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 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.

*Dept. of Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1. Supported by the last author's NSERC grant 327620-09. Email: sahmadian@gmail.com

†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.

‖Dept. of Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1. Supported in part by NSERC grant 327620-09, an NSERC Discovery Accelerator Supplement Award, and an Ontario Early Researcher Award. Email: cswamy@math.uwaterloo.ca.

1 Introduction

Facility-location (FL) problems have been widely studied in the Operations Research and Computer Science communities (see, e.g., [16] and the survey [19]), 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 $d(f, j)$s form a metric.

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 [18]). 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, 12, 11, 3, 15] (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 [18]. 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], 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. 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 completely 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 Jorati [13] for line metrics extends to yield a quasi-PTAS (QPTAS) for tree metrics (Theorem 4.1). 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 MLkFL on star metrics

[1] Jorati [13], in his Master's thesis, obtained a preliminary version of some of our current results.

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), which makes the setting with non-uniform demands strictly more general than the unit-demand setting.

In Section 6, we obtain various computational-complexity and integrality-gap lower bounds for $\mathsf{ML}k\mathsf{FL}$. Complementing our PTAS, we show (Theorem 6.1) that $\mathsf{ML}k\mathsf{FL}$ 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 $\mathsf{ML}k\mathsf{FL}$ is *APX*-hard in the Euclidean plane (Theorem 6.2). Finally, we justify our comment about the difficulty of tackling $\mathsf{ML}k\mathsf{FL}$ via the various LP-based methods developed for facility-location problems by showing that even a configuration-style LP-relaxation for $\mathsf{ML}k\mathsf{FL}$—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 6.4). Note that the configuration LP is stronger than the natural LP-relaxation for $\mathsf{ML}k\mathsf{FL}$. 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.

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 $\mathsf{ML}k\mathsf{FL}$ (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 $\mathsf{ML}k\mathsf{FL}$ 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 $\mathsf{ML}k\mathsf{FL}$ makes it difficult to exploit this. In particular, one can come up with simple examples where a k-median style multi-swap local-search 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.)

At an intuitive level, one way to explain some of the difficulties encountered in dealing with $\mathsf{ML}k\mathsf{FL}$ is as follows. Much of the machinery used for FL problems relies explicitly or implicitly on the fact that the underlying problem is "stable under perturbations": suppose that one perturbs the instance by "moving" each client j by an amount δ_j, where either: (a) $\sum_{j \in \mathcal{C}} \delta_j$ can be bounded in terms of the optimal value (for min-sum problems); or (b) each δ_j can be bounded in terms of the optimal value (for min-max problems). Then, *any* near-optimal solution to the perturbed instance yields a near-optimal solution to the original instance. This principle forms the basis of various clustering-based algorithms that simplify the original instance by aggregating "nearby" groups of clients into clusters, and yields approximation algorithms for a variety of FL problems including uncapacitated FL, k-median, uncapacitated and capacitated k-center [5, 1].

This principle however does not apply to $\mathsf{ML}k\mathsf{FL}$ since connection costs are *aggregated* for each facility. For instance, by moving clients we may end up with a perturbed instance Π' with a location v having a large set S of colocated clients and hence open a facility f at v that serves all of S, but the *actual load* induced by S at f may be arbitrarily large. Moreover, to ensure that near-optimal solutions to Π' translate to near-optimal solutions to the original instance, one needs to solve a *harder* variant of the problem for Π' (where we have some "capacity constraints"), which defeats the purpose of moving to Π' in the first place.

Given these difficulties, 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 $(1 + O(\epsilon))$-optimal 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 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 [20] 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, 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., [16, 19]); 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 [17]), 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 [14]. 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. [8] 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.[2] Since the latter problem admits a PTAS [10], 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 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, MLkFL can also be motivated and viewed as a clustering

[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.

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 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.

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 [20]. 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. 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 $v \in S$, 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 3.1 *There is a $(1 + \varepsilon)$-approximation algorithm for* MLkFL *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, 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 $0 < \epsilon \leq 1$ 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 $(1 + 18\epsilon) \cdot B$.*

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 $d(i, f) \leq B$ 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 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 $(1 + \alpha \cdot \epsilon)(1 + \epsilon) \cdot \epsilon B \leq (1 + (1 + 2\alpha)\epsilon) \cdot B$. 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 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 $\mathsf{ML}k\mathsf{FL}$ satisfies the following properties:

- Each point v_i is an integer between 0 and $(1 + 1/\epsilon) \cdot n^2$.
- 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 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 $\mathsf{ML}k\mathsf{FL}$ 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 [20] 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 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.

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

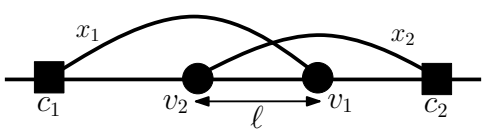
We stress that this is *not* a relaxation for $\mathsf{ML}k\mathsf{FL}$. The objective function is more similar to the objective function for the k-median problem. Rather, we will only be using this to help demonstrate the existence of a well-formed solution. The objective function acts as a potential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

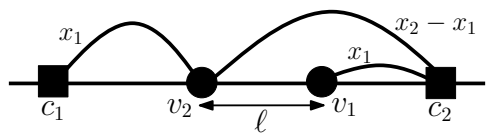
Proof : First observe that there is in fact a feasible solution x because the integer solution $\mathcal{S}_B$ is feasible for this ILP. By standard theory of mixed-integer programming and the fact that the set of feasible solutions is

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.

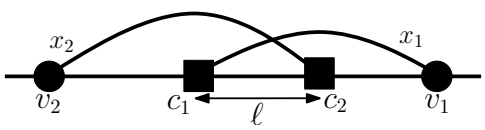
So, suppose x is a feasible solution such that two small arms $I_{i,j}$ and $I_{i',j'}$ in the support of x cross. To simplify notation, let $c_1 = v_{f_i}, c_2 = v_{f_{i'}}$ be the locations of the centers $f_i, f_{i'}$ and $v_1 = v_j$ and $v_2 = v_{j'}$ be the locations of the clients j, j'. Also let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



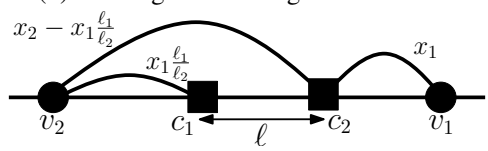
(a) v_1 and v_2 between c_1 and c_2.



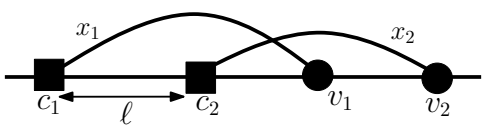
(b) Coverage after fixing the intersection in Case (1).



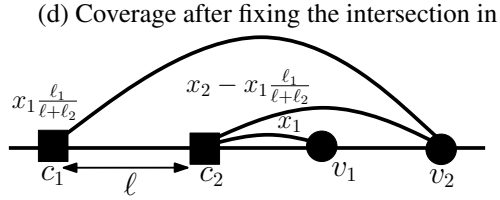
(c) v_1 and v_2 on different sides of c_1c_2.



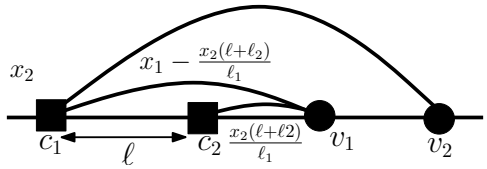
(d) Coverage after fixing the intersection in Case (2).



(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 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.

(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). It can be verified that the fractional load at each center c_1 and c_2 does not

increase. Furthermore, since assignment is shifted to strictly smaller arms, then the objective function strictly decreases.

(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, $\ell_1 < \ell + \ell_2$. It is not hard to see that $(\ell + \ell_2)(\ell_1 - \ell) < \ell_1\ell_2$.

There are two sub-cases: either $x_1\ell_1 \leq x_2(\ell + \ell_2)$ or $x_1\ell_1 > x_2(\ell + \ell_2)$. Assume $x_1\ell_1 \leq x_2(\ell + \ell_2)$ and hence, $x_1\frac{\ell_1}{\ell+\ell_2} \leq x_2$. 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 c_1 changes by $x_1\frac{\ell_1}{\ell+\ell_2}(\ell + \ell_2) - x_1\ell_1 = 0$ and the fractional load at c_2 changes by $x_1(\ell_1 - \ell) - x_1\frac{\ell_1}{\ell+\ell_2}\ell_2 < 0$. Since the total load strictly decreases, then the objective function value also strictly decreases.

For sub-case two, suppose that $x_1\ell_1 > x_2(\ell + \ell_2)$. 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 total fractional coverage of v_1, v_2 remains unchanged. The fractional load at c_1 changes by be $x_1\ell_1 - x_2(\ell + \ell_2) - x_1\ell_1 + x_2(\ell + \ell_2) = 0$ and the fractional load at c_2 changes by $x_2\ell_2 - \frac{x_2(\ell+\ell_2)(\ell_1-\ell)}{\ell_1} < 0$. So the total load stritcly decreases.

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 $p(m_i, j_a) \geq 0$ 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 [20] 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, [20]) *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,*
- $\sum_{a=1}^{n} x(m_i, j_a) \leq B$ *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 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 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; in other words each star is assigned at most $\frac{3}{\epsilon}$ long arms in the solution provided by Theorem 3.7. 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 3.3, recalling that every star in $\mathcal{S}'_{\mathcal{B}}$ has at most $3/\epsilon$ long 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. 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 qth entry ($0 \leq q \leq \frac{1}{\epsilon^2}$) of each of these vectors is an integer between 0 and $|\mathcal{C}|$. The qth 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 $(v_i, v_c]$ 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 $\mathbf{S}_j(q)$ is the number of large arms exiting from v_j with perceived length q past i. **MS: please double check this paragraph.**

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.

- $1 \leq i \leq j \leq n$ corresponds to the interval $I_{i,j}$.
- $0 \leq k' \leq k$ is the number of centers (of stars) in the interval $I_{i,j}$.
- $c \in \mathcal{F}$ 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 β where $0 \leq \beta \leq (1 + 5\epsilon)B$ is an integer.
- $\mathbf{D_i}, \mathbf{D_j}, \mathbf{S_i}, \mathbf{S_j}$ 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 β 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 $i \leq i' \leq j$

- to one of the k' open centers,
- to center c, if $c \neq \perp$,
- 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 in $I_{i,j}$ such that

- 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 $i' \in \mathcal{C}$ with $i \leq i' \leq j$ is at most β,
- 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 β' is a 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 ϵ. 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

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 β. 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 2). 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 β' where $\beta' = \beta - d(v_c, v_{j'})$. We let $c' = c$ if $\beta' > 0$, and $c' = \perp$ if $\beta' = 0$. If $A[k' - k'', j' + 1, j, c, \beta', \mathbf{D_{j'+1}}, \mathbf{S_{j'+1}}, \mathbf{D_j}, \mathbf{S_j}]$ is also true, we set the value of subproblem true.

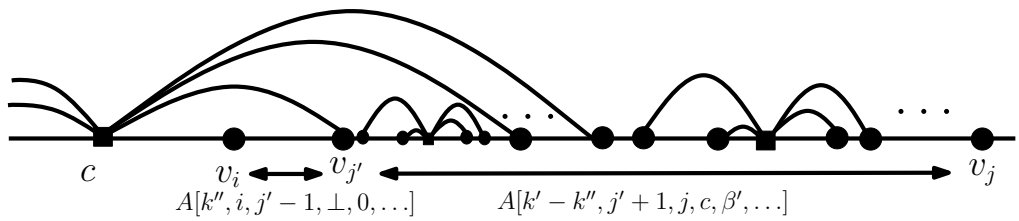


Figure 2: Case 1 of recursive step

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 3). 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 β' where c' provides β' 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 β'' 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 β''. 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 $A[k' - k'' - 1, c' + 1, j, \perp, 0, \mathbf{D_{c'+1}}, \mathbf{S_{c'+1}}, \mathbf{D_j}, \mathbf{S_j}]$).

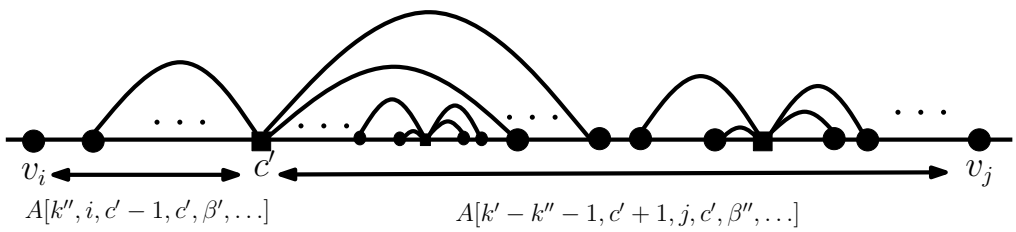


Figure 3: Case 2b of recursive step

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 [13]) which uses a somewhat different approach. We describe the high level idea of that QPTAS (for line metrics) here and refer the reader to [13] for details. Then we explain how that approach can be extended to a QPTAS for tree metrics.

As before assume we have our points $v_1 \leq v_2 \leq \ldots \leq v_n$ 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/ϵ and that B is at most n^4/ϵ. 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 σ 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 ϵ' depending on ϵ, 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 $(1+\epsilon'/\log n)^{\lceil \log n \rceil} \leq (1+\epsilon)$ (for a suitable choice of ϵ') 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 centers to the left of i or right of j. 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 [13] for the details of how to fill in the entries of this table.

Now suppose that the given metric for the MLkFL 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 the 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 [13] for details) for the line metrics, and we obtain the following theorem. **MS: how much details should we really present here? Note that [13] has only the details for QPTAS for line**.

Theorem 4.1 *For any constant $0 < \varepsilon \leq 1$, there is a $(1 + \varepsilon)$-approximation algorithm for* $\mathsf{ML}k\mathsf{FL}$ *on tree metrics that runs in quasi-polynomial time.*

5 A constant-factor approximation algorithm for $\mathsf{ML}k\mathsf{FL}$ in star metric

We now consider $\mathsf{ML}k\mathsf{FL}$ 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 $14B$.

Theorem 5.1 *There is a 14-approximation algorithm for* $\mathsf{ML}k\mathsf{FL}$ *on star metrics with non-uniform demands and integer-splittable assignments.*

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 integer demand of client $j \in \mathcal{C}$. Recall that we consider integer-splittable assignments, where each open facility serves an integer amount of the demand (possibly 0) of each client. We often refer to a pair (i, j), where $i \in \mathcal{F}, j \in C$, as an arm.

Let B be our current guess of the optimal value. We either certify that $B < L^{opt}$, or find a solution with maximum load at most $14 \cdot B$. 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 $i \in \mathcal{F}$) in any integer-splittable assignment; also, if $D_i > 0$, then 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.

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 $6B$. Combining these results, if $B \geq L^{opt}$, we can find k facilities and a fractional assignment of clients to these facilities that has maximum load at most $12B$. Finally, using Theorem 3.6, we can round this solution to an integer solution while increasing the maximum load by at most $\max_{i \in \mathcal{F}, j \in \mathcal{C}} d(i,j) \leq 2B$.

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.

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

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 $d(i,j) \cdot d(i',j') < d(i',j) \cdot d(i,j')$. 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:

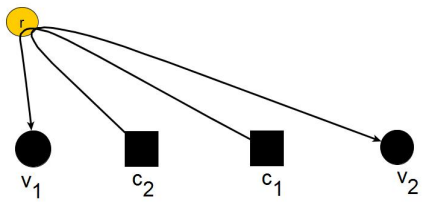
- $x'_{ij} = x_{ij} + \epsilon, \quad x'_{i'j} = x_{i'j} - \epsilon.$
- $x'_{i'j'} = x_{i'j'} + \epsilon', \quad x'_{ij'} = x_{ij'} - \epsilon'.$

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

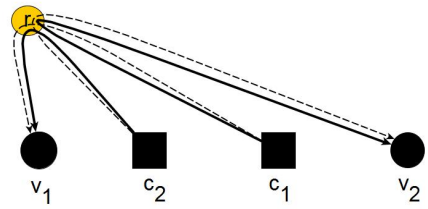
$$\epsilon' \cdot d(i',j') - \epsilon \cdot d(i',j) = \epsilon \left(\frac{d(i,j) \cdot d(i',j')}{d(i,j')} - d(i',j) \right)$$

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

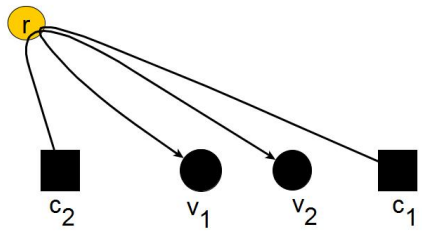
Observe that the above uncrossing property is stronger then the uncrossing that we achieved for line metrics, where we only ensured that small arms do not cross. The figure below illustrates all the (non-symmetric) cases that count as crossing. The figures on the right show the result after modifying x as



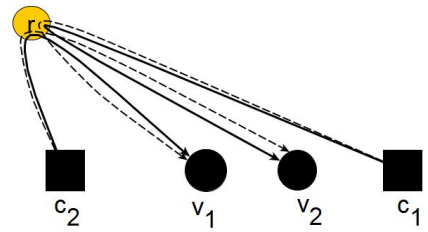
(a) Case(1) :c_1 and c_2 between v_1 and v_2.



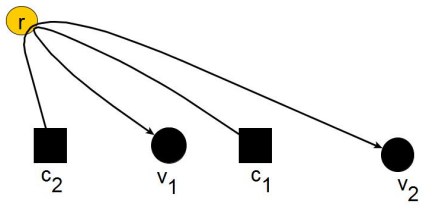
(b) Coverage after fixing the intersection in Case (1).



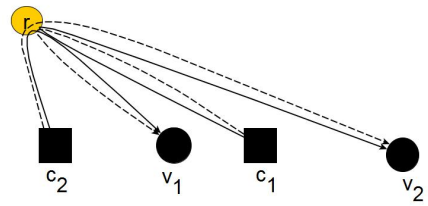
(c) Case (2): v_1 and v_2 between c_1 and c_2.



(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).

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, we have one of the following:
  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 with maximum load at most $6B$.*

Proof : Let x be an optimal solution to (S-P). So x has maximum load at most B, and by Lemma 5.2, it does not have any crossings. We modify x in a number of steps that do not increase the maximum load by more than a constant factor.

- **Step 1) Ensuring all clients are served in only one direction.**
  If client $j \in C_B$ then all the demand at j can be satisfied by the colocated center, we can assume

$x_{ij} = 0$ for $i \neq j$. Otherwise, either $\sum_{i<j} x_{ij} \geq \frac{1}{2}$ or $\sum_{i>j} x_{ij} \geq \frac{1}{2}$. Suppose the former is true (the latter is similar). Then we simply set $x_{ij} = 0$ for $i > j$ and scale the x_{ij} with $i < j$ uniformly until they sum to 1 again. After doing this for all clients, we have that x_{ij} at most doubles for each center-client pair so the maximum load is at most $2B$.

- **Step 2) Ensuring all centers either serve clients only to the left or only to the right, or form their own consecutive partition.**
  Let i be any center in C_B with $x_{ij}, x_{ij'} > 0$ for clients $j < i < j'$. 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.

  Consider the rightmost center $i_L \in C_B$ such that $i_L < i$, see Figure 4. If there is no center to the left of i, then skip this paragraph. Update the assignment x in the following way. For any $j < i_L$ with $x_{ij} > 0$, we update $x_{i_L j} \leftarrow x_{i_L j} + x_{ij}$ and $x_{ij} \leftarrow 0$. This does not introduce any crossings in x since j's assignment moved to a closer location. Note that if i serves a client $j < i_L$ then i_L cannot serve any client $j' > i_L$ as in this case x has a crossing, so i_L only served clients to the left.

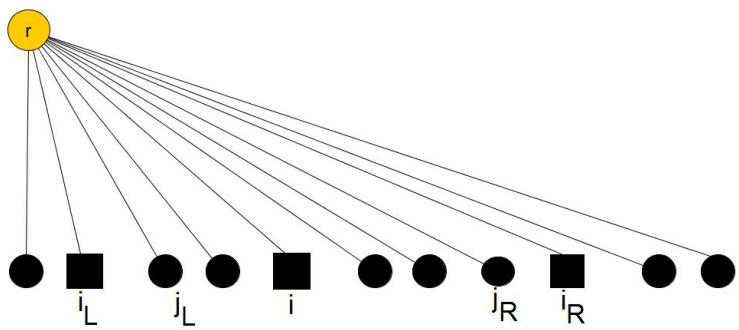


Figure 4: Moving client assignments. Black solid circles denote clients with $x_{ij} > 0$.

  Similarly, if there is a leftmost center $i_R \in C_B$ with $i_R > i$ then move all assignment x_{ij} with $j > i_R$ to i_R. Again this does not introduce any crossings in x since j's assignment is moved to a closer location, and moreover i_R only serves clients to the right.

  Now note the following properties. Let j_L be the leftmost client with $x_{ij_L} > 0$ and, similarly, j_R the rightmost client with $x_{ij_R} > 0$ (if any). Then, we have $i_L < j_L \leq i$ and $i_R > j_R \geq i$ (whenever these clients and centers exist). Furthermore, any j' between j_R and j_L (inclusively) must be completely assigned to i: i.e. $x_{ij'} = D_{j'}$. Form a partition $V_a = \{j_L, \ldots, j_R\}$ and note that this partition 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 $x_{i'j'} = 0$ for any $j' < i < i'$ or $i' < j < j'$. Otherwise, if (say) $j' < i < i'$ has $x_{i'j'} > 0$ then this would cross with the client to the right of i that was assigned to i (the first sentence of this step).

  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_{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, it cannot be i_L for some center i_0 and i_R for some center i_1 where there exists clients $j_0 < i'$ and $j_1 > i'$ with $x_{i_0 j_0}, x_{i_1 j_1} > 0$ as these two arms cross. 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 $2B$. 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 although the load increase on i' is more than the load decrease on i but this increase is at most twice the load of i corresponding to serving clients passed i' which is at most $2B$. Hence the load increase in this case is at most $4B$.

- **Step 3) Dividing the remaining instance.**
  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 $x_{ij} > 0$ only for $i < j$ **or** $j \in C_B$ and $x_{jj'} > 0$ 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 $x_{i'j'} > 0$ 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'. ■

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

We describe the dynamic programming approach in two steps. 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). 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 $O(n)$ 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

$\sum_{j \le i} \left(D_j - \sum_{i' \in C'} x_{i'j}\right)$ among all ways to choose p' centers C' in $\{j_L, \ldots, i\}$ and fractionally assign up to D_j units of demand of each client $j \le i$ to centers in C' such that no center $i' \in C'$ is assigned more than $6B/d_{i'}$ units of demand and $x_{i'j} = 0$ 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 $\sum_{j \ge i} d_j \cdot \left(D_j - \sum_{i' \in C'} x_{i'j}\right)$. We associate the table $T_3(V', p)$ for this case, and we use the auxiliary table $g(i, p')$ for $j_L \le i \le j_R, 0 \le p' \le p$. $g(i, p)$ is the minimum possible excess load $\sum_{j \ge i} d_j \left(D_j - \sum_{i' \in C'} x_{i'j}\right)$ among all ways to choose p' centers C' in $\{i, \ldots, j_R\}$ and fractionally assign up to D_j units of demand of each client $j \ge i$ to centers in C' such that no center $i' \in C'$ is assigned more than $6B$ and $x_{i'j} = 0$ 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 and *APX*-hard in the Euclidean plane (Theorems 6.1 and 6.2). We also demonstrate that a natural configuration-style LP has an unbounded integrality gap (Theorem 6.4).

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 $b_i \le 2^{16} n^4$ and $\frac{B}{4} < b_i < \frac{B}{2}$ for each i (e.g. [9]). 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, we place $3k^2(B+1)$ clients at each point in $\{-\frac{k-1}{3k}, -\frac{k-2}{3k}, \ldots, -\frac{1}{3k}, 0\}$. Next, for each integer $b_i, 1 \le i \le n$, 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

bit complexity 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.

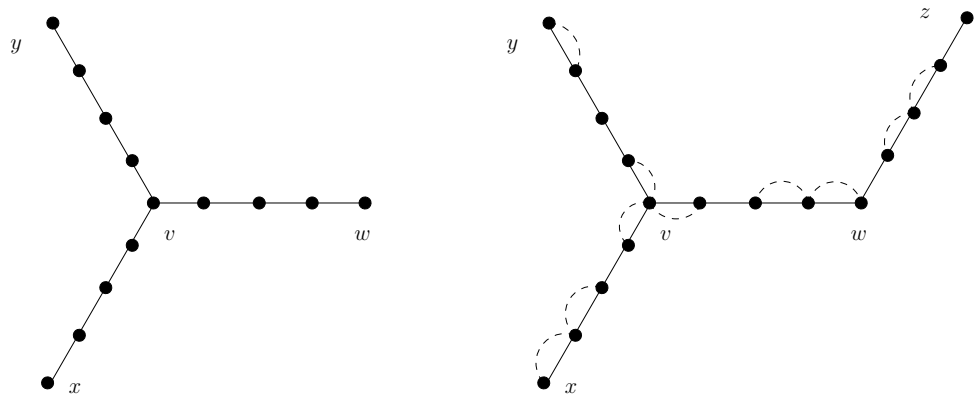
First, suppose there is a partition of the integers $b_1, \ldots, b_n$ into k groups $G_1, \ldots, G_k$ such that the sum of the integers in any group G_i is B. For each $1 \leq i \leq k$ we create a star with center at $-\frac{i-1}{3k}$, assign all clients located at this center to this star, and also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

Theorem 6.2 *It is NP-hard to α-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.*

Proof : We give a gap-introducing reduction from the dominating set problem on planar graphs with maximum degree 3, which is known to be *NP*-hard [9]. The reduction is similar to the *APX*-hardness proof of k-center, given by [7]. We construct an instance Π of MLkFL as follows. A planar graph of degree at most 3 can be embedded into the plane, where each edge e is replaced with a path $p(e)$ of edges of length 1, such that the length of the path $|p(e)|$ is $3i + 1$, for some i. Each degree 3 vertex will have its three edges at 120 degree angles from each other. Let H be this new graph. The Embedding of one degree-3 vertex and its three neighbors in H has been shown in Figure 5a. Note that each of the edges $\{v, w, \}, \{v, x\}, \{v, y\}$ is replaced by a path of length 4. We take the facilities $\mathcal{F}$ and clients $\mathcal{C}$ to be the vertices of H and the metric to be the shortest path metric of graph H.



(a) A degree-3 vertex v and its three neighbors.

(b) A Minimum-load k-facility location of cost 3.

Figure 5: A degree-3 vertex v and its three neighbors w, x, y are shown. Each edge is replaced with a path of length 4.

Lemma 6.3 *Given an undirected planar graph G with maximum degree 3, and an integer $d \in \mathbb{Z}_{\geq 0}$. G has a dominating set of size at most d, if and only if the* MLkFL *instance Π defined above has a solution with maximum load 3, for $k = d + \sum_e \frac{|p(e)|-1}{3}$.*

Proof : If G has a dominating set D of cardinality d, we build a solution to Π as follows. Choose each vertex $v \in D$ as a facility that serves all its neighboring vertices, e.g. vertex v covering its three neighbors and vertex y covering its one neighbor in Figure 5b. Any such facility will bear a cost of 3. Now, consider the remaining nodes on each path $p(e)$. Every three consecutive vertices on this path can be covered by forming facility in the middle that serves its two neighbors and bears a cost of 2. As $|p(e)| = 3i + 1$, for some i, each such path needs $(|p(e)| - 1)/3$ facilities of cost at most 3 to have all the nodes on the path $p(e)$ covered, irrespective of whether or not the endpoints are in D. The three different cases have been depicted in Figure 5b. (i) Both endpoints $v, y \in D$. The middle vertex of $p(\{(v, y)\})$ is not covered by stars rooted at either of v or y and is thus covered by a singleton star of cost 0. (ii) Each of the edges $\{v, x\}$ and $\{v, w\}$ have exactly one endpoint $v \in D$ and the other endpoint $x, w \notin D$. Thus, there is a facility serving two of the clients on each of the paths $p(\{v, x\})$ and $p(\{v, w\})$. This facility also serves the endpoint not in the dominating set, i.e. x and w respectively. And (iii) Neither of the endpoints of the edge $\{w, z\}$ are in D. Thus, all the vertices on the corresponding path $p(\{w, z\})$ can be covered by facilities bearing cost 3. Note that in all the three cases above, we need exactly one facility to serve the clients on the paths $p(e)$.

Conversely, assume that the clients of Π can be served by k facilities bearing cost at most 3 each. We show that G has a dominating set of size $k - \sum_e (|p(e)| - 1)/3$. Note that each facility of cost at most 3 in Π has one of the following structures: (a) a single facility serving one or zero clients, (b) a facility serving its two neighbors, or (c) a facility covering its three neighbors. Consider the internal vertices of $p(e)$, i.e. all the vertices of $p(e)$, minus the two endpoints. There are at least $(|p(e)| - 1)/3$ facilities of type (b) or (a) needed to serve these clients. This bound is true even if two facilities of type (c) serve one client each from the two internal vertices of distance 1 from the two vertices at the endpoint of path $p(e)$. Delete all the facilities located at the internal vertices of all the paths; these will be of type (b) or (a). Let S be the set of remaining facilities (which we can assume are all of type (c)). Since there are at least $\sum_e (|p(e)| - 1)/3$ facilities removed, $|S| \leq k - \sum_e (|p(e)| - 1)/3$. It remains to show that S forms a dominating set in G.

Consider an edge $e = \{u, v\}$ in G and the corresponding path $p(e)$ in H. If neither of u or v have facilities located at them (in S), then we show that u, v are adjacent to vertices that have facilities located at them in S. Suppose neither u nor v have facilities located at them, then they are served by stars of type (b) or (a) in H. Let $e' = \{u, u'\}$ and $e'' = \{v, v'\}$ be two other edges incident to u and v (in G), respectively. Covering $p(e')$ and $p(e'')$ by $(|p(e')| - 1)/3$ and $(|p(e'')| - 1)/3$ facilities of type (b), respectively, each will have two vertices at the end left over that need to be covered by a facility of type (c), i.e. $u', v' \in S$; thus S is a dominating set. ■

For a YES instance (G, d) of dominating set, Π can have k facilities located covering $\mathcal{C}$, each bearing a maximum cost of 3, as shown above. In the NO case, the cardinality of the dominating set of G is at least $d + 1$, and thus Π needs at least $k + 1$ facilities of cost at most 3 to serve all the clients. To serve $\mathcal{C}$ with k facilities, we either need to include a facility located at an internal vertex serving three clients, instead of two, or a facility located at an endpoint of $p(e)$ serving four clients, instead of three. In either case, the cost of the star will be at least 4. This completes the proof of Theorem 6.2. ■

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.

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

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 $\Omega(k/logk)$ 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 $d(i,j) \leq B_{\text{LP}}$ is connected.

Theorem 6.4 *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 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 $1 \leq i \leq m$.

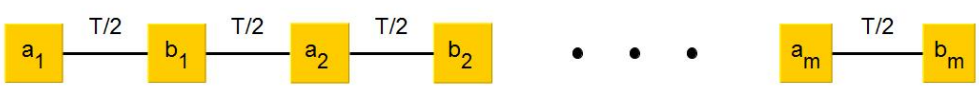


Figure 6: Example showing bad integrality gap for the configuration LP in line metric

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.) ■

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 [20]. 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 ot 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. (a PTAS for line metrics and a QPTAS for trees). We also proved that the problem is APX-hard on Euclidean metrics and it is resilient against a host of standard algorithmic techniques such as local search and LP based relaxations. Several questions remain open, the most prominent one being to find a true (not bicriteria) approximation for the problem on general metrics. Some (perhaps) easier intermediate steps would be to find such a true approximation algorithm for the problem on some restricted familties of graphs that are more general than a line. Getting a PTAS for trees is another 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 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] Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 434–444. ACM, 1988.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Amin Jorati. *Approximation Algorithms for Min-max Vehicle Routing Problems*. PhD thesis, University of Alberta, Department of Computing Science, 2013.

[14] 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.

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

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

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

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

[19] 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.

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