# Approximation Algorithms for Minimum-Load $k$-Facility Location

**Sara Ahmadian[*1], Babak Behsaz[2], Zachary Friggstad[2], Amin Jorati[2], Mohammad R. Salavatipour[†2], and Chaitanya Swamy[‡1]**

1 **Department of Combinatorics and Optimization, University of Waterloo**
**Waterloo, ON, Canada, N2L 3G1**
{sahamdian,cswamy}@math.uwaterloo.ca

2 **Department of Computing Science, University of Alberta**
**Edmonton, AB, Canada, T6G 2E8**
{behsaz,zacharyf,jorati,mrs}@ualberta.ca

## 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* (ML$k$FL) 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 ML$k$FL for when $\mathcal{F} = \mathcal{C}$. ML$k$FL is rather poorly understood, and only an $O(k)$-approximation is currently known for ML$k$FL, *even for line metrics*.

Our main result is the *first polynomial time approximation scheme* (PTAS) for ML$k$FL on line metrics (note that no non-trivial true approximation of any kind was known for this metric). Complementing this, we prove that ML$k$FL is strongly *NP*-hard on line metrics. We also devise a quasi-PTAS for ML$k$FL on tree metrics. ML$k$FL turns out to be surprisingly challenging even on line metrics, and resilient to attack by the 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 ML$k$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. A novel aspect of this proof is that we first move to a mixed-integer LP (MILP) encoding 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.

**1998 ACM Subject Classification** F.2.2 Nonnumerical Algorithms and Problems

**Keywords and phrases** approximation algorithms, min-max star cover, facility location, line metrics

**Digital Object Identifier** 10.4230/LIPIcs.APPROX-RANDOM.2014.17

* Supported by the last author's NSERC grant 327620-09.
† Supported by NSERC.
‡ Supported in part by NSERC grant 327620-09, an NSERC Discovery Accelerator Supplement Award, and an Ontario Early Researcher Award.



17th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'14) / 18th Int'l Workshop on Randomization and Computation (RANDOM'14).
Editors: Klaus Jansen, José Rolim, Nikhil Devanur, and Cristopher Moore; pp. 17–33

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

# 1 Introduction

Facility-location (FL) problems have been widely studied in the Operations Research and Computer Science communities (see, e.g., [13] and the survey [16]), 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 hence 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 are given 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* 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 as $\sum_{j \in \mathcal{C}: f(j)=f} d(f, j)$; that is, the load of $f$ is the total connection cost incurred for the clients assigned to it. We call this the *minimum-load $k$-facility location* (ML$k$FL) 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 $d(f, j)$s form a metric.

Despite the extensive amount of literature on facility-location problems, there is surprisingly little amount of work on ML$k$FL and it remains a rather poorly understood problem (see [15]). 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, 9, 8, 3, 12] (where one seeks to minimize the *sum* of the facility loads). No better approximation algorithms are known for ML$k$FL *even on line metrics*, and this was mentioned as an open problem in [15]. The only work on approximation algorithms for this problem that we are aware of is due to Even et al. [6] and Arkin et al. [2], who refer to this problem as *min-max star cover* (where $\mathcal{F} = \mathcal{C}$). [1] Both works obtain *bicriteria* approximation algorithms for ML$k$FL 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. For ML$k$FL 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").

## 1.1 Our Results

We completely resolve the status of min-load $k$-FL on line metrics. As we elaborate below (see "Our Techniques"), ML$k$FL turns out to be surprisingly challenging even on line metrics, and seems resilient to attack by the variety of techniques that have been successfully applied to facility-location problems, including LP-rounding, local search, and primal-dual methods. Our main result is that despite these difficulties, one can devise a polynomial-time approximation

[1] Jorati [10], in his Master's thesis, obtained a preliminary version of some of our current results.

scheme (PTAS) for ML$k$FL on line metrics (Theorem 1). As mentioned earlier, this is the *first* approximation algorithm for ML$k$FL on line metrics that achieves anything better than an $O(k)$-approximation.

We also consider ML$k$FL in tree metrics (Section 4). First, we observe that the quasi-PTAS obtained by Jorati [10] for line metrics extends to yield a quasi-PTAS (QPTAS) for tree metrics (Theorem 9). Next, we consider the special case of star metrics, but in the more-general setting where clients may have non-uniform integer demands $\{D_j\}_{j\in\mathcal{C}}$ and the demand of a client may be split *integrally* between several open facilities. We now define the load of a facility $f$ to be $\sum_j x_{fj}d(f,j)$, where $x_{fj} \in \mathbb{Z}_{\geq 0}$ is the amount of $j$'s demand that is assigned to $f$. We devise a 14-approximation algorithm for ML$k$FL on star metrics with non-uniform demands (Theorem 10). 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), which makes the setting with non-uniform demands strictly more general than the unit-demand setting.

In Section 5, we obtain various computational-complexity and integrality-gap lower bounds for ML$k$FL. Complementing our PTAS, we show (Theorem 11) that ML$k$FL is *strongly* NP*-hard* on line metrics (and hence, a PTAS is the best approximation that one can hope to achieve in polytime unless $P = NP$). We also show that ML$k$FL is $APX$-hard in the Euclidean plane (Theorem 12). Finally, we justify our comment about the difficulty of tackling ML$k$FL via the various LP-based methods developed for facility-location problems by showing that even a configuration-style LP-relaxation for ML$k$FL—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 13). Note that the configuration LP is stronger than the natural LP-relaxation for ML$k$FL. 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.

## 1.2 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 ML$k$FL (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 ML$k$FL 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 ML$k$FL makes it difficult to exploit this. In particular, one can come up with simple examples where a $k$-median style multi-swap local-search does not yield any bounded approximation ratio even for line metrics. Given these difficulties, one needs to find new venues of attack for ML$k$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). Namely, the collection of intervals corresponding to "small" client assignments forms a laminar family. We prove this by "fractionally uncrossing" the small client assignments while preserving the loads at each facility, so the resulting fractional assignment does not contain large strictly fractional assignments. This solution is then rounded using the rounding algorithm of [17] for the *generalized assignment problem* (GAP), and this rounding procedure preserves the laminarity property for small assignments.

Second, we show in Section 3.2 that these structural properties enable one to find such a structured solution via dynamic programming (DP). Roughly speaking, the DP pieces together solutions to subproblems in a way that corresponds to the tree-like structure of a laminar family. To handle the unstructured large client assignments, the DP carries enough information about how large clients cross the boundary of the subproblem being considered.

## 1.3 Related Work

There is a wealth of literature on facility-location problems (see, e.g., [13, 16]); we limit ourselves to the work that is relevant to ML$k$FL. As mentioned earlier, Even et al. [6] and Arkin et al. [2] are the only two previous works that study ML$k$FL (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, ML$k$FL 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 [14]), 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 [11]. 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.

For ML$k$FL 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 ML$k$FL, even with non-unit demands, can be reduced to the makespan-minimization problem on parallel machines while losing a factor of 2.[2] Since the latter problem admits a PTAS [7], this yields a $(2+\epsilon)$-approximation algorithm for ML$k$FL 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 14-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, ML$k$FL can also be motivated and viewed as a clustering problem: we seek to cluster points in a metric space around $k$ centers, so to minimize the maximum load (or "star cost") of a cluster. Whereas ML$k$FL 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.

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

[2] 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.

algorithm for GAP [17]. 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 (ML$k$FL) 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 $\max_{i=1}^{k} \sum_{j \in \mathcal{C}: f(j)=i} d(i, j)$, 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 ML$k$FL on line metrics and prove Theorem 1. 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. 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 ML$k$FL solution and for any $v \in S$, say that the open interval with endpoints $f$ and $v$ is an *arm* of the star $(f, S)$ and we say that $f$ covers $v$. 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 1.** *There is a $(1 + \varepsilon)$-approximation algorithm for* ML$k$FL *on line metrics for any constant $0 < \varepsilon \leq 1$.*

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, will 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 1 is the following.

▶ **Theorem 2.** *Let $\Pi = (\mathcal{C} \cup \mathcal{F}, d, k)$ be a given* ML$k$FL *instance on a line metric. For any constant $0 < \epsilon \leq 1$ 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 $(1 + 18\epsilon) \cdot B$.*

We show how to complete the proof of Theorem 1, assuming Theorem 2 is true.

**Proof.** Set $\epsilon := \varepsilon/18$. We use binary search to find a value $B \leq L^{opt}$ such that algorithm $\mathcal{A}$ from Theorem 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 $\Delta$ is such that $v_i\Delta \in \mathbb{Z}$ for each point $i$. For each such value $\alpha$ 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 $d(i, f) \leq B$ for any $i \in S$ 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 an instance into instances where the distance of any two consecutive points is at most $B$. For each of the resulting instances $\Pi'$, 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 $\Pi'$ 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 describes how the optimum solutions to the original and new locations relate.

▶ **Lemma 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 $\epsilon B$ 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 $(1 + \alpha \cdot \epsilon)(1 + \epsilon) \cdot \epsilon B \leq (1 + (1 + 2\alpha)\epsilon) \cdot B$. 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 $\epsilon B$. 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 $\Pi'$ of $\Pi$, except we use the new points $v'_i$. By Lemma 3, this gives us a solution to $\Pi$ with cost at most $(1 + 18\epsilon)B$, proving Theorem 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 $\Pi$ of ML$k$FL satisfies the following properties: a) Each point $v_i$ is an integer between 0 and $(1 + 1/\epsilon) \cdot n^2$, b) There is a solution $\mathcal{S}_B$ with cost at most $B = (1 + 1/\epsilon) \cdot n$.

## 3.1 Structure of Near Optimum Solutions

In this section, we show that there is a near-optimum solution to the instance $\Pi$ 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 [17] to round such a fractional solution to an integral solution with cost at most $(1 + 2\epsilon)B$. We emphasize that 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 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 $B$.

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.

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

▶ **Lemma 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 (Sketch).** 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 $x$. We claim that no two small arms in the support of an optimal solution to (MIP) are crossing. The low-level details that support this claim will appear in the full version, but the idea is that if two small arms cross then we can fractionally uncross them to get a strictly better solution to (MIP) and the new fractional arms have length at most $2\epsilon B$. ◀

We will use Lemma 4 to prove the existence of a near-optimum solution to instance $\Pi$ 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 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 $p(m_i, j_a) \geq 0$ between 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 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 6** (Shmoys and Tardos, [17])**.** *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* $j_a$*,*
- $\sum_{a=1}^{n} p(m_i, j_a) x(m_i, j_a) \leq B$ *for each machine* $m_i$*.*

*Then there is an assignment $\phi$ 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 $\phi$ 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 4 to prove the following.

▶ **Theorem 7.** *There is a feasible (integer) solution to the* MLkFL *instance* $\Pi$ *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 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 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 $x^*$. ◀

▶ Remark. Our distinction between small and large arms is not just for the sake of obtaining a PTAS. In fact, we do not know if there is a completely uncrossed, $O(1)$-approximate solution. For instance, we have an example where iterating the fractional uncrossing argument to uncross all arms may create a fractional arm whose length is longer than $B$ by a super-constant factor.

## 3.2 Finding a Well-Formed Solution

### 3.2.1 Step Min-max Cost

Theorem 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 $\text{poly}(n, 1/\epsilon)$ describing the load assigned to $c$ from clients in $I$. We can look for partial solutions parametrized 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 through our dynamic programming approach. We begin by coarsening how we measure the length of long arms.

First, recall that all long arms have length more than $\epsilon B$. 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 $\epsilon B$; in other words each star is assigned at most $\frac{3}{\epsilon}$ long arms in the solution provided by Theorem 7. Say that one such long arm is between client $j$ and center $i$. If we moved both $j$ and $i$ left to their nearest integer multiples of $\epsilon^2 B$, then their distance changes by at most $\epsilon^2 B$. If this is done for all long arms assigned to a center $i$, then the total load of center $i$ due to long arms changes by at most $3\epsilon B$.

Now, notice that this way to measure the distance between client $j$ and center $i$ is simply $\epsilon^2 B$ times the number of integer multiples of $\epsilon^2 B$ that lie in the half-open interval $(v_i, v_j]$ if $v_i < v_j$ or $(v_j, v_i]$ if $v_j < v_i$. In the dynamic programming algorithm described below, we will use this coarse method to measure the distance of long arms and call this the *perceived cost* of the star. More specifically, the perceived cost of a star $(f, S)$ is the total cost of the small arms plus $\sum_{j \in S: I_{f,j} \text{ long}} |v''_f - v''_j|$ where $v''_i$ is the nearest multiple of $\epsilon^2 B$ to the left of $v_i$. The following is proved using arguments similar to the proof of Lemma 3, recalling that every star in $\mathcal{S}'_{\mathcal{B}}$ has at most $3/\epsilon$ long arms.

▶ **Lemma 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. Let the *small span or s-span* of a star be the interval, possibly empty, formed from the left most to the right most small client in this star. 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 forest like structure).

Also, consider the restriction of $\mathcal{S}$ to the interval $I_{i,j}$ for two arbitrary points $v_i$ and $v_j$. Assume that an arm has a direction and goes from the center of the star (i.e. the facility) to the client that it covers. There are some large arms that enter or leave this interval from $v_i$ or $v_j$. There are two types of arms: the arms that enter the interval $I_{i,j}$ from $v_i$ or $v_j$ or the arms that leave the interval $I_{i,j}$ from $v_i$ or $v_j$; for example a center inside the interval

$I_{i,j}$ might cover a client outside this interval or a center to the left of $i$ might cover a client inside the interval or a client to the right of $j$. Note that a large arm may have both these types, i.e., it enters from one endpoint and leaves from the other endpoint. The arms that enter the interval can cover the *deficiency* of coverage for some client in the interval and the arms that leave the interval provide coverage for some client outside of interval and can be viewed as *surplus* to the demand of coverage of the clients in the interval. Also, recall that in the perceived cost, the length of a large arm is measured as an integer multiple $q$ of $\epsilon^2 B$, where $0 \leq q \leq \frac{1}{\epsilon^2}$.

By the above observations, we can keep the information of all large arms that enter or exist the interval $I_{i,j}$ in four size $\frac{1}{\epsilon^2}+1$ vectors $\mathbf{D_i}, \mathbf{S_i}, \mathbf{D_j}$, and $\mathbf{S_j}$, which are the deficiency and surplus vectors of $v_i$ and the deficiency and surplus vectors of $v_j$ with respect to $I_{i,j}$, respectively. The $q$th entry ($0 \leq q \leq \frac{1}{\epsilon^2}$) of each of these vectors is an integer between 0 and $|\mathcal{C}|$. The $q$th element of vector $\mathbf{D_i}$ is the number of large arms entering from $v_i$ having perceived length $q \cdot \epsilon^2 B$ *past* $v_i$. More specifically, it is the number of clients $j'$ with $v_{j'} \geq v_i$ that are assigned to a center $c$ with $v_c < v_i$ such that the interval $(v_i, v_{j'}]$ contains $q$ multiples of $\epsilon^2 B$. In the same vein, entry $q$ of $\mathbf{S_i}$ records the number of clients $j'$ with $v_{j'} < v_i$ that are assigned to a center $c$ with $v_c \geq v_i$ such that the interval $(v_c, v_i]$ contains $q$ multiples of $\epsilon^2 B$. Similarly, $\mathbf{D}_j(q)$ is the number of large arms entering from $v_j$ with perceived length $q$ *prior to* $v_j$ and $\mathbf{S}_i(q)$ is the number of large arms exiting from $v_j$ with perceived length $q$ past $i$.

## 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 the values $A(k', i, j, c, \beta, \mathbf{D_i}, \mathbf{D_j}, \mathbf{S_i}, \mathbf{S_j})$ corresponding to subproblems. The meanings of the parameters are as follows. 1) $1 \leq i \leq j \leq n$ corresponds to the interval $I_{i,j}$, 2) $0 \leq k' \leq k$ is the number of centers (of stars) in the interval $I_{i,j}$, 3) $c \in \mathcal{F}$ denotes a single point with either $c < i$ or $c > j$ (i.e. outside of $I_{i,j}$) that is the center of some star, or else $c = \perp$. If $c \neq \perp$ it is the only center outside of $I_{i,j}$ with small arms going into $I_{i,j}$ and the total cost of small arms that $c$ pays to cover vertices in $I_{i,j}$ is $\beta$ where $0 \leq \beta \leq (1+5\epsilon)B$ is an integer. 4) $\mathbf{D_i}, \mathbf{D_j}, \mathbf{S_i}, \mathbf{S_j}$ are deficiency and surplus vectors for the endpoints of interval $I_{i,j}$. Note that in the above, if $c = \perp$ then the value of $\beta$ can be assumed to be zero. Let $q$ denote the number of multiples of $\epsilon^2 B$ lying in the interval $(v_i, v_j]$.

The subproblem $A(k', i, j, c, \beta, \mathbf{D_i}, \mathbf{D_j}, \mathbf{S_i}, \mathbf{S_j})$ is true if and only if the following holds. It is possible to open $k'$ centers in the interval $I_{i,j}$ and assign each $i' \in \mathcal{C}$ with $i \leq i' \leq j$: 1) to one of these open centers, or 2) to center $c$, if $c \neq \perp$, 3) or as a large arm exiting $I_{i,j}$, and also assign some of the large arms entering $I_{i,j}$ to these open centers such that: 1) the perceived load of each of the $k'$ centers is at most $(1+5\epsilon)B$, 2) the load of $c$ from small arms originating $i' \in \mathcal{C}$ with $i \leq i' \leq j$ is at most $\beta$, 3) and the large arms entering and/or exiting $I_{i,j}$ are *consistent* with $\mathbf{D_i}, \mathbf{D_j}, \mathbf{S_i}, \mathbf{S_j}$.

By consistent, we mean the following. First, for each $0 \leq a \leq \epsilon^{-2}$, each of the $\mathbf{D_i}(a)$ large arms entering $I_{i,j}$ is assigned to an open center $f$ such that $(v_i, v_f]$ contains precisely $a$ integer multiples of $\epsilon^2 B$, or (if $q \leq a$) exits $I_{i,j}$. A similar statement applies to $\mathbf{D_j}(a)$. Then for each $0 \leq a \leq \epsilon^{-2}$ we have that $\mathbf{S_j}(a)$ is precisely the number of large arms represented by $\mathbf{D_i}(a-q)$ that are not assigned to one of the $k'$ open centers in the interval plus the number of large arms originating from clients in the interval that exit by passing $v_j$ and have perceived length $a$ past $v_j$. Finally, we also require that no open center serves more than $3/\epsilon$ clients using large arms.

The number of table entries is polynomial, because $k', i, j, c$ are in $O(n)$ and $\beta'$ is a

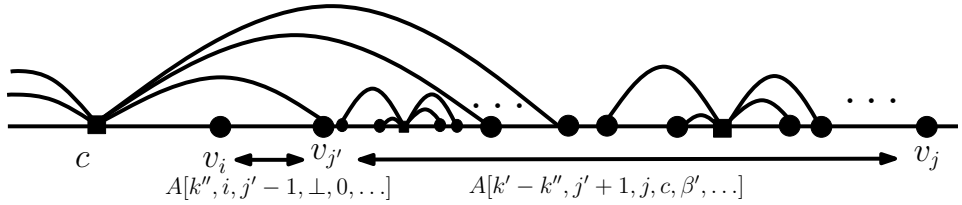


**Figure 1** Case 1 of recursive step.

polynomial in $n$ and $\frac{1}{\epsilon}$ and the deficiency and surplus vectors are in $O(n^{1+1/\epsilon^2})$, which is polynomial for a constant $\epsilon$. We shortly explain how one can compute the table entries in polynomial time. 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 this subsection we present the recurrence for the dynamic program of the PTAS for line.

**Base Case.** The base case is when $k' = 0$ and $i = j$. Without loss of generality assume that $i$ is a client (or else there is nothing to be covered). First, assume $c = \perp$ and so $\beta = 0$ or $c \neq \perp$ but $\beta = 0$. In either case, $v_i$ must be covered with a large arm. Assume this arm comes from left. In this case, the first component of $\mathbf{D_i}$, which corresponds to the number of large arms having perceived length $0 \cdot \epsilon^2 B = 0$ passed $v_i$, must be non-zero and one more than the first component of $\mathbf{S_j}$, because one will be used to cover $i$. All other components of these vectors must be the same. Also, all components of $\mathbf{S_i}$ and $\mathbf{D_j}$ must be the same. The case that the arm comes from right is similar. Now, assume $c \neq \perp$ and $\beta \neq 0$. Then, $v_i$ is a small client and it must be covered by $v_c$. Therefore, $d(v_i, v_c)$ must be equal to $\beta$. Also, we must have $\mathbf{D_i} = \mathbf{S_j}$ and $\mathbf{S_i} = \mathbf{D_j}$. In all other cases, the entry of the table will be set to False.

**Recursive Step.** Next, we show how to determine if $A[k', i, j, c, \beta, \mathbf{D_i}, \mathbf{D_j}, \mathbf{S_i}, \mathbf{S_j}]$ 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 regarding value of $c$:

1. $c \neq \perp$ and $\beta > 0$. There must be a small client in $I_{i,j}$ covered by $c$. We guess $j'$ to be the leftmost small client in $I_{i,j}$ covered by $c$. Now, we can break the subproblem into two smaller subproblems at the left and right sides of $j'$ (see Figure 1). If $j' = i$ or $j' = j$, one of the subproblems is empty and its value can be considered as true. Thus, assume $i < j' < j$. We guess $k''$ the number of stars having center in $I_{i,j'-1}$ and so the remaining $k' - k''$ centers will be in $I_{j'+1,j}$. Also, we guess the deficiency and surplus vectors, $\mathbf{D_{j'-1}}, \mathbf{S_{j'-1}}$, at $v_{j'-1}$ for the interval $I_{i,j'-1}$, and $\mathbf{D_{j'+1}}, \mathbf{S_{j'+1}}$, at $v_{j'+1}$ for the interval $I_{j'+1,j}$, such that all are consistent in the sense that $\mathbf{S_{j'-1}}(q) = \mathbf{D_{j'+1}}(q - q')$ where $q'$ is the number of integer multiples of $\epsilon^2 B$ in $(v_{j'-1}, v_{j'+1}]$, and similarly for large arms crossing $v_{j'}$ from right to left.
   We check to see if $A[k'', i, j'-1, \perp, 0, \mathbf{D_i}, \mathbf{S_i}, \mathbf{D_{j'-1}}, \mathbf{S_{j'-1}}]$ is true. If it is true, we examine the second subproblem. The coverage that $c$ provides to the right of $j'$ can be computed as $\beta'$ where $\beta' = \beta - d(v_c, v_{j'})$. We let $c' = c$ if $\beta' > 0$, and $c' = \perp$ if $\beta' = 0$. If

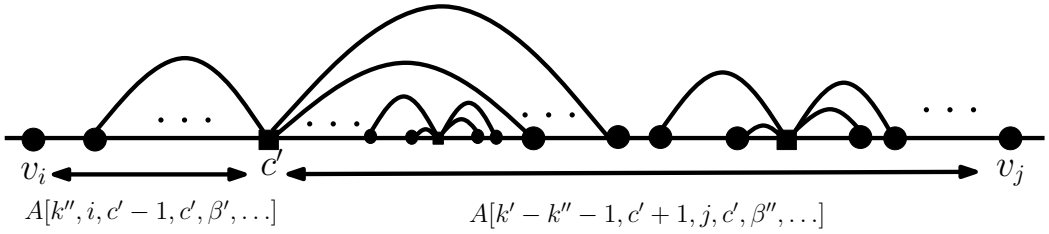


**Figure 2** Case 2b of recursive step.

$A[k'-k'', j'+1, j, c, \beta', \mathbf{D_{j'+1}}, \mathbf{S_{j'+1}}, \mathbf{D_j}, \mathbf{S_j}]$ is also true, we set the value of subproblem true.

2. $c \neq \perp$ and $\beta = 0$, or $c = \perp$. We consider two subcases regarding value of $k'$:
   - **(a)** $k' = 0$. All clients in $I_{i,j}$ must be covered by large arms from centers (facilities) outside the interval. First suppose that $i$ is a client and, without loss of generality, assume $v_i$ is covered by a large arm from the left. Then, the number of large arms having length $0 \cdot \epsilon^2 B = 0$ passed $v_i$ must be non-zero and we use one such arm to cover $v_i$. In this case we define $\mathbf{D'_i} = \mathbf{D_i} - (1, 0, \ldots, 0)$, i.e., the updated deficiency vector after covering $i$. If $i$ is not a client (and so does not need to be covered) we define $\mathbf{D'_i} = \mathbf{D_i}$. In both cases (whether $i$ is a client or not) suppose that $(v_i, v_{i+1}]$ has $q$ multiples of $\epsilon^2 B$ for some $0 \leq q \leq 1/\epsilon^2$. Thus, the value of the first $q$ components in $\mathbf{D'_i}$ must be zero. Define a deficiency vector $\mathbf{D_{i+1}}$ for $i+1$, which is equal to the vector obtained by shifting the values of $\mathbf{D'_i}$, $q$ places to the left (add trailing zeros for the values missing). Also, the last $q$ components of $\mathbf{S_i}$ must be zero, too (or else the arms that start at a node $v_{j'} \geq v_{i+1}$ and exits $v_i$ will have length larger than $B$). Define a surplus vector $\mathbf{S_{i+1}}$ for $i+1$, which is equal to the vector obtained by shifting the values of $\mathbf{S_i}$, $q$ places to the right (add leading zeros for the values missing). We set the value of this subproblem to $A[0, i+1, j, \perp, 0, \mathbf{D_{i+1}}, \mathbf{S_{i+1}}, \mathbf{D_j}, \mathbf{S_j}]$.
   - **(b)** $k' > 0$. Note that since $c \neq \perp$ and $\beta = 0$, or $c = \perp$, no small arm can enter $I_{i,j}$. Consider the set of centers in $I_{i,j}$. 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 2). Observe that the s-span of $c'$ is not contained in the s-span of any other star having a center in $I_{i,j}$. This star has at most $3/\epsilon$ large arms. Recall that in the perceived cost of a star, the length of large arms is measured in multiples of $\epsilon^2 B$. For each $0 \leq q \leq 1/\epsilon^2$, we guess $n_q^{(l)}$ and $n_q^{(r)}$ the number of length $q \cdot \epsilon^2 B$ large arms that $c'$ has (with respect to perceived cost) to its left and its right, respectively. We also guess $k''$ the number of stars having center in $I_{i,c'-1}$. We must have $k'-k''-1 \geq 0$ stars having center in $I_{c'+1,j}$. Also, we guess $\beta'$ where $c'$ provides $\beta'$ coverage to its left side. Finally, we guess the deficiency and surplus vectors, $\mathbf{D_{c'-1}}, \mathbf{S_{c'-1}}$, at $v_{c'-1}$ for the interval $I_{i,c'-1}$ and we guess the deficiency and surplus vectors, $\mathbf{D_{c'+1}}, \mathbf{S_{c'+1}}$, at $v_{c'+1}$ for the interval $I_{c'+1,j}$ and make sure that these vectors are consistent in the sense that $\mathbf{D_{c'+1}}(q-q') = \mathbf{S_{c'-1}}(q) - n_q^{(l)}$ where $q'$ is the number of integer multiples of $\epsilon^2 B$ in $(v_{c'-1}, v_{c'+1}]$, and similarly for the large arms crossing $c'$ from right to left.
     Now, we can break the subproblem into two smaller subproblems at the left and right sides of $c'$. We first check to see if $A[k'', i, c'-1, c', \beta', \mathbf{D_i}, \mathbf{S_i}, \mathbf{D_{c'-1}}, \mathbf{S_{c'-1}}]$ is

true (if $\beta' = 0$, we check $A[k'', i, c' - 1, \perp, 0, \mathbf{D_i}, \mathbf{S_i}, \mathbf{D_{c'-1}}, \mathbf{S_{c'-1}}]$ ). We guess $\beta''$ such that $\beta' + \beta'' + \sum_{q=0}^{\frac{1}{\epsilon^2}}(n_q^{(l)} + n_q^{(r)})q$, where the cost of small arms of $c'$ to clients in $I_{c'+1,j}$ is $\beta''$. We check and if $A[k' - k'' - 1, c' + 1, j, c', \beta'', \mathbf{D_{c'+1}}, \mathbf{S_{c'+1}}, \mathbf{D_j}, \mathbf{S_j}]$ is also true, we set the value of subproblem true (again if $\beta'' = 0$, we check $A[k' - k'' - 1, c' + 1, j, \perp, 0, \mathbf{D_{c'+1}}, \mathbf{S_{c'+1}}, \mathbf{D_j}, \mathbf{S_j}]$).

## 4 Tree Metrics

The extension of the PTAS presented for line metrics to tree metrics is not clear. However, there is a QPTAS for line metrics (see [10]) which uses a somewhat different approach. We describe the high level idea of that QPTAS (for line metrics) here and refer the reader to [10] for details. Then we explain how that approach can be extended to a QPTAS for tree metrics.

As before assume we have our points $v_1 \le v_2 \le \ldots \le v_n$ on a line and we have a guessed bound $B$ on the value of optimum. With a similar scaling approach as in Section 3 we can assume that the minimum distance between two consecutive points $i, i+1$ is at least $\epsilon B/n^2$ and at most $B$; thus the maximum pairwise distance is at most $nB$. Scaling everything by $\epsilon B/n^2$ we can assume the minimum distance is at least 1 the maximum distance between consecutive points is $n^3/\epsilon$ and that $B$ is at most $n^4/\epsilon$. This will increase the cost of the solution to at most $(1+\epsilon)B$. We then use a dynamic programming (DP) that computes a $(1+\epsilon)$-approximate solution to an instance satisfying above conditions. Each subproblem is defined by an interval $I_{i,j}$ and parameter $k'$, and the goal is to cover the clients in this interval with $k'$ stars whose centers are in this interval and some other stars whose centers are outside. We use a binary dissection to break the problem into two (almost) equal parts $I_{i,m}$ and $I_{m+1,j}$ where $m$ is the middle point. This gives rise to a dissection tree of height $O(\log n)$ with the interval $I_{1,n}$ at the root and $n$ singleton intervals $I_{i,i}$ as leaves. So the height of the recursion is $O(\log n)$. As before we keep deficiencies and surpluses vectors for the two ends $v_i$ and $v_j$: $\mathbf{D_i}, \mathbf{D_j}, \mathbf{S_i}, \mathbf{S_j}$, but they are defined slightly differently. Consider vector $\mathbf{S_i} = (s_1^{(i)}, \ldots, s_\sigma^{(i)})$ (with $\sigma$ to be defined soon). Each $s_a$ will keep the number of clients to the left of this interval (i.e. before $v_i$) that are at an approximate distance $l_a$ and are served by centers to the right of $i$ (possibly after $j$). Similarly $\mathbf{S_j}$ stores the number of clients to the right of $v_j$ that are served by centers before $j$. $\mathbf{D_i}$ and $\mathbf{D_j}$ will be representing the number of clients inside $I_{i,j}$ within an approximate certain distance from $v_i$ (or $v_j$) that are to be covered by a center outside of the interval. To cut down on the interface of an interval $I_{i,j}$ with the rest of the line, we round up the surplus and deficiency lengths on each of the right and left sides to the nearest power of $(1+\epsilon'/\log n)$, for some $\epsilon'$ depending on $\epsilon$, at each level of dissection. Thus, we only keep track of lengths $l_a = (1+\epsilon'/\log n)^a$, $a \in \{1, \ldots, \sigma\}$. For instance, for $\mathbf{S_i} = (s_1^{(i)}, \ldots, s_\sigma^{(i)})$, $s_a^{(i)}$ will be the number of clients to the left of $I_{i,j}$ that are at (scaled up) distance $(1+\epsilon'/\log n)^a$ from $i$ that are served by centers inside the interval $I_{i,j}$. So there will be $\sigma = O(\log n \cdot \log B/\epsilon') = O(\log^2 n/\epsilon')$ different lengths and as a result at most $n^{O(\log^2 n/\epsilon')}$ different surplus and deficiency vectors. In this way, each arm of a star will be scaled up by a factor of at most $(1+\epsilon'/\log n)$ at each level of DP computation (to account for the rounding), and since the depth of recursion (dissection) is $\lceil \log n \rceil$, this will result in an extra factor of $(1+\epsilon'/\log n)^{\lceil \log n \rceil} \le (1+\epsilon)$ (for a suitable choice of $\epsilon'$) over the entire length of each arm. In other words, if a subproblem for an interval $i, j$ and parameter $k'$ is feasible (with each star costing at most $B$) without rounding the lengths of deficiency and surplus vectors then the subproblem with rounded (up to nearest power of $(1+\epsilon'/\log n)$) lengths for deficiency and surplus vectors is feasible if each star is allowed to have cost at most $(1+\epsilon) \cdot B$.

Each entry of the table represents a subproblem $(i, j, k', \mathbf{D_i}, \mathbf{D_j}, \mathbf{S_i}, \mathbf{S_j})$, where:

1. $i, j$ represents the interval $I_{i,j}$.
2. $k'$ is the number of centers to be opened from among the points in $I_{i,j}$.
3. $\mathbf{D_i} = (d_1^{(i)}, \ldots, d_\sigma^{(i)})$ and $\mathbf{D_j} = (d_1^{(j)}, \ldots, d_\sigma^{(j)})$ are the deficiency vectors on the left and right sides of the interval $I_{i,j}$, respectively.
4. $\mathbf{S_i} = (s_1^{(i)}, \ldots, s_\sigma^{(i)})$ and $\mathbf{S_j} = (s_1^{(j)}, \ldots, s_\sigma^{(j)})$ are the surplus vectors on the left and right sides of $I_{i,j}$, respectively.

Each surplus and deficiency vector is a vector of size $\sigma = O(\log^2 n/\epsilon')$, where $d_a^{(p)}$ or $s_a^{(p)}$ (for $p \in \{i, j\}$) is the number of broken arm parts of length $(1 + \epsilon'/\log n)^a$ (after rounding). Each entry of the table records in Boolean values the feasibility of having $k'$ stars centered in the points in $I_{i,j}$, such that each star has cost at most $(1 + \epsilon) \cdot B$. Each of the $k'$ stars would cover some clients in $I_{i,j}$ and the clients located at distances $\mathbf{S_i}$ and $\mathbf{S_j}$ from the endpoints $i$ and $j$ of the interval. The rest of the clients have to be covered with the broken arms of $\mathbf{D_i}$ and $\mathbf{D_j}$, thus connected to the two sides $i$ and $j$, respectively. The size of the DP table is $O(n^2 \cdot k \cdot n^{O(\log n \log B/\epsilon')}) = n^{O(\log^2 n/\epsilon')}$, which is quasi-polynomial in $n$. See [10] for the details of how to fill in the entries of this table.

Now suppose that the given metric for the ML$k$FL instance can be represented as a cost function on the edges of a tree $T$. The algorithm, as before, works with a guessed value $B$ as an upper bound for $L^{opt}$. Also, using a scaling argument as for the case of line metrics, we can assume that the aspect ratio of heaviest to cheapest edge cost is polynomially bounded. Next, we can make the tree $T$ binary by introducing zero-cost edges at nodes that have more than two children, keeping one of its children and placing the rest as a subtree hanging from the zero-cost edge added. Repeating this gives a binary tree that still has linear size. So for the rest of this section we assume that the input tree is binary.

For each binary tree with $n$ nodes one can find an edge $e = (u, v)$ (where $u$ is parent of $v$) such that each subtree resulted by deleting $e$ has size in $[n/3, 2n/3]$. This splitting of the tree into two subtrees $T_v$ (tree rooted at $v$) and $T \setminus T_v$ that are almost the same size (by a factor of at most two) plays the role breaking the problem into two almost equal sizes. Given a binary tree $T$ we can recursively partition it into two "almost equal" subtrees until we arrive at subtrees of size 1. The depth of this recursive dissection will be $O(\log n)$ and each time we recursively break the tree into two smaller binary trees (whose sizes differ by a factor of at most 2). The breaking point introduces a new interface (or "portal") point for the two smaller sub-trees: if edge $e = (u, v)$ is cut then $v$ is an interface point (or portal) for the subproblem $T_v$ in addition to any other interface point it might have had passed on to from previous dissection operations, and $u$ is an interface point (portal) for $T \setminus T_v$ in addition to any other portal points generated before. More specifically, each subproblem is of the form $(T', k', \{S^{(p)}\}_{p \in P(T')}, \{D^{(p)}\}_{p \in P(T')})$ where $T'$ is a subtree that is obtained by performing the dissection operation, $0 \leq k' \leq k$ is the number of centers of stars to be opened in $T'$, $P(T') \subseteq V(T')$ is the set of portal points of $T'$. If a tree $\hat{T}$ is cut into two almost equal sized subtrees $T_1$ (rooted at $v$) and $T_2 = \hat{T} \setminus T_1$ by cutting edge $e = (u, v)$ then $P(T_1)$ will consist of all the portals of $\hat{T}$ that are in $T_1$ plus node $v$. Similarly $P(T_2)$ consists of all the portals of $\hat{T}$ that are in $T_2$ plus node $u$. It follows that for each subproblem, the number of portals is at most $O(\log n)$. The dynamic programming then follows along the same lines as the QPTAS described above (see [10] for details) for the line metrics, and we obtain the following theorem.

▶ **Theorem 9.** *For any constant $0 < \varepsilon \leq 1$, there is a $(1 + \varepsilon)$-approximation algorithm for* ML$k$FL *on tree metrics that runs in quasi-polynomial time.*

## 4.1 Star Metrics

We now consider MLkFL in star metrics, but in the more-general setting where each client $j$ has an integer demand $D_j$ that may be split integrally across various open facilities; we call this an *integer-splittable assignment*. The load of a facility $i$ is now defined as $\sum_j x_{ij} d(i,j)$ where $x_{ij} \in \mathbb{Z}_{\geq 0}$ is the amount of $j$'s demand that is served by $i$. We devise a 14-approximation algorithm for this problem. 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 it is easy to ensure via some preprocessing that $d(i,j) \leq 2B$ for every facility $i$ and client $j$. Thus, we either determine that $B < L^{opt}$ or obtain a solution with maximum load at most $14B$. The details are deferred to the full version of this paper.

▶ **Theorem 10.** *There is a 14-approximation algorithm for* MLkFL *on star metrics with non-uniform demands and integer-splittable assignments.*

# 5 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 and *APX*-hard in the Euclidean plane (Theorems 11 and 12). We also demonstrate that a natural configuration-style LP has an unbounded integrality gap (Theorem 13). The details of the first two theorems are deferred to the full version of this paper.

▶ **Theorem 11.** *Minimum-load $k$-facility location is strongly NP-hard even in line metrics.*

▶ **Theorem 12.** *It is NP-hard to $\alpha$-approximate minimum-load $k$-facility location problem on the Euclidean plane, for any $\alpha < 4/3$. Thus,* MLkFL *is* APX*-hard in the Euclidean plane.*

## 5.1 Integrality-gap Lower Bound

Let $(\mathcal{F}, \mathcal{C}, d, k)$ 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.

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

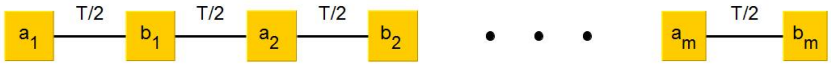


**Figure 3** Example showing bad integrality gap for the configuration LP in line metric.

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 ML$k$FL instance on the line metric, where the smallest value $B_{\mathrm{LP}}$ 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 13.** *The integrality gap of* (P) *is* $\Omega(k/\log k)$ *even for line metrics.*

**Proof.** Assume for simplicity that $k$ is odd (the argument easily extends to even $k$). Consider the following simple ML$k$FL 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 3, 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 $\frac{1}{k+1}$ 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. (If $k$ is even, we may set $B = 2T$ and choose the above configurations for the first $k-2$ facilities and the $k$-th facility; for facility $k-1$, we consider $\binom{n}{2}$ configurations, each of which contains all the clients colocated at facility $k-1$, two clients colocated with the $(k-2)$-th facility and 2 clients colocated with the $k$-th facility.)

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 $L \geq \frac{nT}{4H_k}$. (Note that this argument does not depend on whether $k$ is odd or even.) ◀

## 6 Concluding Remarks

In this paper we present the first true polynomial time approximation for the ML$k$FL restricted to line metrics and the first true approximation in tree metrics. We also show that the standard tools of LP rounding (even for configuration based LP) or local search methods, which have been used successfully for various facility location problems do not seem to work

for this problem (even for this restricted metrics). Obviously, the major open question here is to obtain a true approximation (even an $O(\log n)$-approximation) for the ML$k$FL on general metrics. A smaller step could be to obtain such an algorithm for the Euclidean metrics. Note that the *APX*-hardness result for the Euclidean metrics shows that this problem is significantly more difficult than the uncapacitated facility location or $k$-median (for which there are known PTAS's).

## References


1. H.-C. An, A. Bhaskara, C. Chekuri, S. Gupta, V. Madan, and O. Svensson. Centrality of trees for capacitated $k$-center. In *Proceedings of APPROX*, 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. 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.
8. K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. Greedy facility location algorithms analyzed using dual-fitting with factor-revealing lp. *Journal of the ACM*, 50(6):795–824, 2003.
9. 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.
10. A. Jorati. Approximation algorithms for min-max vehicle routing problems. Master's thesis, University of Alberta, Department of Computing Science, 2013.
11. M. R. Khani and M. R. Salavatipour. Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. In *APPROX*, 2011.
12. S. Li and O. Svensson. Approximating $k$-median via pseudo-approximation. In *Symposium on Theory of Computing (STOC)*, 2013.
13. P. Mirchandani and R. Francis, editors. *Discrete location theory*. Jown Wiley and Sons, 1990.
14. H. Nagamochi and K. Okada. Approximating the minmax rooted-tree cover in a tree. *Information Processing Letters*, 104(5):173–178, 2007.
15. R. Ravi. Workshop on Flexible Network Design, 2012. `http://fnd2012.mimuw.edu.pl/qa/index.php?qa=4&qa_1=approximating-star-cover-problems`.
16. D. B. 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.
17. D. B. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62(3):461–474, 1993.