is feasible and has a smaller objective values as required. ■

We will use Lemma 3.4 to prove the existence of a near-optimum solution to instance Π where the small arms used by clients do not cross. To complete this proof, we rely on a structural result concerning the polytope of a relaxation for the following scheduling problem.

Definition 3.5 In the scheduling problem on unrelated machines, we are given machines $m_1, \ldots, m_k$, jobs $j_1, \ldots, j_n$, and processing times $p(m_i, j_a) \geq 0$ for any job j_a and any machine m_i. The goal is to assign each job j_a to a machine $\phi(j_a) \in \{m_1, \ldots, m_k\}$ to minimize the maximum total running time $\sum_{a:\phi(j_a)=m_i} p(m_i, j_a)$ of any machine.

Shmoys and Tardos [19] prove a result concerning the polytope of an LP relaxation for this problem, as a part of a more general result concerning the related *Generalized Assignment Problem* (GAP). The following summarizes the results they obtain that are relevant for our work.

Theorem 3.6 (Shmoys and Tardos, [19]) *Suppose we have a bound B and fractional values $x(m_i, j_a) \geq 0$ for each job j_a and each machine m_i that satisfy the following:*

- $\sum_{i=1}^{k} x(m_i, j_a) = 1$ *for each job j_a,*
- $\sum_{a=1}^{n} p(m_i, j_a) \cdot x(m_i, j_a) \leq B$ *for each machine m_i.*

Then there is an assignment ϕ of jobs to machines such that $x(\phi(j_a), j_a) > 0$ for each job j_a and the maximum load of any machine under ϕ is at most $B + \max_{a,i:0<x(m_i,j_a)<1} p(m_i, j_a)$.

We use the above theorem together with Lemma 3.4 to prove the following.

Theorem 3.7 *There is a feasible (integer) solution to the* MLkFL *instance Π with maximum load $(1+2\epsilon)B$ on each star such that no two small arms cross.*

Proof : Let x^* be the fractional solution provided by Lemma 3.4. We view x^* as a solution to the following scheduling problem on unrelated machines. We have k machines $m_1, \ldots, m_k$, each corresponding to a

facility $f_i \in C_B$. For each client $a \in \mathcal{C}$, there is a single job j_a. The processing time $p(m_i, j_a)$ of job j_a on machine m_i is $|v_i - v_a|$, the distance between the corresponding locations.

Now, x^* fractionally assigns each job j_a to the machines to a total extent of 1 and the maximum (fractional) load at machine m_i is B. Furthermore, the only strictly fractional assignments (i.e. those with $0 < x_{ij} < 1$) have $|v_i - v_j| \leq 2\epsilon B$. In the scheduling terminology, the only strictly fractional assignments are between a job j_a and a machine m_i such that $p(m_i, j_a) \leq 2\epsilon B$.

Theorem 3.6 shows we can transform this fractional assignment x^* into an integer assignment such that a) if client j is assigned to facility/center i, then $x^*_{ij} > 0$ and b) the maximum load of a facility is $B + \max_{i,j:0<x^*_{ij}<1} |v_i - v_j| \leq B + 2\epsilon B$. In this solution, small arms used by clients do not cross because they come from the support of x^*. ■

3.2 Finding a Well-Formed Solution

3.2.1 Step Min-max Cost

Theorem 3.7 shows that there is a solution of cost at most $(1+2\epsilon)B$ such that no two small arms (i.e. length $\leq \epsilon B$) used to assign clients to centers cross. Call this solution $\mathcal{S}'_{\mathcal{B}}$. We now show that we can find such a well-structured solution of cost at most $(1 + 8\epsilon)B$.

The main idea behind our approach is the following. If it were true that a near-optimum solution did not have any crossing arms (large or small) then we can find such a solution using a dynamic programming approach. At a very high-level, we could exploit the laminar structure of the solution by decomposing the solution into a family of nested intervals $\mathcal{I}$ such that for every $I \in \mathcal{I}$ there is one center c with $v_c \notin I$ such that clients in I are served either by centers in I or by c. From this, we can consider triples (I, c, r) where $I \in \mathcal{I}$, c is a location outside of I, and r is some integer between 0 and $\mathrm{poly}(n, 1/\epsilon)$ describing the load assigned to c from clients in I. We can look for partial solutions parameterized by these triples and relate them through an appropriate recurrence.

Unfortunately, we are only guaranteed that the small arms do not cross in our near-optimum solution so the collection of all arms in the solution is not necessarily laminar. To handle this general case, we must carry extra information about large arms through our dynamic programming approach and keep track of how many large arms of each possible length cross an interval. Since each large arm has a length between ϵB and B, if we store the exact length, then this amounts to storing a vector with roughly B coordinates. There are exponentially many vectors with B coordinates that the values in different coordinates sum up to at most n, so we cannot keep track of this information. However, by coarsening the length, we ensure that large arms have only a constant possible number of different perceived length, so we can keep track of this information when we move to perceived length.

First, recall that all large arms have length more than ϵB. Thus, each facility is serving at most $\frac{(1+2\epsilon)B}{\epsilon B} \leq \frac{3}{\epsilon}$ clients that are at distance more than ϵB from it; in other words each facility is assigned at most $\frac{3}{\epsilon}$ large arms in the solution provided by Theorem 3.7. Since large arms have length at least ϵB, if we store their length in multiple of $\epsilon^2 B$, we will not lose much information about them. Let $\mathsf{MULT}(\mathsf{i}, \mathsf{j})$ denote the number of multiples of $\epsilon^2 B$ in $(v_i, v_j]$ interval for $v_i \leq v_j$, so

$$\mathsf{MULT}(\mathsf{i}, \mathsf{j}) = \left\lfloor \frac{v_j}{\epsilon^2 B} \right\rfloor - \left\lfloor \frac{v_i}{\epsilon^2 B} \right\rfloor.$$

Then, we measure the the length of a large arm I_{ij} between client j and facility i as $\epsilon^2 B \cdot \mathsf{MULT}(\mathsf{i}, \mathsf{j})$ if $v_i \leq v_j$ or $\epsilon^2 B \cdot \mathsf{MULT}(\mathsf{j}, \mathsf{i})$ otherwise. We call this the perceived cost of this arm. In this method of measurement, the length of the arm changes by at most $\epsilon^2 B$, and so the total load for each center changes by at most $3\epsilon B$. Now for this method of measurement, since there are $\frac{1}{\epsilon^2}$ coordinates, the number of possible vectors for keeping track of large arms is at most $n^{\frac{1}{\epsilon^2}}$ which is polynomial.

In the dynamic programming algorithm described below, we will use this coarse method to measure the distance of large arms. Since in dynamic programming approach the problem is often broken into subproblem, a large arm might be partitioned with respect to subproblems. We would like to note that for points $v_{p_0} \leq v_{p_1} \leq \cdots \leq v_{p_t}$ with $p_0 = i$ and $p_t = j$, $\mathsf{MULT}(\mathsf{i},\mathsf{j}) = \sum_{q=0}^{t-1} \mathsf{MULT}(\mathsf{p_q}, \mathsf{p_{q+1}})$ (Note that $\mathsf{MULT}(\mathsf{a},\mathsf{a}) = 0$ for any a.). We use the term the *perceived cost* of a facility i to denote the the total cost of the small arms plus the perceived cost of its large arms. The following is proved using arguments similar to the proof of Lemma 3.3, recalling that every facility i n $\mathcal{S}'_{\mathcal{B}}$ has at most $3/\epsilon$ large arms.

Lemma 3.8 *The perceived cost of every star in $\mathcal{S}'_{\mathcal{B}}$ is at most $(1+5\epsilon)B$. Furthermore, any star with perceived cost at most $(1+5\epsilon)B$ and at most $3/\epsilon$ long arms has (actual) cost at most $(1+8\epsilon)B$.*

Our dynamic programming algorithm will find a solution with perceived cost at most $(1+5\epsilon)B$ and at most $3/\epsilon$ large arms per star, so the actual cost will be at most $(1+8\epsilon)B$.

3.2.2 Dynamic Programming

Before we formally define the subproblems of dynamic programming, we discuss the structure of a well-formed solution, say $\mathcal{S}$. We call a client covered by a small (large) arm a *small client* (*large client*), respectively. It will be convenient to associate a direction with each arm, which goes from the center/facility to the client. For a star S with center f, let S_{small} denote the clients covered by small arms in S. Let the *s-span* of S be the open interval, possibly empty, spanning (l_S, r_S) where

$$l_S = \min\left\{v_f, \min_{j \in S_{\text{small}}} v_j\right\} \quad \text{and} \quad r_S = \max\left\{v_f, \max_{j \in S_{small}} v_j\right\}$$

are the left most and the right most small clients (or facility) in this star, respectively. Since the small arms do not intersect in $\mathcal{S}$, for any two s-spans I_1 and I_2 of two stars, either $I_1 \cap I_2 = \emptyset$ or $I_1 \subseteq I_2$ or $I_2 \subseteq I_1$. Therefore, the $\subseteq$ relation between s-span of stars in $\mathcal{S}$ defines a laminar family. A laminar family can naturally be viewed as a forest where we put a node for each member of the family and each node I has an edge to the minimal member say I' of the family that contains it, i.e., $I \subseteq I'$. If a facility f does not serve any client by a small arm then its s-span is (v_f, v_f) by definition and although this is an empty interval, in the forest corresponding to laminar family, it has an edge to the minimal s-span (interval) that contains v_f. We frequently refer to this forest-view when referring to a laminar family.

Let us try to understand the subproblems that come up in constructing a solution. The dynamic programming in fact stitches together the solutions of these subproblems in order to find the solution to the original problem. For indices $1 \leq l \leq r \leq n$, let $V_{l,r}$ denote the set of points $\{v_l, v_{i+1}, \cdots, v_r\}$ with $l \leq r$, so $V = V_{1,n}$. We want to use the dynamic programming to answer the question if it is possible to open k centers in $V_{1,n}$ and assign clients to these centers such that the perceived cost of each center is at most $(1+5\epsilon)B$ and the s-span of the centers form a laminar family.

Now consider solution $\mathcal{S}'_{\mathcal{B}}$ which has a maximum perceived load of $(1+5\epsilon)B$. Consider the forest corresponding to the s-span of centers in this solution and let c be the center that its s-span is the leftmost root in the forest. We can guess c as there are at most $O(n)$ possible choices for it. Let k_r and k_l denote the number of centers in $\mathcal{S}'_{\mathcal{B}}$ which are on the right and left side of c respectively, so $k_r + k_l = k - 1$. There are $O(k)$ possible choice for values k_r and k_l, so we guess k_r and k_l as well (note that k is dominated by n). Let us now focus on the interval $V_{1,c-1}$. Since c corresponds to the interval at the root of the forest, no center in $V_{c+1,n}$ can serve clients in $V_{1,c-1}$ by small arms (otherwise the s-span of c is a subset of s-span of some other center). We can guess the load corresponding to small clients served by c in $V_{1,c-1}$ as this load is an integer in $\text{poly}(n, 1/\epsilon)$. We may have some clients in $V_{1,c-1}$ served by large arms originating in $V_{c,n}$. We cannot guess the large arms serving these clients as the number of possible such arms is not polynomial (there are $O(n^k n^{3/\epsilon})$ possible choices) but instead we can bundle large arms entering $V_{1,c-1}$ based on their

perceived length past v_{c-1}, that is their perceived length in $V_{1,c-1}$ interval. More precisely, we guess how many large arms have perceived length $q \times \epsilon^2 B$ past v_{c-1} for each $0 \leq q \leq \frac{1}{\epsilon^2}$ as there are $O(n^{\frac{1}{\epsilon^2}})$ possible choices for our guess. Similarly, we may have some large arms originating in the interval $V_{1,c-1}$ which serve clients in $V_{c+1,n}$. Again we can bundle these arms based on their perceived length past v_{c+1} (their perceived length in $V_{c+1,n}$) and we guess the number of large arms with perceived length $q \times \epsilon^2 B$ past v_{c+1} for each $0 \leq q \leq \frac{1}{\epsilon^2}$. This give us an idea of what parameters are needed for describing the subproblems.

Now consider some interval $V_{l,r}$ between two arbitrary points v_l, v_r and consider how $\mathcal{S}'_{\mathcal{B}}$ looks in this interval. There may be some large arms that enter and/or leave this interval from v_l or v_r. The arms that enter the interval can cover the *deficiency* of coverage for some clients in the interval and the arms that leave the interval provide coverage for some clients outside of interval and can be viewed as *surplus* to the demand of coverage of the clients in the interval. We keep track of all large arms crossing the sides of interval $V_{l,r}$ in terms of deficiency and surplus vectors as follows:

- **Deficiencies:** Vector $\mathbf{D}_l$ is the deficiency vector in $[n]^{\frac{1}{\epsilon^2}}$ for v_l where $\mathbf{D}_l[q]$ for $0 \leq q \leq \epsilon^{-2}$ is the number of large arms from a center location $i < l$ to a client $j \geq l$ such that $\mathsf{MULT}(\mathsf{l}, \mathsf{j}) = q$. So $\mathbf{D}_l[q]$ keeps track of the number of large arms originating in $i < l$ and crossing exactly q multiples of $\epsilon^2 B$ in interval $(v_l, v_j]$. Note that client $j \geq l$ can be located outside of interval $V_{l,r}$, i.e., $j > r$ as well. The vector $\mathbf{D}_r$ is defined similarly for v_r, that is, $\mathbf{D}_r[q]$ is the number of large arms from a center location $i > r$ to a client $j \leq r$ such that $\mathsf{MULT}(\mathsf{j}, \mathsf{r}) = q$. Let $q' = \mathsf{MULT}(\mathsf{l}, \mathsf{r})$, then all arms represented by $\mathbf{D}_l[q]$ for $q < q'$ must end at clients located in $V_{l,r}$ and all arms represented by $\mathbf{D}_l[q]$ for $q > q'$ must end at clients located to the right of $V_{l,r}$. If $q = q'$ then some arms may end at clients in $V_{l,r}$ and some may end at clients located to the right of $V_{l,r}$.

- **Surpluses:** Vector $\mathbf{S}_l$ is the surplus vector in $[n]^{\frac{1}{\epsilon^2}}$ for v_l where $\mathbf{S}_l[q]$ for $0 \leq q \leq \epsilon^{-2}$ is the number of large arms from a center location $i \geq l$ to a client $j < l$ such that $\mathsf{MULT}(\mathsf{j}, \mathsf{l}) = q$. So $\mathbf{S}_l[q]$ keeps track of large arms originating at $i \geq l$ and crossing exactly q multiples of $\epsilon^2 B$ in interval $(v_j, v_l]$. Note that center $i \geq l$ can be larger than r and does not need to be located in the interval $V_{l,r}$. The vector $\mathbf{S}_r$ is defined similarly, that is, $\mathbf{S}_r[q]$ is the number of large arms from a center location $i \leq r$ to a client $j > r$ such that $\mathsf{MULT}(\mathsf{r}, \mathsf{j}) = q$. Recall that $q' = \mathsf{MULT}(\mathsf{l}, \mathsf{r})$. Note that for $q > q'$, any arm contributing to $\mathbf{D}_l[q]$ also contributes to $\mathbf{S}_r[q - q']$ and similarly, any arm contributing to $\mathbf{D}_r[q]$ also contributes to $\mathbf{S}_l[q - q']$.

3.3 The Table

The table we build in our dynamic programming algorithm captures "snapshots" of solutions bound between two given points plus some information on how arms cross these points. We consider boolean value $A[k', l, r, c, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r]$ corresponding to subproblems. The meanings of the parameters are as follows.

- $0 \leq k' \leq k$ is the number of centers in the interval $V_{l,r}$.
- $1 \leq l \leq r \leq n$ corresponds to the interval $V_{l,r}$.
- $c \in \mathcal{F}$ denotes a single point with either $c < l$ or $c > r$ (i.e. outside of $V_{l,r}$), that is the center of some star, or else $c = \perp$. If $c \neq \perp$ then it is supposed to be the only center outside of $V_{l,r}$ with small arms going into $V_{l,r}$ and the total cost of small arms that c pays to cover clients i' with $l \leq i' \leq r$ is β where $0 \leq \beta \leq (1 + 5\epsilon)B$ is an integer.
- $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$ are deficiency and surplus vectors for the endpoints of interval $V_{l,r}$.

Note that in the above, if $c = \perp$ then the value of β can be assumed to be zero.

The value $A[k', l, r, c, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r]$ is TRUE if and only if the following holds. It is possible to open k' centers in the interval $V_{l,r}$ and assign each client $j \in \mathcal{C}$ with $l \leq j \leq r$

- to one of the k' open centers,
- to center c, if $c \neq \perp$,
- or to a large arm entering $V_{l,r}$

and also assign the start of some of the large arms exiting the interval to these open centers in $V_{l,r}$ such that the following hold.

- The perceived load of each of the k' centers is at most $(1 + 5\epsilon)B$.
- The load of c from small arms originating from clients $j \in \mathcal{C}$ with $l \leq j \leq r$ is β,
- The large arms entering and/or exiting $V_{l,r}$ are *consistent* with $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$.

By consistent, we mean the following. For each interval $[a, b]$ where a and b are consecutive multiples of $\epsilon^2 B$, we check the number of promised clients to be served by $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$ in this interval (depending on position of a and b with respect to l and r, some of these vectors may not give any information). More precisely, when $b \geq r$, each of $\mathbf{D}_l[q_1]$ and $\mathbf{S}_r[q_2]$ where $q_1 = \mathsf{MULT}(\mathsf{l}, \mathsf{a})$ and $q_2 = \mathsf{MULT}(\mathsf{r}, \mathsf{a})$ gives the number of clients in $[a, b)$ that are supposed to be served by centers $i \leq l$ and $i' \leq r$, respectively. Now since $l \leq r$, any large arm counted in $\mathbf{D}_l[q_1]$ must be counted in $\mathbf{S}_r[q_2]$, i.e., $\mathbf{S}_r[q_2] \geq \mathbf{D}_l[q_1]$, and we must have at least $\mathbf{S}_r[q_2]$ clients in $[a, b)$. Note that $\mathbf{S}_r[q_2] - \mathbf{D}_l[q_1]$ correspond to long arms that start in interval $V_{l,r}$. Similar arguments must be made for when $a \leq l$ and $\mathbf{D}_r$ and $\mathbf{S}_l$. For intervals containing l or r, we have to subdivide the interval, e.g., $[a, r)$ and $[r, b)$, and then check the consistency.

The number of table entries is polynomial, because k', l, r, c are in $O(n)$ and β is a polynomial in n and $\frac{1}{\epsilon}$ and the deficiency and surplus vectors, in total, can take one of $O(n^{1+1/\epsilon^2})$ values. We shortly explain how one can compute the values A in polynomial time through dynamic programming. After that, to find out if there is a feasible solution having perceived cost $(1 + 5\epsilon)B$, one simply needs to look at the value of $A[k, 1, n, \perp, 0, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}]$, where $\mathbf{0}$ is a vector having $1 + 1/\epsilon^2$ zero components.

3.4 The Recurrence

In the remaining of the section, we explain how the value of a table entry is calculated. We call a subproblem *feasible* if $A[k', l, r, c, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r]$ is TRUE.

Base case. The base case is when $k' = 0$ and $l = r$. Since $k' = 0$, we are not allowed to open a facility at v_l (l might not be a facility anyway). There are two possible main cases based on whether c can serve a possible client at r or not.

- $c = \perp$ and so $\beta = 0$ or $c \neq \perp$ but $\beta = 0$.
  If r is a client, it has to be served by a large arm coming from outside of $V_{l,r}$. Suppose r is served by a large arm entering from left, so $\mathbf{D}_l[0]$ must be non-zero and moreover, values of $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$ must be consistent, i.e., $\mathbf{D}_l[0] = \mathbf{S}_r[0] + 1$, $\mathbf{D}_l[q] = \mathbf{S}_r[q]$ for $1 \leq q \leq \epsilon^{-2}$, and $\mathbf{D}_r = \mathbf{S}_l$. Similar arguments works for a large arm serving r from right. So if either of these conditions hold, then the subproblem is feasible and we set the table entry to TRUE.

  If r is a facility, then it cannot be opened as $k' = 0$. So we only check consistency of $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$, i.e., $\mathbf{D}_l = \mathbf{S}_r$ and $\mathbf{D}_r = \mathbf{S}_l$. If these conditions hold, we set the value of the table entry to TRUE otherwise we set it to FALSE.

- $c \neq \perp$ and $\beta \neq 0$.
  If r is a client, it has to be served by a small arm originating at c, so β must be equal to $d(v_j, v_c)$ and it must be smaller than ϵB. The vectors $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$ must be consistent, i.e., $\mathbf{D}_l = \mathbf{S}_r$ and $\mathbf{D}_r = \mathbf{S}_l$.

  If r is a facility, then the subproblem is infeasible by the definition.

Recursive step. Next, we show how to determine if $A[k', l, r, c, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r]$ is TRUE when the parameters do not represent a base case by relating its value to values of smaller problems. In what follows, by *guessing* a parameter, we mean that we try all *polynomially* many possible values of that parameter and if one of them results in a feasible solution, we set the value of the current subproblem to TRUE. We consider two cases:

Case (1): $c \neq \perp$ and $\beta > 0$.
Without loss of generality, suppose $c < l$. There must be a small client j with $l \leq j \leq r$ covered by c. We guess j to be the leftmost such small client along with how many of k' facilities in $V_{l,r}$ are in $V_{l,j-1}$, call this k'' (the rest of facilities, $k' - k''$, will be in $V_{j+1,r}$). For subproblem constructed for $V_{l,j-1}$, no small arm can enter $V_{l,j-1}$, and for subproblem constructed for $V_{j+1,r}$, the center outside $V_{j+1,r}$ is c with allowed load of $\beta' = \beta - d(v_j, v_c)$. We can also guess the large arms leaving and/or entering $V_{l,j-1}$ as well as $V_{j+1,r}$ and in polynomial time, we check if these vectors are consistent with each other as well as $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$. So $A[k', l, r, c, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r]$ is set to TRUE if one of the following expressions is TRUE (see Figure 2).

$$A[k'', l, j-1, \perp, 0, \mathbf{D}_l, \mathbf{D}_{j-1}, \mathbf{S}_l, \mathbf{S}_{j-1}] \wedge A[k'-k'', j+1, r, c, \beta - |v_c - v_j|, \mathbf{D}_{j+1}, \mathbf{D}_r, \mathbf{S}_{j+1}, \mathbf{S}_r]$$

for some $l \leq j \leq r$ such that there is a small arm from c to j, denoted by I_{cj} and $|v_c - v_j| \leq \beta$, some $0 \leq k'' \leq k'$, and $\mathbf{D}_{j-1}, \mathbf{S}_{j-1}, \mathbf{D}_{j+1}, \mathbf{S}_{j+1}$ consistent with. $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$, (using the assumption that j is served with a small arm)

Note that when $j = l$, we just check one other subproblem, namely $A[k', j+1, r, c, \beta - |v_c - v_j|, \mathbf{D}_{j+1}, \mathbf{D}_r, \mathbf{S}_{j+1}, \mathbf{S}_r]$ (we must have I_{cj} is a small arm, and $|v_c - v_j| \leq \beta$). Similarly, when $j = r$, we just check one subproblem, namely $A[k', l, j-1, \perp, 0, \mathbf{D}_l, \mathbf{D}_{j-1}, \mathbf{S}_l, \mathbf{S}_{j-1}]$ (we must have I_{cj} is a small arm, and $|v_c - v_j| = \beta$). If $l = r$ then k' has to be non-zero (otherwise we are in a base case), and since there is no facility to open at j, we say the subproblem is infeasible. Similar argument can be made when $c > r$ (in this case, we guess the rightmost client served by c).

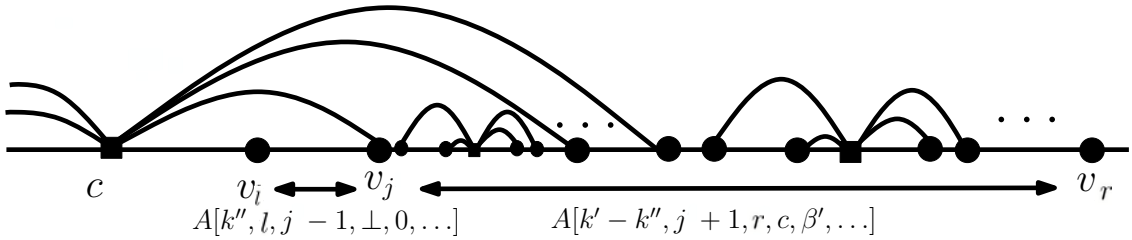


Figure 2: Case 1 of recursive step

Case (2): $c \neq \perp$ and $\beta = 0$, or $c = \perp$.
We consider two subcases regarding value of k':

Case (2-a): $k' = 0$.

In this case $l \neq r$ (otherwise we are in a base case). All clients in $V_{l,r}$ must be covered by large arms from centers (facilities) outside the interval.

If l is a facility then it must be closed so we recursively check $A[0, l{+}1, r, c, 0, \mathbf{D}_l{}', \mathbf{D}_r{}', \mathbf{S}_l{}', \mathbf{S}_r{}']$ where the new deficiency and surplus vectors are obtained from $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$ after taking into account the number α of multiples of $\epsilon^2 B$ between v_l and v_{l+1}. If this is not possible, e.g. if $\mathbf{D}_r(q) > 0$ for some $q < \alpha$ or similarly, then the subproblem is not feasible.

So, suppose that l is a client. First assume l is covered by a large arm from the left. Then $\mathbf{D}_l(0) > 0$ and we use one to cover client l. In this case, define $\mathbf{D}_l{}', \mathbf{D}_r{}', \mathbf{S}_l{}', \mathbf{S}_r{}'$ to reflect the fact that this one large arm covers l and the perceived length of the remaining ones accounted for by $\mathbf{D}_l, \mathbf{S}_l$ that entered or exited by v_r have a different perceived length depending on the number of multiples of $\epsilon^2 B$ between v_l and v_{l+1} (again, if this is not possible then the subproblem is not feasible).

Then we declare the subproblem to be feasible if and only if $A[0, l{+}1, r, c, 0, \mathbf{D}_l{}', \mathbf{D}_r{}', \mathbf{S}_l{}', \mathbf{S}_r{}'] =$ TRUE. A similar argument works if l is covered by a large arm from the right.

Case (2-b): $k' > 0$.

Note that since $c \neq \perp$ and $\beta = 0$, or $c = \perp$, no small arm can enter $V_{l,r}$. Consider the set of centers in $V_{l,r}$. The s-span (interval of small arms) of these centers forms a laminar family. Consider the roots of the forest of this laminar family and let c' be the center corresponding to the leftmost root; we guess c' (see Figure 3) along with the contribution of small arms originating at c' going to the left (call β') and the right (call β''), and also the number of centers located between l and i, say k''. Observe that the s-span of i is not contained in the s-span of any other center in $V_{l,r}$. Center c' has at most $3/\epsilon$ large arms. We guess the large arms of c' along with large arms entering/leaving $V_{l,c'-1}$ and $V_{c'+1,r}$. For all the guesses that the perceived cost of c' is at most $(1 + 5\epsilon)B$ and the large arms are consistent with each other as well as $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$. So $A[k', l, r, c, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r]$ is set to TRUE if one the following expressions is TRUE (see Figure 3).

$$A[k'', l, c'{-}1, c', \beta', \mathbf{D}_l, \mathbf{D}_{c'-1}, \mathbf{S}_l, \mathbf{S}_{c'-1}] \wedge A[k'{-}k''{-}1, c'{+}1, r, c', \beta'', \mathbf{D}_{c'+1}, \mathbf{D}_r, \mathbf{S}_{c'+1}, \mathbf{S}_r]$$

for some $l \leq c' \leq r$, $0 \leq k'' \leq k' - 1$ guessed $\leq 3/\epsilon$ long arms such that the perceived cost at c' is at most $(1 + 5\epsilon)B$, guessed large arms and $\mathbf{D}_{c'-1}, \mathbf{S}_{c'-1}, \mathbf{D}_{c'+1}, \mathbf{S}_{c'+1}$ consistent with $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$.

4 Tree Metrics

The extension of the PTAS presented for line metrics to tree metrics is not clear because we heavily exploited the linear structure of line metrics in the PTAS from Section 3. However, we can obtain a QPTAS for this case using a different approach. Our approach is inspired by the QPTAS for line metrics from [12] in that it uses a recursive decomposition of the tree into geometrically smaller subtrees, but the parameters used to define a subproblem are a fair bit different.

Let T be a tree with edge costs $d_e, e \in T$. For two nodes $u, v \in T$, let $d(u, v)$ be the cost of all edges on the unique $u - v$ path in T. Also for a node v and edge $e = uw$, let $d(v, e) = \min\{d(v, u), d(v, w)\}$. Let n denote the total number of nodes in T. Not all nodes are required to be a client or a facility. As in the case of line metrics, we focus on unit demands for (mostly notational) simplicity.

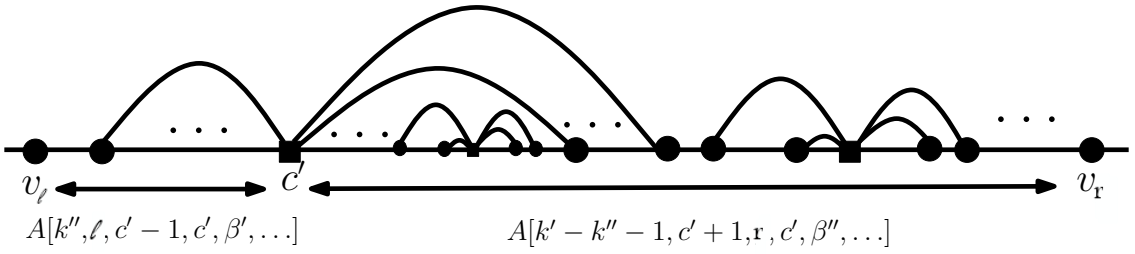


Figure 3: Case (2-b) of recursive step

The algorithm, as before, works with a guessed value B as an upper bound for L^{opt}. Delete every edge with cost greater than B and focus on one of the resulting subtrees. We try all guesses for $k' \leq k$ to determine the smallest one for which there is a solution with cost $(1 + O(\epsilon)) \cdot B$ in this subtree (if any). If any of these subtrees does not have such a solution for any $k' \leq k$ or if the sum of these values k' over all subtrees exceeds k, we determine there is no solution with cost $\leq B$. From now on, we will simply let k denote this guessed value for k' in the subtree and assume all edge costs are at most B.

Next, we make T have degree at most 3 by "expanding" any node with at least four neighbors. That is, if v is a node with at least four neighbors then we add a new vertex v' that is neither a client nor a facility, connect v' to v with a 0-cost edge, and have two neighbors, say u, w, of v instead be neighbors of v' with the same edge cost (i.e. $d_{uv'} = d_{uv}$ and $d_{wv'} = d_{wv}$). The optimum solution cost does not change. The degree of v strictly decreases, so iterating this process until all nodes have degree at most 3 produces a tree with at most $2n$ nodes. Let T' denote this tree and say it has $n' \leq 2n$ nodes.

Using a scaling argument as for the case of line metrics, we can assume that the aspect ratio of heaviest to cheapest nonzero edge cost is polynomially bounded: edges with cost less than $\epsilon B/n'^2$ are changed to have a cost of 0. Let T'' denote this modification of T'. Since each of the n' clients hops across at most n' edges to their assigned facility in any solution, the difference in cost when measuring a solution in T' or T'' is at most ϵB. Finally, scale the costs by $n'^2/(\epsilon B)$, so each edge has cost either 0 or in the range $[1, n'^2/\epsilon]$. Let T'' denote the tree produced from T' after the edge costs are scaled this way.

Observation 4.1 *If there was a solution with cost B in the original tree T, then there is a solution with cost at most $(1+\epsilon) \cdot n'^2/\epsilon$ in T''. Conversely, any solution with cost at most $(1+\epsilon) \cdot n'^2/\epsilon$ in the T'' will have cost at most $(1+\epsilon) \cdot B$ in T.*

From now on, we will assume that T has maximum degree 3 and each nonzero edge cost lies in the range $[1, n^2/\epsilon]$ where n denotes the number of nodes in T. Summarizing the above discussion, we are assuming:

- For each edge e of T, either $d_e = 0$ $d_e \in [1, n^2/\epsilon]$.
- $B = (1+\epsilon) \cdot n^2/\epsilon$.

4.1 The Decomposition Tree

We construct a rooted *decomposition tree* $\mathcal{T}$ for T, which is itself a tree whose nodes correspond to connected subtrees of T. The tree T itself will be the root of $\mathcal{T}$.

To construct $\mathcal{T}$, initially set it to be the tree with just T as its only node. Then, while there is some leaf node T' in $\mathcal{T}$ (taking $T' = T$ if $\mathcal{T}$ only includes the one node T) with $n' \geq 2$ nodes, find an edge $e = (u, v)$

such that the two connected components of $T' - e$ have size in $[n'/3, 2n'/3]$. It is well known that such an edge exists in any tree with degree at most 3. Add two subtrees as children of T' in $\mathcal{T}$. Denote this edge e by $e_{T'}$.

The height of $\mathcal{T}$ is $O(\log n)$ because the sizes of the subtrees on any root-to-leaf path in $\mathcal{T}$ decrease geometrically. Each edge $e \in T$ is of the form $e_{T'}$ for precisely one non-leaf $T' \in \mathcal{T}$. For a subtree T' in $\mathcal{T}$, we let

$$p(T') = \{e_{T^*} : T^* \text{ a proper ancestor of } T' \text{ in } \mathcal{T}\}$$

be the set of *portal* edges for T. Note that have that $p(T) = \emptyset$ and if T'' is a child of T' in $\mathcal{T}$ then $p(T'') = p(T') \cup \{e_{T'}\}$. This also shows $|p(T')| \leq \text{height}(\mathcal{T}) = O(\log n)$ for any subtree $T' \in \mathcal{T}$.

Observation 4.2 *Let $T' \in \mathcal{T}$ and consider any $u \in T'$, $v \notin T'$. The $u - v$ path crosses at least one edge in $p(T')$ (perhaps more). In particular, it crosses $e_{T''}$ where T'' is the least common ancestor of the leaf nodes T_u and T_v in $\mathcal{T}$ corresponding to the singleton subtrees $\{u\}$ and $\{v\}$, respectively.*

As in the case of paths, we refer to the directed path from a center c to a client j being served by c as an *arm*. Our recurrence coarsens how we measure different parts of an arm. Let $\mathcal{C} = \{0\} \cup \{(1+\epsilon)^\sigma : 0 \leq \sigma \leq \lfloor \log_{1+\epsilon} B \rfloor\}$ be the set of possible *coarse arm lengths*. Note $|\mathcal{C}| = O(\epsilon^{-1} \log n)$.

We classify arms in the following way. For some $e \in T$ and some $\alpha, \beta \in \mathcal{C}$, we say that the arm $c - j$ has type (e, α, β) if

- The $c - j$ path traverses e and $e = e_{T'}$ where T' is the least common ancestor of the leaf nodes corresponding to the singletons $\{c\}$ and $\{j\}$ in $\mathcal{T}$.
- $\alpha \leq d(c, e) \leq (1+\epsilon) \cdot \alpha$
- $\beta \leq d(e, j) \leq (1+\epsilon) \cdot \beta$

That is, e is the "highest" portal edge traversed by the $c - j$ arm and α, β approximate the $c - e$ and $e - j$ distances within a factor of $1 + \epsilon$. We will say the *perceived* distance of this arm is $\alpha + \beta + d_e$.

4.2 The Recurrence

We still maintain deficiency and surplus vectors for various subtrees, but they are indexed somewhat differently than in our recurrence for line metrics. A subproblem consists of a subtree $T' \in \mathcal{T}$, an integer $0 \leq k' \leq k$, and surplus and deficiency integer vectors $\mathbf{S}, \mathbf{D}$, both indexed by tuples (e, α, β) where $e \in p(T')$ and $\alpha, \beta \in \mathcal{C}$.

Let $A[T', k', \mathbf{S}, \mathbf{D}]$ be a boolean value that is TRUE if it is possible to:

- Open exactly k' centers in T'.
- Assign each client $c \in T'$ to one of these centers or is served by a center outside T' using an arm of type (e, α, β) where $e \in p(T')$. This should be done such that exactly $\mathbf{D}(e, \alpha, \beta)$ clients are assigned to outside T' using an arm of type (e, α, β) and a client c can only be assigned to such an arm if $\beta \leq d(e, j) \leq (1+\epsilon) \cdot \beta$.
- Assign precisely $\mathbf{S}(e, \alpha, \beta)$ arms of type (e, α, β) between the k' open centers for each $e \in p(T')$. Such an arm can only be assigned to a center c if $\alpha \leq d(c, e) \leq (1+\epsilon) \cdot \alpha$.
- The perceived distance of all arms and clients assigned to any one of these k' centers is at most $(1+\epsilon) \cdot B$.

Note that one major difference between this table and the table used in the PTAS for line metrics is that no arm will be counted by both $\mathbf{S}$ and $\mathbf{D}$ in a subproblem. That is, while some arms may "enter and exit" T', they won't be accounted for in this subproblem. An arm is only counted in this subproblem if it has one endpoint in T' and the other endpoint not in T'.

Simply by definition of A, if there is a solution with maximum load B, then $A[T, k, \mathbf{0}, \mathbf{0}]$ will be true and if $A[T, k, \mathbf{0}, \mathbf{0}]$ is true then by replacing the perceived distance of each arm with its actual distance we see there is in fact a solution with maximum load $(1 + \epsilon) \cdot B$.

The recurrence for A is somewhat simpler than the recurrence we used for line metrics. The base cases are precisely when T' corresponds to a subtree with exactly one node (i.e. when T' is a leaf in $\mathcal{T}$).

Base Cases

1. $T' = \{v\}$ where v is neither a client or a facility. Then $A[T', k', \mathbf{S}, \mathbf{D}] = \textsc{True}$ if and only if $k' = 0$ and $\mathbf{S} = \mathbf{D} = \mathbf{0}$ as there are no client or facility nodes in this subtree.

2. $T' = \{j\}$ where j is a client node. Then $A[T', k', \mathbf{S}, \mathbf{D}] = \textsc{True}$ if and only if $k' = 0, \mathbf{S} = \mathbf{0}$, $\mathbf{D}(e, \alpha, \beta) = 1$ for some $e \in p(T')$ where $\beta \leq d(j, e) \leq (1 + \epsilon)\beta$, and the remaining $\mathbf{D}$ entries are 0.

3. $T' = \{c\}$ where c is a facility node. Then either it is closed, in which case we must have $k' = 0$ and $\mathbf{S} = \mathbf{D} = \mathbf{0}$, or it is open, in which case we must have $k' = 1$, $\mathbf{D} = \mathbf{0}$, and

$$\sum_{\substack{e \in p(T') \\ \alpha, \beta \in \mathcal{C}}} \mathbf{S}(e, \alpha, \beta) \cdot (\alpha + \beta + d_e) \leq (1 + \epsilon) \cdot B.$$

   That is, the total perceived distance of all arms accounted for by the surplus vector should not exceed $(1 + \epsilon) \cdot B$.

Inductive Step

Suppose T^1, T^2 are the two children of a non-leaf node. At a high level, to determine if $A[T', k', \mathbf{S}, \mathbf{D}] = \textsc{True}$, we simply have to guess which arms represented by the surplus and deficiency vectors start/end from T^1 and which ones start/end from T^2, and also how many arms of various types reach between T^1 and T^2 (in which case $e_{T'}$ would be the highest portal edge they cross).

More specifically, $A[T', k', \mathbf{S}, \mathbf{D}] = \textsc{True}$ if and only if there are values $0 \leq k^1, k^2 \leq k$ and integer vectors $\mathbf{S}^1, \mathbf{D}^1, \mathbf{S}^2, \mathbf{D}^2$ corresponding to the subproblems T^1, T^2, respectively, such that

- $A[T^i, k^i, \mathbf{S}^i, \mathbf{D}^i] = \textsc{True}$ for both $i = 1, 2$
- $k^1 + k^2 = k'$
- $\mathbf{S}^1(e_{T'}, \alpha, \beta) = \mathbf{D}^2(e_{T'}, \alpha, \beta)$ and $\mathbf{S}^2(e_{T'}, \alpha, \beta) = \mathbf{D}^1(e_{T'}, \alpha, \beta)$. In words, the subproblems agree on the arms that have $e_{T'}$ as their highest edge.
- For each $\alpha, \beta \in \mathcal{C}$ and each $e \in p(T')$, $\mathbf{S}^1(e, \alpha, \beta) + \mathbf{S}^2(e, \alpha, \beta) = \mathbf{S}(e, \alpha, \beta)$ and $\mathbf{D}^1(e, \alpha, \beta) + \mathbf{D}^2(e, \alpha, \beta) = \mathbf{D}(e, \alpha, \beta)$. In words, the arms that enter/exit each subproblem but do not cross between T^1 and T^2 form a partition of the arms exiting T'.

Verifying correctness of this recurrence is straightforward. Note that the number of subproblems is $n^{O(\epsilon^{-1} \cdot \log n)}$ because there are $O(n)$ subtrees in $\mathcal{T}$, at most $k + 1 \leq n + 1$ possible values for k', and each of the $O(\epsilon \cdot \log n)$ components of the surplus and deficiency vectors is an integer between 0 and n. Similar counting shows the number of subproblems that have to be checked when computing $A[T', k', \mathbf{S}, \mathbf{D}]$ is $n^{O(\epsilon \cdot \log n)}$.

Theorem 4.3 *For any constant $0 < \varepsilon \leq 1$, there is a $(1+\varepsilon)$-approximation algorithm for* MLkFL *on tree metrics that runs in quasi-polynomial time.*

5 A constant-factor approximation algorithm for MLkFL in star metrics

A star is a rooted tree of height 1 and a star metric is the shortest-path metric completion of a weighted star. We now consider MLkFL in star metrics, and in the more-general setting where each client j has a demand D_j that may be split across various open facilities. The load of a facility i is now defined as $\sum_j x_{ij} d(i,j)$ where $x_{ij} \in \mathbb{R}_{\geq 0}$ is the amount of j's demand that is served by i. We also consider the *integer-splittable version* where the demands are integers and may be split *integrally* across the open facilities. (As noted earlier, with a star metric, we cannot reduce (even) the integer-demand setting to the unit-demand setting by creating colocated clients as this will destroy the star topology.)

We devise a 12-approximation algorithm for this problem, which also yields a 14-approximation for the integer-splittable version. At a high level our approach is similar to the one used to obtain the PTAS for line metrics. We again "guess" the optimal value B. We argue via a slightly different uncrossing technique that if $B \geq L^{opt}$, then there exists a well-structured fractional solution with maximum load at most $6B$, and use DP to obtain a fractional solution with maximum load at most $12B$. This can then be converted to an integer-splittable assignment with maximum load at most $14B$ using the GAP-rounding algorithm, since, for the integer-splittable version, it is easy to ensure via some preprocessing that $d(i,j) \leq 2B$ for every facility i and client j. We can now combine this with a binary-search procedure to find the "right" B; this yields the 12-approximation algorithm (and a 14-approximation for the integer-splittable version).

Let r be the root of the star graph defining the star metric, V denote the set of all leaf nodes, and let $d_i = d(i,r)$ for leaf i. We may assume that $r \notin \mathcal{F} \cup \mathcal{C}$ since we can add an extra leaf with distance zero to r. Number the nodes of V from 1 to n so that $d_1 \leq d_2 \leq \cdots \leq d_n$. Let D_j be the demand of client $j \in \mathcal{C}$. We often refer to a pair (i,j), where $i \in \mathcal{F}, j \in C$, as an arm.

Theorem 5.1 *There is a 12-approximation algorithm for* MLkFL *on star metrics with non-uniform demands, and a 14-approximation algorithm for the integer-splittable version.*

Proof : Let B be our current guess of the optimal value. In Section 5.1, we show that if $B \geq L^{opt}$, then there exist k facilities, and a well-structured fractional assignment of clients to these facilities of cost (i.e., maximum load) at most $6B$. In Section 5.2, we devise a dynamic programing approach that finds k facilities and a well-structured fractional assignment of clients to these facilities of cost at most $12B$ provided there is such a solution of cost at most $6B$. In the integer-splittable version, we may assume that $d_i \leq B$ for all $i = 1, \ldots, n$. Otherwise, if $d_i > B$, then no client may assign any demand to i (if $i \in \mathcal{F}$); also, if $D_i > 0$, then we must have $i \in \mathcal{F}$ for the instance to be feasible, and all of D_i must be served by i. Thus, we can remove i from V, and in the latter case, decrease k by 1, and proceed with the smaller instance.

Combining these ingredients, we can either certify that $B < L^{opt}$, or find k facilities and a fractional assignment of clients to these facilities that has maximum load at most $12B$. In the integer-splittable version, the above preprocessing ensures that $d(i,j) \leq 2B$ for every facility i and client j. Thus, using Theorem 3.6, we can round the fractional assignment (with maximum load at most $12B$) to an integer solution while increasing the maximum load by at most $\max_{i \in \mathcal{F}, j \in \mathcal{C}} d(i,j) \leq 2B$.

Finally, we discuss the binary-search procedure used to find a suitable B. We can detect if $L^{opt} = 0$, so assume otherwise. It is easy to find an upper bound UB on L^{opt} such that $\log UB$ is polynomially bounded in the input size. In the integer-splittable version, since L^{opt} is an integer, it is straightforward to do binary search in the interval $[0, UB]$ to find the smallest integer B (which must be at most L^{opt}) for which the above procedure returns a solution with maximum load at most $14B$. With non-integer demands (and fractional assignments), the binary search is somewhat more involved. We can obtain an integer K such that such that

L^{opt} is a rational number with denominator at most K and $\log K$ is polynomially bounded in the input size. This is because given the set of open facilities in an optimal solution, L^{opt} is the solution to a polynomial-size LP, and the optimal value to a rational LP has bit size polynomial bounded in the size of the LP. Now we perform binary search in the interval $[0, UB]$ to find integers A and $U = A + 1$ such that $L^{opt} > \frac{A}{2K^2}$ and, for $B = \frac{U}{2K^2}$, the above procedure returns a solution with maximum load at most $12B$. Now, using continued fractions (see [**?**]), we can find the rational number, p, closest to U having denominator at most K, in time polynomial in the bit sizes of U and K. We claim that if $L^{opt} < \frac{U}{2K^2}$, then $p = L^{opt}$ since $|p - L^{opt}| \leq 2\left|L^{opt} - \frac{U}{2K^2}\right| < \frac{1}{K^2}$, and any two distinct rational numbers having denominator at most K have a spacing of at least $\frac{1}{K^2}$. So if $p \in \left(\frac{L}{2K^2}, \frac{U}{2K^2}\right)$, then we run our procedure with $B = p$; if the procedure returns a solution for this value of B, we return this solution, otherwise we return the solution found for $B = \frac{U}{2K^2}$. ■

5.1 A well-structured near-optimal solution

We show that if $B \geq L^{opt}$, then there exists a fractional assignment satisfying various nice structural properties, which will then enable us to find such a solution via DP (Section 5.2). Let C_B be the set of open facilities in some integer-splittable solution having maximum load at most B. Consider the following LP. For notational convenience, we set $D_j = 0$ if $j \notin \mathcal{C}$, and think of every node as a client (but with potentially 0 demand).

$$\begin{aligned}
\text{minimize} \quad & \sum_{i \in C_B} \sum_{j \in \mathcal{C}} d(i,j) \cdot x_{ij} && && \text{(S-P)}\\
\text{subject to} \quad & \sum_{i \in C_B} x_{ij} = D_j && \forall j \in \mathcal{C} \\
& \sum_{j \in V} d(i,j) \cdot x_{ij} \leq B && \forall i \in C_B \\
& x_{ii} = D_i && \forall i \in C_B \\
& x_{ij} \geq 0 && \forall i \in C_B, j \in \mathcal{C}.
\end{aligned}$$

Given a solution x to (S-P), we say that arms (i, j') and (i', j) *cross* in x if $x_{ij'} \cdot x_{i'j} > 0$ and $d(i,j) \cdot d(i',j') < d(i',j) \cdot d(i,j')$. Note this is different from the notion of crossing in our PTAS for line metrics, but it can be seen to generalize it; that is, one can check that if two arms (i, j'), and (i', j) on the line cross according to the definition in Section 3.1 then $d(i,j) \cdot d(i',j') < d(i',j) \cdot d(i,j')$.

We know that (S-P) is feasible. We prove that the optimal solution to (S-P) does not have any crossing arms in its support.

Lemma 5.2 *The optimal solution to* (S-P) *does not have any crossing arms in its support.*

Proof : Let x be an optimal solution to (S-P) Suppose (i, j') and (i', j) cross in x. If $d(i,j) = 0$ then simply update x by moving all of j's demand to i. Similarly, if $d(i',j') = 0$ then move all of the demand of j' to i'. In both cases, the objective value of x decreases, which is a contradiction. So suppose that $0 < d(i,j) \cdot d(i',j')$.

For some $\epsilon, \epsilon' > 0$ to be specified shortly, we create a new assignment x' that agrees with x in all center-client pairs except that:

- $x'_{ij} = x_{ij} + \epsilon, \quad x'_{i'j} = x_{i'j} - \epsilon.$
- $x'_{i'j'} = x_{i'j'} + \epsilon', \quad x'_{ij'} = x_{ij'} - \epsilon'.$

It must be that $d(i,j) < d(i,j')$ or $d(i',j') < d(i',j)$ so assume, without loss of generality, that $d(i,j) < d(i,j')$. We chose ϵ, ϵ' such that the load at i does not change, so $\epsilon \cdot d(i,j) = \epsilon' \cdot d(i,j')$ and so that either $x'_{i'j} = 0$ or $x'_{ij'} = 0$ while the other remains nonnegative. The change in the load of i' as well as the change in objective value is given by

$$\epsilon' \cdot d(i',j') - \epsilon \cdot d(i',j) = \epsilon \left(\frac{d(i,j) \cdot d(i',j')}{d(i,j')} - d(i',j) \right)$$

which is nonpositive because $d(i,j) \cdot d(i',j') < d(i,j') \cdot d(i',j)$. This yields a contradiction. ■

Observe that the above uncrossing property is stronger then the uncrossing that we achieved for line metrics, where we only ensured that small arms do not cross. Figure 4 illustrates all the (non-symmetric) cases that count as crossing. The figures on the right show the result after modifying x as described in the above proof. In each picture, at most one of the dotted arrows has a non-zero x_{ij} value. We use c_1, c_2 for the location of centers i, i' and v_1, v_2 for the location of clients j', j to be consistent with the notation used in the PTAS for line-metrics.

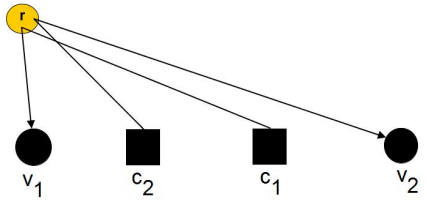
Definition 5.3 A fractional solution x to (S-P) is *well-structured* if we can partition $V = \{1, \ldots, n\}$ into consecutive subsequences $V_1, V_2, \ldots, V_m$ such that:

- For each V_a and each $j \in V_a \cap \mathcal{C}$, we have $x_{ij} = 0$ for $i \notin V_a$. That is, each client is completely served within its partition.
- For each V_a, at least one of the followings hold:
  1. $|C_B \cap V_a| = 1$
  2. $x_{ij} = 0$ for all $j \in V_a \cap \mathcal{C}$ and $i < j$ (clients are only satisfied by centers to the right)
  3. $x_{ij} = 0$ for all $j \in V_a \cap \mathcal{C}$ and $i > j$ (clients are only satisfied by centers to the left)

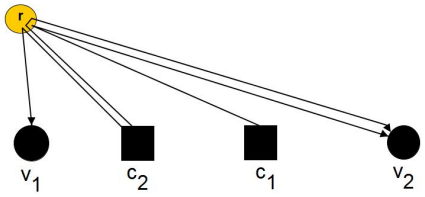
Lemma 5.4 *There is a well-structured fractional solution x to* (S-P) *with maximum load at most* $6B$.

Proof : Let x^0 be an optimal solution to (S-P). So x^0 has maximum load at most B, and by Lemma 5.2, it does not have any crossings. We start with x^0 and in each step modify given solution such that the maximum load is not increased by more than a constant factor. We use x^r to denote the output of Step r, for $1 \leq r \leq 3$.

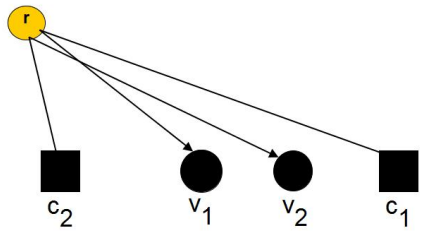
- **Step 1) Ensuring all clients are served in only one direction.**
  We initialize x^1 to x^0. For any client $j \in C_B$, all the demand at j can be satisfied by the collocated center so we can assume $x^1_{ij} = 0$ for $i \neq j$. For any other client $j \notin C_B$, either $\sum_{i<j} x^1_{ij} \geq \frac{D_j}{2}$ or $\sum_{i>j} x^1_{ij} \geq \frac{D_j}{2}$. Suppose the former is true (the latter is similar). Then we simply set $x^1_{ij} = 0$ for $i > j$ and scale the x^1_{ij} with $i < j$ uniformly until they sum to D_j again. After doing this for all clients, we have that x^1_{ij} at most doubles for each center-client pair so the maximum load is at most $2B$. Note that any non-zero coordinate in x^1 is a non-zero coordinate in x^0 as well, so x^1 does not have any crossings either.
- **Step 2) Ensuring all centers either serve clients only to the left or only to the right, or form their own consecutive partition.**
  First, we initialize x^2 to x^1. Let i be any center in C_B with $x^2_{ip}, x^2_{iq} > 0$ for two clients $p < i < q$ (note that $x^1_{ip}, x^1_{iq} > 0$). If there is no such center, then this step is done. We will modify the assignment to i and, perhaps, some nearby centers and then form a consecutive subsequence of V whose only center is i.



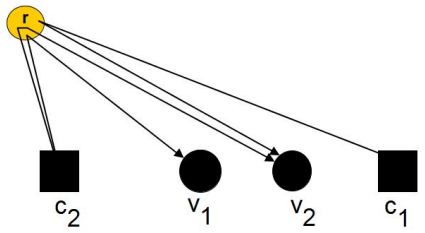
(a) Case(1) :c_1 and c_2 between v_1 and v_2.



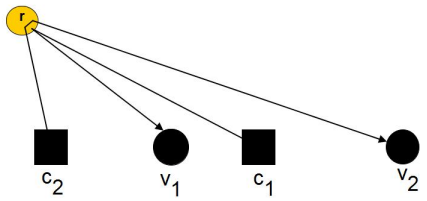
(b) Coverage after fixing the intersection in Case (1).



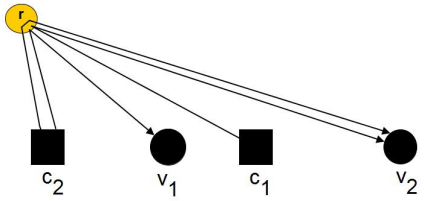
(c) Case (2): v_1 and v_2 between c_1 and c_2.



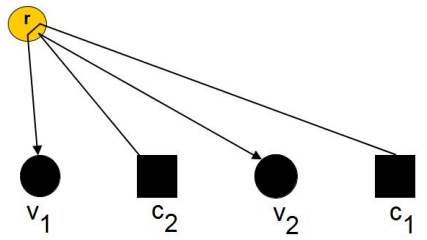
(d) Coverage after fixing the intersection in Case (2).



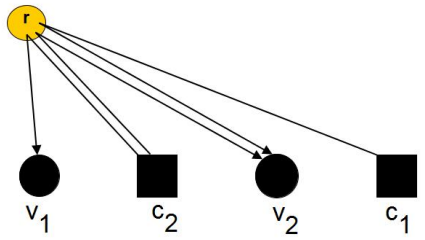
(e) Case (3): c_1 and c_2 on the left side of v_1 and v_2.



(f) Coverage after fixing the intersection in Case (3).



(g) Case (4): c_1 and c_2 on the right side of v_1 and v_2.



(h) Coverage after fixing the intersection in Case (4).

Figure 4: Fixing the crossing of two arms.

Consider the rightmost center $i_L \in C_B$ such that $i_L < i$, see Figure 5. If there is no center to the left of i, then the operations in this paragraph are skipped. Otherwise we update the assignment x^2 in the following way: For any client $j < i_L$ with $x^2_{ij} > 0$, we update $x^2_{i_L j} \leftarrow x^2_{i_L j} + x^2_{ij}$ and $x^2_{ij} \leftarrow 0$. Note that $x^2_{i_L i_L} = D_{i_L}$ by (S-P) constraint. Note that this modification does not introduce any crossings in x^2 as each client j is now assigned to a closer location. Moreover, now any client $j < i$ with $x^2_{ij} > 0$, has to be on the right of i_L, i.e., $i_L < j$.

Similarly, if there is a leftmost center $i_R \in C_B$ with $i_R > i$ then move all assignment x^2_{ij} with $j > i_R$ to i_R. Again after these modifications, no new crossing is introduced, and any client $j > i$ with $x^2_{ij} > 0$, has to be on the left of i_R, i.e., $j < i_R$.

Let j_L be the leftmost client with $x^2_{ij_L} > 0$; if no such client exists define $j_L = n + 1$. Similarly, let j_R be the rightmost client with $x^2_{ij_R} > 0$; if no such client exist let $j_R = 0$. Define interval V_a to be $j_1, j_2, \cdots, j_m$ where $j_1 = \min(j_L, i)$ and $j_m = \max(i, j_R)$ (See Figure 5). Note that partition V_a

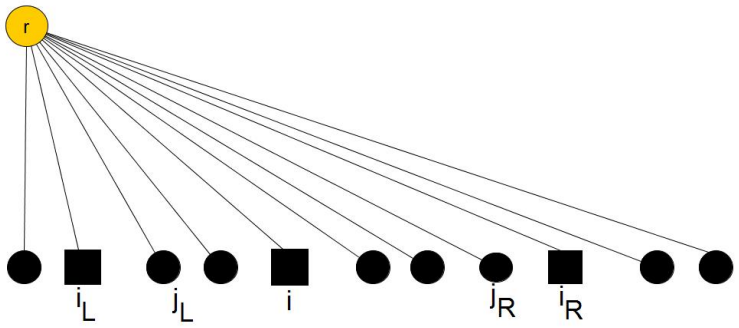


Figure 5: Moving client assignments. Black solid circles denote clients with $x_{ij} > 0$.

satisfies $|C_B \cap V_a| = 1$. Also, no clients outside of V_a are assigned to i in x^2. We claim that all clients in V_a are completely assigned to i in x^2. Let j be an arbitrary client in V_a. Without loss of generality assume $j < i$ (the other case is similar). Note that in this case $j_1 \neq i$, and so $j_1 = j_L < n + 1$. Note that i in x^1 serves some $q > i$ (initial condition for picking i). We have two possible cases:

- $j = j_L$. By definition of j_L, $x^2_{ij_L} > 0$. By step 1, we know that j_L is served in one direction which has to be right as i is on the right of j_L. Suppose facility $i' > i$, serves j_L. This gives us a contradiction as $x^1_{i'j_L} \cdot x^1_{iq} > 0$ and arms (i, q) and (i', j_L) cross (Case (1) or Case (4) depending on the position of q with respect to i', see Figure 4).
- $j > j_L$. First, we show j cannot be served by a facility $i' < i$. Let us assume the contrary. By definition of i_L, we have $i' \leq i_L$, and hence arms (i', j) and (i, j_L) cross which gives us a contradiction as $x^1_{i'j} \cdot x^1_{ij_L} > 0$ (Case (2) in Figure 4c). Next, j cannot be served by facility $i' > i$ as this implies that arms (i, q) and (i', j) cross for solution x^1 while $x^1_{i'j} \cdot x^1_{iq} > 0$ (Case (1) or Case (4) depending on the position of q with respect to i', see Figure 4).

Note that partition V_a satisfies $|C_B \cap V_a| = 1$ and that all clients in V_a are completely assigned to the sole center in V_a. Removing V_a from V effectively divides the instance into two subinstances. We only note that $x^2_{i'j'} = 0$ for any $j' < i < i'$ or $i' < i < j'$. Otherwise, if (say) $j' < i < i'$ has $x^1_{i'j'} > 0$ then this would cross arm (i, q) that was assigned to i in x^1 which gives us a contradiction.

We claim the maximum load after performing the second step has increased by at most $4B$. The only times the load increases are when some centers of the form i_L or i_R have some x^2_{ij} reassigned to them. Each center i' can be some center of the form i_L only once and of the form i_R only once. Moreover, the load of i' cannot be increased in both cases. Suppose the contrary, i.e., i' is i_L for some center i_0 and i_R for some center i_1 where there exist clients $j_0 < i'$ and $j_1 > i'$ with $x^1_{i_0 j_0}, x^1_{i_1 j_1} > 0$. Since the arms (i_0, j_0) and (i_1, j_1) cross, we get a contradiction. So at most one center can increase the load of i', therefore it remains to show that the increase in the load if i' is i_L or i_R of some center is at most $4B$.

First assume that i' is of the form i_L for some facility i. Since $d(i', j) < d(i, j)$ for each client $j \neq i$, the load of i' after the process is increased by the portion of load of i that corresponds to serving clients $j < i_L$ which is at most $2B$. Now assume i' is of the form i_R for some facility i. Note that any client j moved to i' has $d(i, j) < d(i', j)$ but $d(i', j) < 2d(i, j)$ as $j > i'$. So the load increase on i' is at most twice the load on i due to clients $j > i'$; therefore, the load increase is at most $4B$.

- **Step 3) Dividing the remaining instance.**
  First initialize x^3 to x^2. Now each $j \in \mathcal{C} \setminus C_B$ assigns all demand either completely to the left or completely to the right. Similarly, any $i \in C_B$ collects demand either completely from the left or

completely from the right. Say that $j \in C_B \cup \mathcal{C}$ "goes left" if $j \notin C_B$ and $x^3_{ij} > 0$ only for $i < j$ **or** $j \in C_B$ and $x^3_{jj'} > 0$ only for $j' \geq j$. If j does not "go left" then say it "goes right". Now V naturally breaks up into maximal consecutive intervals of nodes, each of which only includes clients that "go left" or "go right". These form the remaining partitions V_a.

The only thing left to note is that a client is completely served within its partition. Suppose $j \in V_a$ and that j "goes left". Either V_a is the first partition, or the preceding partition $V_{a'}$ "goes right". Since $V_{a'}$ is not a singleton (that was taken care of at the end of step 2), then there is some $j' \in V_{a'}$ with $x^3_{i'j'} > 0$ for some $i' \in C_B \cap V_{a'}, i' \neq j'$. Thus, j cannot assign any demand to a client to the left of V_a since this assignment would cross with the assignment of j'.

Now let $x = x^3$ which is a desired solution satisfying the condition of the lemma. ■

5.2 A dynamic-programming algorithm for finding a well-structured solution

We describe the dynamic programming approach in two steps. First, for any subsequence $V' = \{j_L, \ldots, j_R\}$ of V, and any $1 \leq p \leq k$ we describe a boolean value $I(V', p)$ that is true if and only if one of the following is true.

1. $p = 1$ and there is some facility $i \in V' \cap \mathcal{F}$ such that assigning each client from V' to i places load at most $6B$ on i.

2. There is a set $C' \subseteq V' \cap \mathcal{F}$ of p facilities, and a fractional assignment $(x_{ij})_{i \in C', j \in V'}$ such that for every $j \in V'$ we have $x_{ij} = 0$ for $i < j$ and $\sum_{i \in C'} x_{ij} = D_j$, and for every $i \in C'$ we have $d_i \cdot \sum_{j \in V'} x_{ij} \leq 6B$.

3. There are some p locations C' and a fractional assignment $(x_{ij})_{i \in C', j \in V'}$ such that for every $j \in V'$ we have $x_{ij} = 0$ for $i > j$ and $\sum_{i \in C'} x_{ij} = D_j$, and for every $i \in C'$ we have $\sum_{j \in V'} d_j \cdot x_{ij} \leq 6B$.

If we can compute $I(V', p)$ for all (V', p) tuples (as well as the solution that generates it), then we claim that we are done. Observe that if $I(V', p) = \texttt{true}$ when $p > 1$, then the fractional assignment x corresponding to $I(V', p)$ induces a maximum load of $12B$ on the centers opened from V'. If $I(V', p)$ is true due to the second condition, then this is because if $x_{ij} > 0$, then $d(i, j) \leq 2d_i$, and we have $d_i \cdot \sum_{j \in V'} x_{ij} \leq 6B$. Similarly, if $I(V', p)$ is true due to the third condition, then $x_{ij} > 0$ implies that $d(i, j) \leq 2d_j$, and we have $\sum_{j \in V'} d_j \cdot x_{ij} \leq 6B$.

Now it is a simple matter to determine, using another dynamic program, how to partition V into consecutive intervals $V_1, \ldots, V_m$ with positive integers $p_1, \ldots, p_m$ summing to k such that $I(V_i, p_i) = \texttt{true}$ for each $1 \leq i \leq m$.

To wrap up the proof, we describe how to compute $I(V', p)$. We can associate a table for each case in the definition of $I(V', p)$. Let $T_1(V')$ be the table corresponding to the first case. There are $O(n)$ possible choices for the center, so each table entry can be computed in $O(n)$ time and there are $O(n^2)$ table entries.

For the second case, we consider the table $T_2(V', p)$. In order to compute the entries of T_2, we use an auxiliary table $f(i, p')$ for $j_L \leq i \leq j_R, 0 \leq p' \leq p$. $f(i, p')$ is the minimum possible excess demand $\sum_{j \leq i} \left(D_j - \sum_{i' \in C'} x_{i'j}\right)$ among all ways of choosing p' centers C' in $\{j_L, \ldots, i\}$ and fractionally assigning up to D_j units of demand of each client $j \leq i$ to centers in C' such that: (i) no center $i' \in C'$ is assigned more than $6B/d_{i'}$ units of demand from clients $j < i'$, and (ii) $x_{i'j} = 0$ if $i' < j$.

The base cases with $i = j_L$ are easy: $f(i, 0) = D_i$ and $f(i, 1) = 0$ if $i \in \mathcal{F}$; we set $f(i, p') = \infty$ if $p' > 1$, or ($p' > 0$ and $i \notin \mathcal{F}$). Also, for $i > j_L$ but $p' = 0$ we have $f(i, p') = f(i - 1, p') + D_i$. Otherwise,

if $i > j_L$ and $p' > 0$ we have

$$f(i,p') = \begin{cases} \min\{\max\{0, f(i-1,p'-1) - 6B/d_i\}, f(i-1,p') + D_i\}; & \text{if } i \in \mathcal{F} \\ f(i-1,p') + D_i & \text{otherwise.} \end{cases}$$

The first term in the min says that if we open i, then we assign as much leftover demand that we can. The second term says that if we do not open i then all of the demand at i must go to the right of i. Once we compute this, we set $T_2(V',p)$ to `true` if and only if $f(j_R,p) = 0$.

For the last case, we consider a similar dynamic programming algorithm in a "right-to-left" manner, except we are concerned with the minimum value of $\sum_{j \geq i} d_j \cdot \left(D_j - \sum_{i' \in C'} x_{i'j}\right)$. We associate the table $T_3(V',p)$ for this case, and we use the auxiliary table $g(i,p')$ for $j_L \leq i \leq j_R, 0 \leq p' \leq p$. $g(i,p')$ is the minimum possible excess load $\sum_{j \geq i} d_j \left(D_j - \sum_{i' \in C'} x_{i'j}\right)$ among all ways of choosing p' centers C' in $\{i, \ldots, j_R\}$ and fractionally assigning up to D_j units of demand of each client $j \geq i$ to centers in C' such that: (i) no center $i' \in C'$ is assigned a load more than $6B$, and (ii) $x_{i'j} = 0$ if $i' > j$.

The base cases with $i = j_R$ are easy: $g(i,0) = d_i \cdot D_i$ and $g(i,1) = 0$ if $i \in \mathcal{F}$; we set $g(i,p') = \infty$ if $p' > 1$, or ($p' > 0$ and $i \notin \mathcal{F}$). Also, for $i < j_R$ but $p' = 0$ we have $g(i,p') = g(i-1,p') + d_i \cdot D_i$. Otherwise, if $i < j_R$ and $p' > 0$ we have

$$g(i,p') = \begin{cases} \min\{\max\{0, g(i-1,p'-1) - 6B\}, g(i-1,p') + d_i \cdot D_i\}; & \text{if } i \in \mathcal{F} \\ g(i-1,p') + d_i \cdot D_i & \text{otherwise.} \end{cases}$$

The first term in the min says that if we open i, then we assign as much leftover load that we can. The second term says that if we do not open i then all of the demand at i must go to the left of i. Once we compute this, we set $T_3(V',p)$ to `true` if and only if $g(j_L,p) = 0$.

Finally, once all of the $T_1(V')$, $T_2(V',p)$ and $T_3(V',p)$ are computed, we can set $I(V',1) = T_1(V')$ and $I(V',p) = T_2(V',p) \vee T_3(V',p)$ for $p > 1$.

6 Hardness results and integrality-gap lower bounds

We now present various hardness and integrality-gap results. We prove that MLkFL is strongly *NP*-hard on line metrics (Theorems 6.1). We also demonstrate that a natural configuration-style LP has an unbounded integrality gap (Theorem 6.2).

Theorem 6.1 *Minimum-load k-facility location is strongly NP-hard even in line metrics.*

Proof : We reduce from 3-partition, where we are given $n = 3k$ integers $b_1, \ldots, b_n$ and a bound B such that $\sum_{i=1}^{n} b_i = kB$. The goal is to partition the integers into k groups such that the sum of the integers in any group is at most B. It is NP-complete to determine if there is a feasible solution, even when $b_i \leq 2^{16}n^4$ and $\frac{B}{4} < b_i < \frac{B}{2}$ for each i (e.g. [8]). In particular, any feasible solution will have precisely three integers in each group of the partition.

We create an instance of MLkFL on the line by creating two groups of clients. First, for each point $p \in \{-\frac{k-1}{3k}, -\frac{k-2}{3k}, \ldots, -\frac{1}{3k}, 0\}$ we place $3k^2(B+1)$ clients at p. Next, for each integer $b_i, 1 \leq i \leq n$, we add a single client at position b_i. Let N be the number of clients in the resulting instance and notice that all values have values bounded by a polynomial in N. The claim is that there is a solution with cost $B + \frac{k-1}{k}$ if and only if the 3-partition problem is a **yes** instance.

First, suppose there is a partition of the integers $b_1, \ldots, b_n$ into k groups $G_1, \ldots, G_k$ such that the sum of the integers in any group G_i is B. For each $1 \leq i \leq k$ we create a star with center at $-\frac{i-1}{3k}$, assign all clients located at this center to this star, and also assign the clients in group G_i to this star. The only clients

that move some positive distance to the center of the star are those from the group G_i, and they move a total distance of $B + \frac{i-1}{k} < B + \frac{k-1}{k}$.

Conversely, suppose there is a solution with maximum load at most $B + \frac{k-1}{k}$. First, we claim that every point of the form $-\frac{i}{3k}, 0 \leq i < k$ must be the center of a star. Otherwise, the $3k^2(B+1)$ clients at this location must be assigned to other stars. The minimum distance each of these client travels is $\frac{1}{3k}$ and one of the open centers receives at least $3k(B+1)$ of these clients, so its load is at least $B + 1 > B + \frac{k-1}{k}$. Therefore, the centers are at locations $-\frac{i}{3k}, 0 \leq i < k$.

Since $\frac{B}{4} < b_i < \frac{B}{2}$, then every star must contain exactly three clients corresponding to integers $b_1, \ldots, b_n$ in the 3-partition instance. Without loss of generality, say b_1, b_2, b_3 are the three integers in some star. The total distance they travel lies between $b_1 + b_2 + b_3$ and $b_1 + b_2 + b_3 + \frac{k-1}{k}$ so $b_1 + b_2 + b_3 \leq B$. Therefore, if we let G_i be the clients corresponding to integers $b_1, \ldots, b_n$ that are in the star with center $-\frac{i-1}{3k}$ for each $1 \leq i \leq k$, then $G_1, \ldots, G_k$ is a feasible solution to the 3-partition problem. ■

Integrality-gap lower bound. Let $\left(\mathcal{F}, \mathcal{C}, d, k\right)$ be an MLkFL instance. Given a candidate "guess" B of the optimal value, we can consider the following LP-relaxation of the problem of determining if there is a solution with maximum load at most B. We propose the following linear programming for the MLkFL . For each facility $i \in \mathcal{F}$, define $\mathcal{S}(B;i) := \{C \subseteq \mathcal{C} : \sum_{j \in C} d(i,j) \leq B\}$ to be the set of all stars centered at i that induce load at most B at i. We will often refer to a star in $\mathcal{S}(B;i)$ as a configuration. (Note that $\mathcal{S}(B;i)$ contains $\emptyset$.) Our LP Will be a *configuration-style LP*, where for every facility i and star $C \in \mathcal{S}(B;i)$, we have a variable denoting if star C is chosen for facility i. This yields the following natural feasibility LP.

$$\left.\begin{aligned} \sum_{i \in \mathcal{F}} \sum_{C \in \mathcal{S}(B;i): j \in C} x(i,C) &\geq 1 \qquad \forall j \in \mathcal{C} \qquad (1) \\ \sum_{C \in \mathcal{S}(B;i)} x(i,C) &\leq 1 \qquad \forall i \in \mathcal{F} \qquad (2) \\ \sum_{i \in \mathcal{F}} \sum_{C \in \mathcal{S}(B;i)} x(i,C) &\leq k \qquad (3) \\ x &\geq 0. \end{aligned}\right\} \quad \text{(P)}$$

Constraint (1) ensures that each client belongs to some configuration, and constraints (2) and (3) ensure that each facility belongs to at most one configuration, and that there are at most k configurations. We show that there is an MLkFL instance on the line metric, where the smallest value B_{LP} for which (P) is feasible is smaller than the optimal value by an $\Omega(k/\log k)$ factor; thus, the "integrality gap" of (P) is $\Omega(k/\log k)$. Moreover, in this instance, the graph containing the (i,j) edges such that $d(i,j) \leq B_{\mathrm{LP}}$ is connected.

Theorem 6.2 *The integrality gap of* (P) *is* $\Omega(k/\log k)$ *even for line metrics.*

Proof : Assume for simplicity that k is odd. Consider the following simple MLkFL instance. We have $\mathcal{F} = \{a_1, b_1, a_2, b_2, \ldots, a_m, b_m\}$, where $2m = k+1$. These facilities are located on a line as shown in Figure 6, with the distance between any two consecutive nodes being $T/2$. There are $n = 2k$ clients colocated with each facility. Let A_i (respectively B_i) denote the set of clients located at a_i (respectively b_i) for $1 \leq i \leq m$.

There is a feasible solution to (P) with $B = T$. For all $i = 1, \ldots, m$, we set $x(a_i, A_i \cup \{j, j'\}) = \frac{k}{(k+1)\cdot\binom{n}{2}}$ for all $j, j' \in B_i$; note that all these configurations lie in $\mathcal{S}(T;a_i)$. Similarly, we set $x(b_i, B_i \cup \{j, j'\}) = \frac{k}{k+1\cdot\binom{n}{2}}$ for all $j, j' \in A_i$. It is easy to verify that x is a feasible solution. It is clear that constraints (2) and (3) hold since every facility belongs to exactly $\binom{n}{2}$ configurations. Consider a client $j \in A_i$. j is covered to an extent of $\frac{k}{k+1}$ by the $\binom{n}{2}$ configurations $\left\{A_i \cup \{k, \ell\}\right\}_{k,\ell \in B_i}$ in $\mathcal{S}(a_i;T)$ and to an extent of

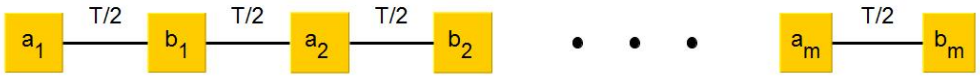


Figure 6: Example showing bad integrality gap for the configuration LP in line metric

$\frac{1}{k+1}$ by the $n-1$ configurations $\left\{B_i \cup \{j,k\}\right\}_{k \in A_i : k \neq j}$. A symmetric argument applies to clients in some B_i set.

Finally, we show that any feasible solution must have maximum load at least $T \cdot \frac{k}{2H_k}$, where $H_r := 1 + \frac{1}{2} + \ldots + \frac{1}{r}$ is the r-th harmonic number, which proves the theorem. In any feasible solution, there is some location v that does not have an open facility. For $i = 1, \ldots, k$, let n_i be the number of clients colocated at v that are assigned to a facility at a location that is i hops away from v; set $n_i = 0$ if there is no such location. Then, $\sum_{i=1}^{k} n_i = n$, and the maximum load L at a facility is at least $\max_{i=1,\ldots,k} \frac{n_i i T}{4}$ since there are at most two facilities that are i hops away from v, and one of them must have at least $\frac{n_i}{2}$ clients assigned to it. Thus, we have $n_i \leq \frac{4L}{iT}$ for all $i = 1, \ldots k$, and so $n \leq \frac{4L}{T} \cdot H_k$, or $L \geq \frac{nT}{4H_k}$. ■

7 An unbounded locality gap for the multi-swap local-search algorithm for MLkFL

A natural local-search heuristic for MLkFL is one where given a current set S of k facilities, we may swap out a facility in S and swap in a facility not in S. More generally, we may consider a p-swap heuristic where we swap out and swap in at most p facilities. Note that given a set of k facilities, one can find the assignment of clients to facilities by solving an instance of the generalized assignment problem [19]. We keep performing such local moves as long as it improves the maximum load of the solution. One can come up with simple examples showing that the *locality gap* of the p-swap heuristic, which is the worst-case ratio between the maximum load at a local optimum and the (global) optimal value, can be arbitrarily large, even on line metrics.

Theorem 7.1 *The locality gap of the p-swap heuristic is unbounded, even on line metrics.*

Proof : Choose any $\epsilon < 1$. Consider $3k$ consecutive locations $s_1, j_1, o_1, s_2, j_2, o_2, \ldots, s_k, j_k, o_k$ located on a line with the $d(s_i, j_i) = 1$, $d(j_i, o_i) = \epsilon$ for all $i = 1, \ldots, k$, and $d(o_i, s_{i+1}) = 1 - \epsilon$ for all $i = 1, \ldots, k-1$. The facility set is $\mathcal{F} = S \cup O$, where $S = \{s_1, \ldots, s_k\}$ and $O = \{o_1, \ldots, o_k\}$, and the client set is $\mathcal{C} = \{j_1, \ldots, j_k\}$.

We claim that the solution S, which has maximum load 1, is a local optimum for the p-swap heuristic, for any $p < k$. Consider a p-swap move where we swap out $s_{i_1}, \ldots, s_{i_p}$ and swap in $o_{\ell_1}, \ldots, o_{\ell_p}$. We claim that this move does not decrease the maximum load, and hence is not an improving move. If it were, then the load of every facility in $F = S \setminus \{s_{i_1}, \ldots, s_{i_k}\} \cup \{o_{\ell_1}, \ldots, o_{\ell_k}\}$ must be strictly less than 1. But then none of the facilities in $S \setminus \{s_1, \ldots, s_k\}$ may be assigned any clients; thus, no facility in S serves any client. Since $p < k$, there is some i such that o_i is not swapped in. Then, j_i is not assigned to s_i or o_i and hence has connection cost larger than 1, which contradicts the assumption that the maximum load is less than 1.

Thus, S is a local optimum, whereas the global optimum is to open the facilities in O and assign each j_i to o_i incurring a maximum load of ϵ. ■

In some sense, the rather simplistic nature of the above example exemplifies the difficulties in applying local search to min-max problems.

8 Conclusion

In this paper we considered the minimum load k-facility location problem, which generalizes the min-max star cover problem studied earlier and presented the first true approximation algorithms for it on line and tree metrics. We devise a PTAS for line metrics, a QPTAS for trees, and an $O(1)$-approximation for star metrics. We also proved that the problem is APX-hard on Euclidean metrics.

Several questions remain open, the most prominent one being to find a true (not bicriteria) approximation for the problem on general metrics. An (perhaps) easier question would be to find such a true approximation algorithm for some structured metrics that are more general than line metrics, e.g., Euclidean metrics, or the shortest-path metric of a family of graphs that is more general than paths. To gain some insight, as an intermediate step, it may be useful to investigate if one could simplify our algorithm for line metrics if one is willing to settle for an $O(1)$-approximation (instead of a PTAS). (We do not know of a simpler approach than our current DP that yields a PTAS.) Finally, obtaining a PTAS for trees is another concrete interesting question.

References

[1] Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. In *Proceedings of IPCO*, 2014.

[2] E.M. Arkin, R. Hassin, and A. Levin. Approximations for minimum and min-max vehicle routing problems. *Journal of Algorithms*, 59(1):1–18, 2006.

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

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

[5] M. Cygan, M.T. Hajiaghayi, and S. Khuller. Lp rounding for k-centers with non-uniform hard capacities. *Arxiv preprint arXiv:1208.3054*, 2012.

[6] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha. Covering graphs using trees and stars. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 24–35, 2003.

[7] G.N. Frederickson, M.S. Hecht, and C.E. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7:178, 1978.

[8] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[9] D. Hochbaum and D. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. *SIAM Journal on Computing*, 17:539–551, 1988.

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

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

[12] Amin Jorati. Approximation algorithms for min-max vehicle routing problems. Master's thesis, University of Alberta, Department of Computing Science, 2013.

[13] M. R. Khani and M. R. Salavatipour. Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. *Algorithmica*, 69:443–460, 2014.

[14] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Symposium on Theory of Computing (STOC)*, 2013.

[15] P. Mirchandani and R. Francis, editors. *Discrete location theory*. Jown Wiley and Sons, 1990.

[16] H. Nagamochi and K. Okada. Approximating the minmax rooted-tree cover in a tree. *Information Processing Letters*, 104(5):173–178, 2007.

[17] R. Ravi. Workshop on Flexible Network Design, 2012. `http://fnd2012.mimuw.edu.pl/qa/index.php?qa=4&qa_1=approximating-star-cover-problems`.

[18] D. Shmoys. The design and analysis of approximation algorithms: facility location as a case study. In S. Hosten, J. Lee, and R. Thomas, editors, *Trends in Optimization, AMS Proceedings of Symposia in Applied Mathematics 61*, pages 85–97. 2004.

[19] D B Shmoys and E Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62(3):461–474, 1993.