The Container Selection Problem

Viswanath Nagarajan[1], Kanthi K. Sarpatwar[*2], Baruch Schieber[2], Hadas Shachnai[†3], and Joel L. Wolf[2]

1 **University of Michigan, Ann Arbor, MI, USA**
viswa@umich.edu
2 **IBM T.J. Watson Research Center, Yorktown Heights, NY, USA**
sarpatwa@us.ibm.com, sbar@us.ibm.com, jlwolf@us.ibm.com
3 **Technion, Haifa, Israel**
hadas@cs.technion.ac.il

Abstract

We introduce and study a network resource management problem that is a special case of *non-metric k-median*, naturally arising in cross platform scheduling and cloud computing. In the continuous d-dimensional *container selection problem*, we are given a set $\mathcal{C} \subset \mathbb{R}^d$ of input points, for some $d \geq 2$, and a budget k. An input point p can be assigned to a "container point" c only if c dominates p in every dimension. The assignment cost is then equal to the ℓ_1-norm of the container point. The goal is to find k container points in $\mathbb{R}^d$, such that the total assignment cost for all input points is minimized. The discrete variant of the problem has one key distinction, namely, the container points must be chosen from a given set $\mathscr{F}$ of points.

For the continuous version, we obtain a *polynomial time approximation scheme* for any fixed dimension $d \geq 2$. On the negative side, we show that the problem is NP-hard for any $d \geq 3$. We further show that the discrete version is significantly harder, as it is NP-hard to approximate without violating the budget k in any dimension $d \geq 3$. Thus, we focus on obtaining bi-approximation algorithms. For $d = 2$, the bi-approximation guarantee is $(1+\epsilon, 3)$, i.e., for any $\epsilon > 0$, our scheme outputs a solution of size $3k$ and cost at most $(1+\epsilon)$ times the optimum. For fixed $d > 2$, we present a $(1+\epsilon, O(\frac{1}{\epsilon} \log k))$ bi-approximation algorithm.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases non-metric k-median, geometric hitting set, approximation algorithms, cloud computing, cross platform scheduling.

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

This paper introduces and studies the *container selection* problem, a special case of the non-metric k-median problem [13]. This network resource management problem naturally occurs in virtualized distributed computer environments, the goal being to maximize resource utilization. This environment may consist, e.g., of a private cloud [2], or a collection of in-house, physical computer processors employing a cluster manager such as Mesos [10] or YARN [16].

We describe and motivate the container selection problem as follows. The input points correspond to *tasks*, each of which can be described in terms of multiple resource requirements.

* Work done when the author was a student at the University of Maryland (College Park) and was supported by NSF grant CCF-1217890
† This work was partly carried out during a visit to DIMACS supported by the National Science Foundation under grant number CCF-1144502.

These dimensions typically include both CPU and memory, sometimes also network and I/O bandwidth. The tasks are then placed and executed in *virtual containers*, and of course each task must "fit" into its assigned container. For a variety of reasons, including ease of selection, maintenance and testing, it is important to create only a modest number k of container sizes. Amazon's EC2 cloud offering [1], for example, allows its customers to choose from $k = 13$ standard "instance types". The goal is to select k container sizes so that the aggregate resource usage (when each task is assigned its "smallest" dominating container) is minimized. We use the (normalized) sum of resources as the aggregate resource usage of a container. In these applications, the container sizes are usually determined in advance, before the actual tasks arrive: so suitably massaged historical task data is used as input. We refer the reader to [17] for more details.

Formally, an instance of the *continuous container selection* problem consists of a set of input points $\mathcal{C}$ in a d-dimensional space $\mathbb{R}^d$, and a budget k. We say that a point $c(c_1, c_2, \ldots, c_d)$ *dominates* (or, *contains*) a point $p(x_1, x_2, \ldots, x_d)$ if $x_i \leq c_i$, for all $i \in [d]$. The cost of assigning any input point p to a container point $c(c_1, c_2, \ldots, c_d)$ is the ℓ_1-norm of the container point, i.e, $c_1 + c_2 + \ldots + c_d$, if c dominates p; else, the assignment cost is ∞. The goal is to choose a set $S \subseteq \mathbb{R}^d$ of k container points, such that each input point is assigned to a container point in S, and the total assignment cost is minimized. In the *discrete* version of the problem, we have a further restriction that $S \subseteq \mathscr{F}$, where $\mathscr{F} \subseteq \mathbb{R}^d$ is an arbitrary, but finite, subset of points in the space. This problem variant is motivated by the fact that each container must itself "fit" into at least one physical processing node, or by the fact that certain resources (memory, for instance) are only allocated in fixed increments.

Related work. Clustering problems such as k-median, k-center, and k-means have received considerable attention in recent decades [11, 12, 4] (and references therein). Below, we only discuss the highlights directly relevant to our work. Our problem is a special case of the *non-metric k-median* problem. It also bears some similarity to the *Euclidean k-median* problem under the ℓ_1-norm metric. However, this similarity cannot be leveraged due to the "non-metric" characteristics of our problem. There is a $(1 + \epsilon, (1 + \frac{1}{\epsilon}) \ln n)$ bi-approximation algorithm for non-metric k-median [13], which finds a solution whose cost is within a $(1 + \epsilon)$ factor of optimal, for any constant $\epsilon > 0$, while using at most $k(1 + \frac{1}{\epsilon}) \ln n$ centers. The paper [13] also shows, using a reduction from the set cover problem, that these guarantees are the best one can hope for. On the other hand, the metric variant of the k-median problem is known to have small constant-factor approximation algorithms, with no violation of k. The best known ratio $2.611 + \epsilon$ is due to [7]. For the Euclidean k-median problem, which is a special case of metric k-median, there is a *polynomial time approximation scheme (PTAS)* [5].

Ackermann et al. [3] obtain PTAS for the non-metric k-median problem assuming that the following property holds: the corresponding 1-median problem can be approximated within a $1 + \epsilon$ factor by choosing a constant size sample and computing the optimal 1-median of such a sample. However, we note that the container selection problem does not satisfy this property. Indeed, consider a simple 1-dimensional instance with $n - 1$ points close to origin, and one point far away from origin. Clearly, with high probability, any constant size sample will, not contain the point far away from origin. An optimal 1-median of such a sample would in turn be infeasible for the original instance.

Our contribution. As noted above, the container selection problem is a special case of non-metric k-median, which is inapproximable unless we violate k significantly [13]. However, our problem still has sufficient geometric structure. This structure allows us to obtain near optimal algorithms that, in the case of continuous container selection, do not violate k, and in the discrete case, violate k mildly. In particular, we show that:

- the *continuous container selection problem* admits a PTAS, for any fixed d.
  On the negative side, we show that the problem is NP-hard for $d \geq 3$.
- the *discrete* variant (for $d \geq 3$) is NP-hard to approximate within *any* guarantee if the budget k is not violated. On a positive note, we obtain constant factor bi-approximation algorithms for this variant. For any constant $\epsilon > 0$, the guarantees are $(1+\epsilon, 3)$, for $d = 2$, and $(1+\epsilon, O(\frac{d}{\epsilon} \log dk))$, for any $d \geq 3$. The latter result is an improvement over the bi-approximation that follows from non-metric k-median [13] as long as $k = o(\log^{O(1)} n)$, $d = o(\frac{\log n}{\log \log n})$.

Techniques and outline. Our PTAS for the continuous problem (Section 2) relies on showing the existence of a near-optimal solution, where every container point lies on one among a constant number of rays through the origin. Ensuring this structure costs us a $1+\epsilon$ factor in the approximation ratio. The algorithm itself is then a dynamic program which optimally solves such a "restricted" container selection problem. A seemingly simpler approach is to use the near-optimal structure, where every container point lies on a grid with $O(\log n)$ geometrically spaced values in each dimension; however, this is not directly useful, as we do not know an exact algorithm for the resulting sub-problem.

The flexibility of using container points in the continuous space is essential − not just for our algorithm, but for any approach: we show the discrete version is NP-hard to approximate to any factor when $d \geq 3$. The reduction (Section 4) is from a restricted version of planar vertex cover [8], and in fact shows that even testing feasibility is NP-hard. We also reduce the discrete container selection problem to the continuous version (not approximation preserving), which proves its NP-hardness when $d \geq 3$.

We obtain two different algorithms for the discrete container selection problem, both of which provide bi-approximation guarantees. The first algorithm (Section 3.1) is specialized to dimension two and is a $(1+\epsilon, 3)$-approximation. The main idea here is a partitioning of $\mathbb{R}^2_+$ into $O(\log n)$ "cells", where all points in a cell have roughly the same ℓ_1-norm, thus allowing to decouple "local assignments" within a single cell, and "distant assignments" from one cell to another. This partitioning uses the definition of rays from the algorithm for the continuous problem. (Using a more standard partitioning yields $O(\log^2 n)$ cells which is too large for a polynomial-time algorithm.) The algorithm then uses enumeration to handle distant assignments and a dynamic-program for the local assignments. This decoupling necessitates the violation in the bound k.

The second algorithm for the discrete version (Section 3.2) works for any dimension d and yields a $(1+\epsilon, O(\frac{d}{\epsilon} \log dk))$-approximation. This is based on the natural linear programming relaxation used even for the non-metric k-median problem [13]. However, we obtain a sharper guarantee in the violation of k, using the geometry specific to our setting. In particular, we show an LP-based reduction to hitting-set instances having VC-dimension $O(d)$. Then our algorithm uses the well-known result of [9, 6] for such hitting-set instances. We note that a constant bi-approximation algorithm for $d = 2$ also follows from this approach, using a known $O(\frac{1}{\epsilon})$-size ϵ-net construction for "pseudo-disks" [14]. However, the constant obtained here is much larger than our direct approach.

Remark. There is also a *quasi-polynomial* time approximation scheme (no violation of the bound k) for the discrete container selection problem in dimension $d = 2$. This is based on a different dynamic program (details deferred to the full version). However, this approach does not lead to any non-trivial approximation ratio in polynomial time. We leave open the possibility of a polynomial-time $O(1)$-approximation algorithm for this problem ($d = 2$).

Notation. For integers $a < b$, we use $[b] := \{1, 2, \cdots b\}$ and $[a, b] := \{a, a+1, \ldots, b\}$. All co-ordinates of input points are assumed to be non-negative. A point $c(c_1, c_2, \ldots, c_d) \in \mathbb{R}^d$

dominates (or, *contains*) another $p(x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ if, for all $i \in [d]$, $x_i \leq c_i$. By $p \prec c$, we mean c dominates p. Two points p_1 and p_2 are called *incomparable* if $p_2 \not\prec p_1$ and $p_1 \not\prec p_2$. The ℓ_1-norm of a point $c(c_1, c_2, \ldots, c_d)$ is denoted by $\|c\|$, i.e., $\|c\| = c_1 + c_2 \ldots + c_d$. For a subset of container points, S, we denote the total cost of assignment by $\mathsf{cost}(S)$. The cartesian product of two sets A and B is denoted by $A \times B$.

2 The Continuous Container Selection Problem

In this section, we describe a polynomial time approximation scheme for the continuous container selection problem. We start with a formal definition.

▶ **Definition 1** (continuous container selection)**.** In an instance of the problem, we are given a set of input points $\mathcal{C}$ in $\mathbb{R}^d_+$ and a budget k. The goal is to find a subset S of k container points in $\mathbb{R}^d_+$, such that the following cost is minimized.

$$\underset{\substack{S \subseteq \mathbb{R}^d \\ |S| \leq k}}{\mathsf{Min}} \sum_{p \in \mathcal{C}} \underset{\substack{c \in S \\ p \prec c}}{\mathsf{Min}} \|c\|$$

We describe the algorithm for $d = 2$ in Section 2.1 and subsequently, in Section 2.2 we extend this to dimension $d > 2$.

2.1 The two dimensional container selection problem

We denote the set of input points by $\mathcal{C} = \{p_i(x_i, y_i) : i \in [n]\}$. Let S_{opt} denote an optimal set of k container points. Let $X = \{x_i : i \in [n]\}$ and $Y = \{y_i : i \in [n]\}$. It is an easy observation that $S_{opt} \subseteq X \times Y$. We call $X \times Y$ the set of potential container points and denote it by $\mathscr{F} = \{c_j(u_j, v_j) : j \in [m]\}$, where $m \leq n^2$.

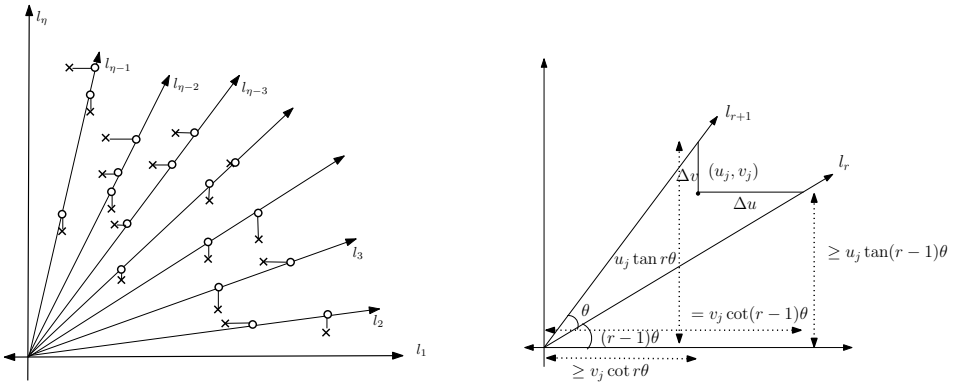
Algorithm outline. Given an instance of the problem, we transform it into an easier instance where all the chosen container points must lie on a certain family of rays. The number of rays in this family will be bounded by a constant that depends on ϵ, where $1 + \epsilon$ is the desired approximation ratio. Subsequently, we show that the restricted problem can be solved in polynomial time using a dynamic program.

Transformation of container points. Fix a constant $\theta \approx \frac{\epsilon}{2} \in (0, \frac{\pi}{4}]$, such that $\eta = \frac{\pi}{2\theta}$ is an integer. Define the following lines $l_r \equiv y \cos(r-1)\theta - x \sin(r-1)\theta = 0$, for $r \in [\eta + 1]$. We define the following transformation of any point $c_j(u_j, v_j) \in \mathscr{F}$ to construct the set of potential container points $\mathscr{F}^T$. If c_j lies on the line l_r, for some $r \in [\eta]$, then $c_j^T = c_j$. Otherwise, c_j is contained in the region bounded by the lines l_r and l_{r+1}, for some $r \leq \eta$. Now define two points $c_j^u(u_j + \Delta u, v_j)$ and $c_j^v(u_j, v_j + \Delta v)$, such that c_j^u is on l_r and c_j^v is on l_{r+1}. Now, the transformed point can be defined as follows:

$$c_j^T = \begin{cases} c_j^u, & \text{if } \Delta u \leq \Delta v \\ c_j^v, & \text{otherwise} \end{cases}$$

Figure 1a illustrates this transformation. We emphasize that this transformation is only performed on the potential container points $\mathscr{F}$. The input points $\mathcal{C}$ themselves are unchanged. Under this transformation the optimal solution is preserved within an approximation factor of $(1 + \epsilon)$.

▶ **Lemma 2.** *For instance $\mathcal{I} = (\mathcal{C}, k)$, let $S_{opt} = \{o_1, o_2, \ldots, o_k\}$ be an optimal solution. Further, let $S^T_{opt} = \{o_1^T, o_2^T, \ldots, o_k^T\} \subseteq \mathscr{F}^T$ be the set of transformed points corresponding to S_{opt}. Then, S^T_{opt} is a feasible solution to $\mathcal{I}$ and $\mathsf{cost}(S^T_{opt}) \leq (1 + \epsilon)\mathsf{cost}(S_{opt})$.*



(a) Transformation of the container points **(b)** Computing the error in transformation

Figure 1 Continuous container selection problem.

Proof. Recall that $\eta = \frac{\pi}{2\theta}$ and $\theta \approx \frac{\epsilon}{2}$. The feasibility of S^T_{opt} follows from the observation that if a point $p_i \in \mathcal{C}$ is dominated by a container $o_i \in S_{opt}$, it is also dominated by the point o^T_i. We now argue that $\mathsf{cost}(S^T_{opt}) \leq (1+\epsilon)\mathsf{cost}(S_{opt})$. It suffices to show that for every point $o_j = (u_j, v_j)$, $u^T_j + v^T_j \leq (1+\epsilon)(u_j + v_j)$, where $o^T_j = (u^T_j, v^T_j)$. The claim holds trivially in the case where o_j lies on a line l_r, for $r \in [1, 2, \ldots, \eta+1]$. Hence, assume that o_j lies in the region bounded by the two lines l_r and l_{r+1}, where $r \in [1, 2, \ldots, \eta]$. Further, let $o^u_j = (u_j + \Delta u, v_j)$ and $o^v_j = (u_j, v_j + \Delta v)$, be the points on lines l_r and l_{r+1} respectively. By geometry (refer to Figure 1b), we have the following equations:

$$\Delta u \leq v_j \left(\frac{\cos(r-1)\theta}{\sin(r-1)\theta} - \frac{\cos r\theta}{\sin r\theta} \right) = v_j \frac{\sin\theta}{\sin r\theta \sin(r-1)\theta} \tag{1}$$

$$\Delta v \leq u_j \left(\frac{\sin r\theta}{\cos r\theta} - \frac{\sin(r-1)\theta}{\cos(r-1)\theta} \right) = u_j \frac{\sin\theta}{\cos r\theta \cos(r-1)\theta} \tag{2}$$

Let $\Delta = \min(\Delta u, \Delta v)$. From Equations 1 and 2, we have,

$$(u_j + v_j)\sin\theta \geq \Delta(\sin r\theta \sin(r-1)\theta + \cos r\theta \cos(r-1)\theta) \;=\; \Delta\cos\theta.$$

$$\text{So } \Delta \leq (u_j + v_j)\tan\theta \leq (u_j + v_j)(2\theta) = (u_j + v_j)\epsilon. \tag{3}$$

Now, the claim follows from Equation 3 and the fact that $u^T_j + v^T_j = (u_j + v_j) + \Delta$. ◀

In Section 2.3, we show that the following restricted problem can be solved in polynomial time (by dynamic programming), for any fixed dimension $d \geq 2$.

▶ **Definition 3** (restricted container selection)**.** For a constant $\eta \geq 0$, let $L_d = \{l_1, l_2, \ldots, l_\eta\}$ be a given family of η rays in $\mathbb{R}^d_+$. The input is a set of points $\mathcal{C} \subseteq \mathbb{R}^d_+$, a set of potential container points $\mathscr{F}$ that lie on the lines in L_d and a budget k. The goal is to find a subset $S \subseteq \mathscr{F}$ with $|S| \leq k$ such that $\mathsf{cost}(S)$ is minimized.

By Lemma 2, the 2D continuous container selection problem reduces to this restricted problem, at a $(1+\epsilon)$-factor loss. So we obtain a PTAS for the 2D continuous container selection problem.

2.2 Continuous container selection in dimension $d > 2$

We now consider the container selection problem in higher, but fixed, dimensions. Formally, an instance, $\mathcal{I} = (\mathcal{C}, k)$, of the d-dimensional container selection problem consists of a set of input points, $\mathcal{C} = \{p_i(x_1^i, x_2^i, \ldots, x_d^i) : i \in [n]\}$ and a budget k.

Potential container points. For each dimension $j \in [d]$, we define $X_j = \{x_j^i : i \in [n]\}$, as the set of j^{th} coordinates of all input points. An easy observation is that any container point chosen by any optimal solution must belong to $\mathscr{F} = X_1 \times X_2 \times \ldots \times X_d = \{c_i(u_1^i, u_2^i, \ldots, u_d^i) : i \in [m]\}$ where, $m \leq n^d$.

Algorithm outline. As in the two dimensional case, the main idea is a reduction to the following restricted problem. An instance is $\mathcal{I} = (\mathcal{C}, k, L_d)$ where $\mathcal{C}$ is a set of input points in $\mathbb{R}^d$, k is an integer and L_d is a family of rays in $\mathbb{R}_+^d$ with $|L_d| = O_d(1)$. The goal is to choose k container points that lie on the rays in L_d, such that the total assignment cost of $\mathcal{C}$ is minimized.

Transformation of container points. Fix a constant $\theta \approx \frac{\epsilon}{2} \in (0, \frac{\pi}{4}]$, such that $\eta = \frac{\pi}{2\theta}$ is an integer. In order to construct L_d, we use the recursive procedure described in Algorithm 1. Let $\bar{u}_i$ denote the i^{th} unit vector ($i \leq d$), i.e., $\bar{u}_i$ is a 0-1 vector with value 1 at the i^{th} coordinate and 0 elsewhere. Starting from the family L_2 of rays in two dimensions (using the transformation in Section 2), we add one dimension at a time and construct the corresponding families for higher dimensions. In the recursive step, we start with the family L_{r-1} and observe that each of these rays will induce a 2-D plane in r-dimensions. Then, we use the two dimensional construction to handle the extra dimension. Observe that $|L_d| \leq (\pi/\theta)^d = O(1)$ for any fixed θ and d.

Algorithm 1 Construction of the family of lines in r-dimensions: L_r

```
let ū_1, ū_2, ..., ū_r be the unit vectors along the axis lines
if r = 2 then return equiangular rays in R^2_+ from Section 2 (see also Figure 1)
construct the family L_{r-1} in R^{r-1}_+ recursively.
initiate: L_r ← ∅
for all ℓ ∈ L_{r-1} do
    let ℓ̄ be the unit vector along the line ℓ
    consider the (two dimensional) plane Π_ℓ formed by the vectors ū_r and ℓ̄
    let Q_ℓ be the family of rays obtained by applying the 2D transformation in Section 2
    to the plane Π_ℓ
    L_r ← L_r ∪ Q_ℓ
end for
return L_r
```

Algorithm 2 describes a recursive procedure to transform a point $c(u_1, u_2, \ldots, u_d) \in \mathscr{F}$ to a point c^T that lies on some line in L_d. The idea is as follows: for any $r \geq 3$, first recursively transform the point $c_{r-1}(u_1, u_2, \ldots, u_{r-1}) \in \mathbb{R}^{r-1}$ into a point $c_{r-1}^T(u'_1, u'_2, \ldots, u'_{r-1})$ that lies on some line $\ell \in L_{r-1}$. Now, consider the point $c'_r(u'_1, u'_2, \ldots, u'_{r-1}, u_r)$, where u_r is the r^{th} coordinate of the original point c. The point c'_r lies on the 2D plane spanned by $\bar{\ell}$, the unit vector along the line ℓ, and $\bar{u}_r$. Using the 2D transformation we move c'_r to a point c_r^T that lies on some line in L_r.

▶ **Lemma 4.** *For any $\theta = \frac{\epsilon}{2} \in (0, \frac{1}{2d-2}]$ and point $c(u_1, u_2, \ldots, u_d) \in \mathscr{F}$, applying Algorithm 2, we obtain $c^T = (u_1^T, u_2^T, \ldots, u_d^T)$ where $c \prec c^T$ and:*

$$\|c^T\| \leq (1 + 2(d-1)\epsilon)\|c\|.$$

Algorithm 2 The transformation of $c_r = (u_1, u_2, \dots u_r)$ onto L_r, $r \leq d$

1: if $r = 2$ then use the 2D transformation from the Section 2 (see also Figure 1)
2: $c_{r-1} \leftarrow (u_1, u_2, \dots, u_{r-1})$
3: recursively transform c_{r-1} into a point on some line ℓ in L_{r-1} and compute the transformed point $c^T_{r-1} = (u'_1, u'_2, \dots, u'_{r-1})$
4: $c'_r \leftarrow (u'_1, u'_2, \dots, u'_{r-1}, u_r)$, which lies on the plane Π_ℓ spanned by $\bar{u_r}$ and $\bar{\ell}$
5: let Q_ℓ denote the lines on plane Π_ℓ from Algorithm 1 step 8.
6: use the 2D transformation (Section 2) on plane Π_ℓ to move c'_r onto a line in Q_ℓ and obtain $c^T_r = (u^T_1, u^T_2, \dots, u^T_{r-1}, u^T_r)$
7: **return** c^T_r

Proof. It is straightforward to see $c \prec c^T$. Using induction we will show that

$$\|c^T_r\| \leq (1+\epsilon)^{r-1}\|c_r\|$$

The base case $r = 2$ follows from Lemma 2. Now consider $r \geq 3$ and assume the statement for $r-1$. In Algorithm 2, c^T_r is obtained by transforming the point c'_r in the 2D plane Π_ℓ. Note that c'_r has coordinates $\sqrt{(u'_1)^2 + (u'_2)^2 + \ldots + (u'_{r-1})^2}$ and u_r in plane Π_ℓ. Hence, as shown in Lemma 2, we can obtain the following:

$$\begin{aligned} u^T_1 + u^T_2 + \ldots + u^T_{r-1} + u^T_r &\leq (1+\epsilon)(\sqrt{(u'_1)^2 + (u'_2)^2 + \ldots + (u'_{r-1})^2} + u_r) \\ &\leq (1+\epsilon)(u'_1 + u'_2 + \ldots + u'_{r-1} + u_r) \end{aligned} \tag{4}$$

By the inductive hypothesis, $u'_1 + u'_2 + \ldots + u'_{r-1} = \|c^T_{r-1}\| \leq (1+\epsilon)^{r-2}\|c_{r-1}\|$, i.e.

$$u'_1 + u'_2 + \ldots + u'_{r-1} \leq (1+\epsilon)^{r-2}(u_1 + u_2 + \ldots + u_{r-1}) \tag{5}$$

Using Equations 4, 5, we have

$$\begin{aligned} u^T_1 + u^T_2 + \ldots + u^T_{r-1} + u^T_r &\leq (1+\epsilon)((u'_1 + u'_2 + \ldots + u'_{r-1}) + u_r) \\ &\leq (1+\epsilon)((1+\epsilon)^{r-2}(u_1 + u_2 + \ldots + u_{r-1}) + u_r) \\ &\leq (1+\epsilon)^{r-1}(u_1 + u_2 + \ldots + u_r) \end{aligned}$$

Now since $(d-1)\epsilon \leq 1$, using $r = d$ above, $\|c^T\| \leq (1+\epsilon)^{d-1}\|c\| \leq (1 + (2d-2)\epsilon) \cdot \|c\|$. ◀

For any $\epsilon' > 0$, setting $\epsilon = \frac{\epsilon'}{2(d-1)}$, we can restrict the loss to a $(1+\epsilon')$ factor. Thus, we have reduced the original instance to a restricted instance, where the potential container points lie on a family with a constant number of lines. Using the exact algorithm for this problem (Section 2.3) we obtain:

▶ **Theorem 5.** *There is a PTAS for continuous container selection in fixed dimension d.*

2.3 Algorithm for restricted container selection

Here we provide an exact algorithm for the restricted container selection problem (Definition 3). We need the following notion of a profile of a given subset of container points.

Profile of a subset. For a given line l_i and $S \subseteq \mathscr{F}$, let $c_i \in S$ be the container point on l_i with maximum ℓ_1-norm; if there is no such point then c_i is set to the origin. We define the *profile* of S, denoted by $\Pi(S)$, as the ordered tuple $(c_1, c_2, \dots, c_\eta)$. The *feasible* region of a profile $\Pi(S) = (c_1, c_2, \dots, c_\eta)$, denoted by $\mathsf{feas}(\Pi(S))$, is the set of those input points that are dominated by at least one of the points c_i, $i \in [\eta]$. We slightly abuse this notation and refer to the tuple itself as a profile, without any mention of S.

▶ Observation 6. The number of distinct profiles is at most $\left(\frac{|\mathscr{F}|}{\eta}\right)^{\eta}$.

Proof. Let n_i be the number of potential container points on the line l_i. The total number of distinct profiles is simply the number of ways of choosing the tuple $(c_1, c_2, \ldots, c_\eta)$, which is equal to $n_1 n_2 \ldots n_\eta \leq \left(\frac{\sum_{i=1}^{\eta} n_i}{\eta}\right)^{\eta} = \left(\frac{|\mathscr{F}|}{\eta}\right)^{\eta}$. ◀

For a given profile $\Pi = (c_1, c_2, \ldots, c_\eta)$, let c_m denote the profile point with maximum ℓ_1-norm, i.e., $c_m = \arg\max_{c_i} \|c_i\|$. Further, let $c'_m \prec c_m$ be some potential container point such that both the points are on the line l_m; if c'_m does not exist we set it to the origin. We define the *child profile* of Π corresponding to c'_m, denoted by $\mathsf{chld}(\Pi, c'_m)$, as the profile $(c_1, c_2, \ldots, c_{m-1}, c'_m, \ldots, c_\eta)$. A profile tuple could have multiple child profiles. The following observation is immediate from the definition of a child profile.

▶ Observation 7. Any profile tuple Π has at most $|\mathscr{F}|$ child profile tuples.

The DP variable. For every possible profile tuple $\Pi = (c_1, c_2, \ldots, c_\eta)$ and all budgets $k' \leq k$, define the dynamic program variable, $\mathscr{M}(\Pi, k')$ as the cost of an optimal solution $S \subseteq \mathsf{feas}(\Pi) \cap \mathscr{F}$, to assign all the input points in $\mathsf{feas}(\Pi)$, such that $|S| \leq k'$, and $c_i \in S$, for $i \in [\eta]$. The following lemma allows us to set up the dynamic program recurrence.

▶ **Lemma 8.** *Let $\Pi = (c_1, c_2, \ldots, c_\eta)$ be a profile with c_m as the point with maximum ℓ_1-norm. For a given child profile $\mathsf{chld}(\Pi, c'_m)$ of Π, let $n(c'_m) = |\mathsf{feas}(\Pi) \setminus \mathsf{feas}(\mathsf{chld}(\Pi, c'_m))|$. Then, for any $k' \geq 1$, the following holds.*

$$\mathscr{M}(\Pi, k') = \operatorname*{Min}_{c'_m} \left(\mathscr{M}(\mathsf{chld}(\Pi, c'_m), k' - 1) + n(c'_m)\|c_m\|\right)$$

Proof. We denote the optimal solution corresponding to the variable $\mathscr{M}(\Pi, k')$ by $S(\Pi, k')$. Firstly, note that, for any c'_m, the solution $S(\mathsf{chld}(\Pi, c'_m), k' - 1) \cup \{c_m\}$ is a feasible candidate for the computation of $\mathscr{M}(\Pi, k')$. Hence, we have

$$\mathscr{M}(\Pi, k') \leq \operatorname*{Min}_{c'_m} \left(\mathscr{M}(\mathsf{chld}(\Pi, c'_m), k' - 1) + n(c'_m)\|c_m\|\right) \tag{6}$$

Let l_m be the ray containing the point c_m. Further, let $q_0 = (0^d), q_1, \ldots, q_{j-1}, q_j = p_i$ be the container points, on l_m and in $S(\Pi, k)$, in the increasing order of ℓ_1-norm. Now, we set $q' = q_{j-1}$ and prove that the child profile corresponding to q' satisfies the following equation:

$$\mathscr{M}(\Pi, k') = \mathscr{M}(\mathsf{chld}(\Pi, q'), k' - 1) + n(q')\|c_m\|$$

To this end, we first observe that, without loss of generality, no point in $\mathsf{feas}\ (\mathsf{chld}(\Pi, q'))$ is assigned to c_m. Indeed, this follows from the fact that c_m is the container point with maximum cost and therefore, any point in the above feasible region can be assigned to some container point on the profile $\mathsf{chld}(\Pi, q')$ without increasing the solution cost. Further, any point in $\mathsf{feas}(\Pi) \setminus \mathsf{feas}(\mathsf{chld}(\Pi, q'))$ must be assigned to c_m, since it is the only potential container point that dominates these points. Now,

$$\begin{aligned} \mathscr{M}(\Pi, k') &= \mathscr{M}(\mathsf{chld}(\Pi, q'), k' - 1) + n(q')\|c_m\| \\ &\geq \operatorname*{Min}_{c'_m} \left(\mathscr{M}(\mathsf{chld}(\Pi, c'_m), k' - 1) + n(c'_m)\|c_m\|\right) \end{aligned} \tag{7}$$

From Equations 6 and 7, we have our lemma. ◀

Algorithm 3 describes the dynamic program.

Algorithm 3 Dynamic program for the restricted container selection problem

Input: Family of lines $L_d = \{l_1, l_2 \ldots, l_\eta\}$, input points $\mathcal{C}$, potential container points set $\mathscr{F}$ on L_d and a budget k

for all profile tuples Π (w.r.t L_d) and integers $k' \leq k$ **do**
  if $k' = 0$ **then**
    if $\Pi = ((0^d), (0^d), \ldots, (0^d))$ **then**
      $\mathscr{M}(\Pi, k') = 0$
    else
      $\mathscr{M}(\Pi, k') = \infty$
    end if
  else
    let c_m be the container point with maximum ℓ_1-norm in Π
    for all $c'_m \prec c_m$ such that both c_m and c'_m lie on the same line l_m **do**
      $n(c'_m) \leftarrow |\mathsf{feas}(\Pi) \setminus \mathsf{feas}(\mathsf{chld}(\Pi, c'_m))|$
      $f(c'_m) \leftarrow (\mathscr{M}(\mathsf{chld}(\Pi, c'_m), k' - 1) + n(c'_m)\|c_m\|)$
    end for
    $\mathscr{M}(\Pi, k') \leftarrow \min_{c'_m} f(c'_m)$
  end if
end for
return profile Π with least cost $\mathscr{M}(\Pi, k)$ such that $\mathcal{C} = \mathsf{feas}(\Pi)$.

3 The Discrete Container Selection Problem

In this section, we consider the discrete version of the container selection problem. We start with the problem definition.

▶ **Definition 9** (discrete container selection)**.** In an instance of the problem, $\mathcal{I} = (\mathcal{C}, \mathscr{F}, k)$, we are given a set of input points $\mathcal{C} \subset \mathbb{R}^d_+$, a set of potential container points $\mathscr{F} \subset \mathbb{R}^d_+$ and a budget k. The goal is to find a subset of container points $S \subseteq \mathscr{F}$, such that $|S| \leq k$ and the total assignment cost of all the input points, $\mathsf{cost}(S)$ is minimized.

This problem is considerably harder than the continuous version, as we show that there is no true approximation algorithm for this problem, unless $P = NP$, for $d \geq 3$. Hence, we look for bi-approximation algorithms, defined as follows. An (α, β) bi-approximation algorithm obtains a solution S, such that $|S| \leq \beta \cdot k$ and $\mathsf{cost}(S) \leq \alpha \cdot \mathsf{cost}(S_{opt})$.

▶ **Theorem 10** (two-dimenions)**.** *For $d = 2$, and any constant $\epsilon > 0$, there is a $(1 + \epsilon, 3)$-bi-approximation algorithm for the discrete container selection problem.*

▶ **Theorem 11** (higher-dimensions)**.** *For $d > 2$ and $\epsilon > 0$, there is a $(1 + \epsilon, O(\frac{d}{\epsilon} \log dk))$-bi-approximation algorithm for the discrete container selection problem.*

3.1 Two dimensional discrete container selection problem

Algorithm outline. The first step is to partition the plane into a logarithmic number of "cells", such that the ℓ_1-norms of points in a particular cell are approximately uniform. One standard way of doing this, where we create a two-dimensional grid with logarithmic number of lines in each dimension, fails because such a process would yield $\Omega(\log^2 n)$ cells. Our approach uses the *rays partitioning* idea. Given such a partitioning, we "guess" the

"good" cells that have any container points belonging to a fixed optimal solution. For each one of these good cells, we then pick two representative container points. These points are chosen such that if, in the optimal solution, an input point i outside a cell e is assigned to a container point inside e, at least one of the representative points in e dominates i. This enables us to make "local decisions" for each cell independently. We then solve this localized instance, using k more container points. Hence, in total we use $3k$ container points.

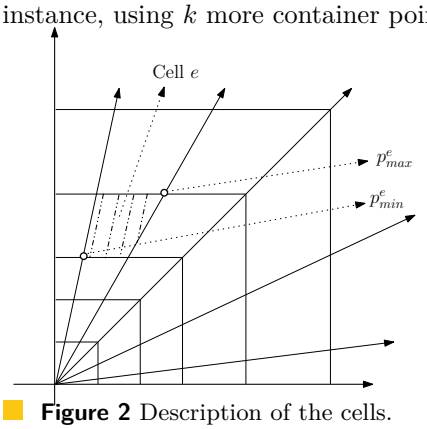


Figure 2 Description of the cells.

The algorithm. Choose $\delta = \frac{\epsilon}{11}$ such that $\frac{\pi}{4\delta} = \eta$ is an integer. We first use a simple scaling argument to bound the maximum to minimum ratio of ℓ_1-norms by $O(n)$. We guess the maximum norm container point p_{max} that is used in some fixed optimal solution (there are only $|\mathscr{F}|$ guesses) and delete all larger points from $\mathscr{F}$. Let p_{min} be the point in $\mathcal{C} \cup \mathscr{F}$ with minimum positive norm. We increase the x-coordinates of all the input points and the container points by $\frac{\delta}{n}\|p_{max}\|$ and then divide all the co-ordinates of all points by $\|p_{min}\|$.

▶ Observation 12. Let S_{opt} and S'_{opt} be the optimal solutions of a given instance before and after scaling respectively. $\|p_{min}\|\mathsf{cost}(S'_{opt}) \leq \mathsf{cost}(S_{opt})(1+\delta)$

Proof. Since all the points are increased and scaled uniformly, the feasibility is maintained. Further, we note that $\mathsf{cost}(S_{opt}) \geq \|p_{max}\|$ since our guess $p_{max} \in S_{opt}$. If the cost of assignment of any input point is C in the original instance, the new cost is equal to $(C + \frac{\delta}{n}\|p_{max}\|)/\|p_{min}\|$ and the lemma follows. ◀

From now on, we assume that all the points are scaled as above and therefore $\|p_{min}\| = 1$ and $\|p_{max}\| \leq \frac{n}{\delta}$. Let $t = \log_{1+\delta}\|p_{max}\|$ and define the following families of rays.

$$\mathcal{L}_1 = \{x\sin(r\delta) - y\cos(r\delta) = 0 : r \in [0,\eta)\} \quad \mathcal{L}_3 = \{y = (1+\delta)^i : i \in [0,t]\}$$

$$\mathcal{L}_2 = \{x\sin(r\delta) - y\cos(r\delta) = 0 : r \in [\eta,2\eta]\} \quad \mathcal{L}_4 = \{x = (1+\delta)^i : i \in [0,t]\}$$

Cells. We define the notion of cell as exemplified in the Figure 2. A *cell* is a quadrilateral formed with the following bounding lines: either, two consecutive lines in $\mathcal{L}_1$ and two consecutive lines in $\mathcal{L}_4$, or, two consecutive lines in $\mathcal{L}_2$ and two consecutive lines in $\mathcal{L}_3$. The number of cells formed is at most $(2\eta+1)t = O(\log n)$

▶ **Lemma 13.** *For a given cell e, let p^e_{min} and p^e_{max} be the points of minimum and maximum cost, respectively. Then,*

$$\|p^e_{max}\| \leq (1+\epsilon)(\|p^e_{min}\|)$$

Proof. Without loss of generality, let e be formed by lines $y = (1+\delta)^i$, $y = (1+\delta)^{i+1}$, $x\sin\theta - y\cos\theta = 0$ and $x\sin(\theta+\delta) - y\cos(\theta+\delta) = 0$, where $\theta \geq \frac{\pi}{4}$. Clearly, as shown in Figure 2, we have

$$p^e_{min} = ((1+\delta)^i\cot(\theta+\delta), (1+\delta)^i)$$

$$p^e_{max} = ((1+\delta)^{i+1}\cot\theta, (1+\delta)^{i+1})$$

$$\begin{aligned}
\frac{\|p^e_{max}\|}{\|p^e_{min}\|} &= \frac{(1+\delta)^{i+1}(1+\cot\theta)}{(1+\delta)^i(1+\cot(\theta+\delta))} = (1+\delta)\frac{(\sin\theta+\cos\theta)\sin(\theta+\delta)}{(\sin(\theta+\delta)+\cos(\theta+\delta))\sin\theta} \\
&= (1+\delta)\frac{\sin\theta\sin(\theta+\delta)+\cos\theta\sin(\theta+\delta)}{\sin(\theta+\delta)\sin\theta+\cos(\theta+\delta)\sin\theta} \\
&= (1+\delta)\left(1+\frac{\cos\theta\sin(\theta+\delta)-\cos(\theta+\delta)\sin\theta}{\sin(\theta+\delta)\sin\theta+\cos(\theta+\delta)\sin\theta}\right) \\
&= (1+\delta)\left(1+\frac{\sin\delta}{\sin(\theta+\delta)\sin\theta+\cos(\theta+\delta)\sin\theta}\right) \leq (1+\delta)\left(1+\frac{\sin\delta}{\sin^2\theta}\right) \\
&\leq (1+\delta)(1+2\delta) = (1+3\delta+2(\delta)^2) \leq (1+\epsilon)
\end{aligned}$$

We note that the second last inequality follows from the fact that $\sin^2\theta \geq \sin^2\frac{\pi}{4} \geq \frac{1}{2}$. ◀

Representative points. For a given optimal solution, a cell is good if at least one container point is chosen from it (we break the ties between two cells sharing an edge arbitrarily). Since, there are $O(\log n)$ cells, there are a polynomial number of good-bad classifications. Therefore, we can try out all possible configurations and assume that we know which cells are good. For each good cell e, let p^e_x be the container point with maximum x-coordinate and p^e_y the one with maximum y-coordinate. We define the set of representative points, $\mathcal{R} = \{\ p^e_x,\ p^e_y : \forall\ e \text{ good cell }\}$. Clearly $|\mathcal{R}| \leq 2k$. We will show (in Lemma 15) that any input point that is not assigned to a "local container" (one in the same cell) in the optimal solution, can be re-assigned to some point of $\mathcal{R}$ at approximately the same cost.

Localized container selection problem. In an instance of the *localized container selection problem*, $(\mathcal{C}, \mathscr{F}_1, \mathscr{F}_2, k)$, we are given a set of input points $\mathcal{C}$, a set of potential container points $\mathscr{F}_1$, a set of pre-chosen container points $\mathscr{F}_2$ and a budget k. Moreover, for each cell e, the points in $\mathscr{F}_1 \cap e$ are all incomparable to each other. For a cell e, let $\Delta^e_{max} = \underset{p\in\mathscr{F}_1\cap e}{\mathsf{Max}} \|p\|$, be the maximum ℓ_1-norm of any container point in e. The cost of assignment of any input point to any point, in $\mathscr{F}_1 \cap e$, is uniform and equal to Δ^e_{max}. The cost of assignment of an input point to a container point $c \in \mathscr{F}_2$ is $\|c\|$. Further, any input point p in the cell e, can only be assigned to:

- a container point $c \in \mathscr{F}_2$ such that $p \prec c$, or
- a container point $c \in \mathscr{F}_1$ such that c belongs to e and $p \prec c$.

Given an instance of the discrete container selection problem, $\mathcal{I} = (\mathcal{C}, \mathscr{F}, k)$, we construct the following instance of the *localized container selection problem*, $\mathcal{I}' = (\mathcal{C}, \mathscr{F}_1, \mathscr{F}_2, k)$.

▶ Construction 14. The input point set $\mathcal{C}$, remains the same and $\mathscr{F}_2$ is the set of representative points, i.e., $\mathscr{F}_2 = \mathcal{R}$. $\mathscr{F}_1$ is constructed as follows: starting with $\mathscr{F}_1 = \mathscr{F} \setminus \mathcal{R}$, while there are two points p and p' in $\mathscr{F}_1$ that belong to same cell e and $p \prec p'$, delete p from $\mathscr{F}_1$.

▶ **Lemma 15.** *For a given instance of the discrete container selection problem, $\mathcal{I} = (\mathcal{C}, \mathscr{F}, k)$, with the optimal solution cost OPT, the corresponding localized container selection instance $\mathcal{I}' = (\mathcal{C}, \mathscr{F}_1, \mathscr{F}_2, k)$ has an optimal cost of at most $(1+\epsilon)OPT$.*

Proof. Suppose S is an optimal solution for the instance $\mathcal{I}$. We iteratively construct a solution, S', for the instance $\mathcal{I}'$. Initiating $S' = \phi$, we add exactly one container point for every container point $c \in S$ in the following way: let c belong to a cell e. If $c \in \mathscr{F}_1$, then we add c to S'; otherwise, we add some $c' \in \mathscr{F}_1 \cap e$, such that $c \prec c'$, which must exist by Construction 14. Clearly $|S'| \leq |S| \leq k$. We show that S' is a feasible solution, with a cost at most $(1+\epsilon)OPT$, for the instance $\mathcal{I}'$.

Consider an input point p that is assigned to some container point $c \in S$, in the optimal solution for $\mathcal{I}$. Suppose, firstly, that c and p are contained in the same cell e. By the

construction of S', there must be some $c' \in S' \cap e$ (possibly $c = c'$) such that $c \prec c'$ and we can assign p to c'. Further, note that since p and c' belong to the same cell this is a valid "local" assignment and by Lemma 13, the cost of assignment equals $\Delta^e_{max} \leq \|c\|(1+\epsilon)$.

Subsequently, assume that p belongs to a cell e_1 and c belongs to a cell e_2, such that $e_1 \neq e_2$. We show that p can be assigned to one of the two representative points of e_2, namely $p^{e_2}_x$ or $p^{e_2}_y$. Recall that $p^{e_2}_x$ (resp. $p^{e_2}_y$) is a container point in e_2 with maximum x-coordinate (resp. y-coordinate). We first claim that there must exist a separating line $y = mx + C$ with slope $m \geq 0$, such that e_1 and e_2 lie on the opposite sides of this line (they could share a boundary along this line). We overload notation and allow $m = \infty$ in which the line is $x + C = 0$. So when $m = 0$ the line $(y = C)$ is parallel to the x-axis and when $m = \infty$ the line $(x = -C)$ is parallel to the y-axis.

Observe that by our construction, all the boundary lines have non-negative slopes. Therefore, if e_1 and e_2 share a boundary line segment, this will be our separating line. Suppose, on the other hand, that they do not share a boundary line segment and therefore are disjoint. If e_1 and e_2 are on the opposite sides of the line $y = x$, this will be our separating line. So, we assume that both the cells are on the same side of $y = x$, without loss of generality say above $y = x$. Then both these cells must be bounded by lines from the families $\mathcal{L}_2$ and $\mathcal{L}_3$. Let the lines bounding e_1 and e_2, respectively be, $B_1 = \{y = (1+\delta)^i,\ y = (1+\delta)^{i+1},\ x\sin\theta - y\cos\theta = 0,\ x\sin(\theta+\delta) - y\cos(\theta+\delta) = 0\}$ and $B_2 = \{y = (1+\delta)^j,\ y = (1+\delta)^{j+1},\ x\sin\theta' - y\cos\theta' = 0,\ x\sin(\theta'+\delta) - y\cos(\theta'+\delta) = 0\}$. Now, if $i = j$, then for the cells not to intersect, we must have $\theta \geq \theta' + \delta$ or $\theta' \geq \theta + \delta$. Without loss of generality, let $\theta \geq \theta' + \delta$. In this case, clearly the separating line is $x\sin\theta - y\cos\theta = 0$. In the case, where $i > j$ (resp. $i < j$), $y = (1+\delta)^j$ (resp. $y = (1+\delta)^i$) is a separating line.

We consider two different cases based on the value of m and prove that p can be assigned to some representative point in e_2.

Case 1: $m \in \{0, \infty\}$. The separating line between e_1 and e_2 is axis parallel, say $x = a$, without loss of generality. Since $p \prec c$, we have that the x-co-ordinates of all points in e_1 are less than a and x-coordinates of all points in e_2 are more than a. Hence, clearly the point with maximum y-coordinate in e_2, namely $p^{e_2}_y$ must dominate p.

Case 2: $m > 0$ and finite. Let the separating line be $y = mx + C$. There are two further cases here. First assume that p lies below the $y = mx + C$ and c lies above it. Letting $p = (x_1, y_1)$, $c = (x_2, y_2)$ and $p^{e_2}_x = (x_3, y_3)$, we have $y_1 \leq mx_1 + C$ and $y_2 \geq mx_2 + C$ and $y_3 \geq mx_3 + C$. By definition, $x_1 \leq x_2 \leq x_3$ and we focus on showing that $y_1 \leq y_3$. Indeed we have $y_1 \leq mx_1 + C \leq mx_2 + C \leq mx_3 + C \leq y_3$. Thus, $p \prec p^{e_2}_x$. Next, we assume that p lies above $y = mx + C$ and c lies below it. Letting $p = (x_1, y_1)$, $c = (x_2, y_2)$ and $p^{e_2}_y = (x_3, y_3)$, we have $y_1 \geq mx_1 + C$, $y_2 \leq mx_2 + C$ and $y_3 \leq mx_3 + C$. By definition, $y_1 \leq y_2 \leq y_3$. Further, $x_1 \leq y_1/m - C/m \leq y_2/m - C/m \leq y_3/m - C/m \leq x_3$. Hence, $p \prec p^{e_2}_y$. Therefore, we have shown that if p is assigned to c, we can assign it to a representative point, c_r, that lies in the same cell as c. From Lemma 13, this implies that our cost of assignment is $\|c_r\| \leq (1+\epsilon)\|c\|$. ◀

We now describe a dynamic program based poly-time algorithm to solve the *localized container selection problem*. This completes the proof of Theorem 10.

Algorithm for localized container selection. We define the dynamic program variable, $\mathcal{M}(e, k_e)$, for a given cell e, as the optimal cost of assigning all input points in e, to $k_e \leq k$ newly chosen container points in e, along with the set $\mathcal{R}$ of representative container points. We note that this variable can be computed in polynomial time using ideas in [15]. For completeness, we describe a simple algorithm to compute this variable for every e and $k_e \leq k$.

We recall that by the problem definition, all the container points in e are incomparable and have the same cost, C. Let $c_1(x_1, y_1), c_2(x_2, y_2), \ldots, c_l(x_l, y_l)$ be the ordering of the container points in e, in the descending order of the y_i. That is $y_1 \geq y_2 \geq \ldots \geq y_l$ and $x_1 \leq x_2 \leq \ldots \leq x_l$. For a given index $i \in [l]$ and integer $k_i \leq k_e$, we define the variable $\mathcal{N}(i, k_i, j)$ as the optimal cost of assigning every input point, (x, y), in e, such that $y > y_{i+1}$, by choosing k_i container points with index $\leq i$, with $j \leq i$ being the highest index container point chosen (that is c_j is chosen and none of $c_{j+1}, \ldots, c_i$ are chosen). The following recurrence computes the variable $\mathcal{N}(i, k_i, j)$. Let n_i be the number of input points contained by c_i, whose y-co-ordinates are $> y_{i+1}$. If c_i is chosen,

$$\mathcal{N}(i, k_i, i) = \operatorname*{Min}_{j<i} \mathcal{N}(i-1, k_i - 1, j) + n_i C$$

Now, if c_i is not chosen and c_j is the highest index container point chosen, with $j \leq i$, we assign the input points contained in c_i with x-coordinate $> x_j$ and y-coordinate $> y_{i+1}$ to the nearest representative container point (if no such point exists, then the cost of assignment is ∞). Further, we assign those, so far, unassigned input points with y-co-ordinate $> y_{i+1}$ and x-co-ordinate $\leq x_j$ to c_j. Let C_i denote the total cost of assignment of all these input points. We have

$$\mathcal{N}(i, k_i, j) = \mathcal{N}(i-1, k_i, j) + C_i$$

We can compute $\mathcal{M}$, using the following equation: $\mathcal{M}(e, k_e) = \operatorname*{Min}_{j \leq l} \mathcal{N}(l, k_e, j)$ Let there be μ cells in total. We order them arbitrarily as $e_1, e_2 \ldots e_\mu$. We define the variable $\mathcal{D}(i, k_i)$ as the total cost of assigning all the input points in the cells e_j, for $j \in [i]$, while choosing k_i new container points from these cells and using the representative set $\mathcal{R}$. The following simple recurrence defines the dynamic program.

$$\mathcal{D}(i, k_i) = \operatorname*{Min}_{\ell \leq k_i} \mathcal{D}(i-1, k_i - \ell) + \mathcal{M}(e_i, \ell)$$

The optimal solution has a cost $\mathcal{D}(\mu, k)$.

Remark. This approach does not extend directly even to dimension $d = 3$. There are issues in both main steps of the algorithm (1) we do not know a similar construction with $O(\log n)$ cells, and (2) the localized container selection problem also appears hard. In Section 3.2 we obtain an algorithm for the discrete container selection problem in $d > 2$ dimensions, using a linear programming relaxation and prove Theorem 11.

3.2 Discrete container selection in higher dimension

We now consider the discrete container selection problem in any dimension $d > 2$. Recall that $\mathcal{C}$ denotes the input points and $\mathscr{F}$ the potential container points. We prove Theorem 11. Our algorithm is based on the linear programming relaxation in the adjacent figure.

$$\begin{aligned}
\text{Min} \quad & \sum_{i \in \mathscr{F}} \|i\| \sum_{j \in \mathcal{C}} y_{ij} \\
s.t. \quad & y_{ij} \leq x_i, \qquad \forall i \in \mathscr{F},\, j \in \mathcal{C}, \\
& y_{ij} = 0, \qquad \forall j \not\prec i, \\
& \sum_{i \in \mathscr{F}} y_{ij} \geq 1, \qquad \forall j \in \mathcal{C}, \\
& \sum_{i \in \mathscr{F}} x_i \leq k, \\
& x, y \geq 0.
\end{aligned}$$

When the x and y variables are restricted to lie in $\{0, 1\}$ note that we obtain an exact formulation. This LP relaxation is similar to the one for (non-metric) facility location [13]. Indeed, our problem is a special case of non-metric k-median, for which the result of [13]

implies a $\big(1+\epsilon, O(\frac{1}{\epsilon}\log n)\big)$-bicriteria approximation algorithm. Our result (Theorem 11) is an improvement for fixed dimensions since $k \leq n$.

The first step in our algorithm is to solve the LP. Let (x, y) denote an optimal LP solution. The second step performs a filtering of the y variables, as in [13]. Let $C_j^* = \sum_{i \in \mathscr{F}} \|i\| \cdot y_{ij}$ denote the contribution of input point $j \in \mathcal{C}$ to the optimal LP objective. Define:

$$\overline{y}_{ij} = \begin{cases} (1+\frac{1}{\epsilon})y_{ij} & \text{if } \|i\| \leq (1+\epsilon)C_j^* \\ 0 & \text{otherwise.} \end{cases}$$

Also define $\overline{x}_i = (1+\frac{1}{\epsilon})x_i$ for all $i \in \mathscr{F}$, and $\overline{C}_j = (1+\epsilon)C_j^*$ for $j \in \mathcal{C}$.

▶ Claim 16. For each $j \in \mathcal{C}$, $\sum_{i \in \mathscr{F}} \overline{y}_{ij} \geq 1$. For each $j \in \mathcal{C}$ and $i \in \mathscr{F}$, $\overline{y}_{ij} \leq \overline{x}_i$.

Proof. Fix any $j \in \mathcal{C}$ and let $F_j = \{i \in \mathscr{F} : \|i\| > (1+\epsilon)C_j^*\}$. By Markov's inequality we have $\sum_{i \in F_j} y_{ij} < \frac{1}{1+\epsilon}$. So $\sum_{i \in \mathscr{F}} \overline{y}_{ij} = (1+\frac{1}{\epsilon}) \sum_{i \in \mathscr{F} \setminus F_j} y_{ij} \geq 1$. ◀

The third step of our algorithm formulates a geometric hitting-set problem with VC-dimension d. For each input point $j \in \mathcal{C}$, define a polytope $P_j \subseteq \mathbb{R}^d$ given by

$$P_j = \{v \in \mathbb{R}^d : j \prec v \text{ and } \|v\| \leq \overline{C}_j\} = \left\{ v \in \mathbb{R}^d : v_r \geq j_r \, \forall r \in [d], \sum_{r=1}^{d} v_r \leq \overline{C}_j \right\}.$$

Note that each P_j is described by $d+1$ *parallel inequalities* of the form:

$$\{-e_r^t v \leq -j_r\}_{r=1}^d \cup \{e^t v \leq \overline{C}_j\}.$$

Above e_r denotes the r^{th} coordinate unit vector and $e = (1, 1, \ldots, 1)$.

▶ Claim 17. For each $j \in \mathcal{C}$, $\sum_{i \in \mathscr{F} \cap P_j} \overline{x}_i \geq 1$.

Proof. This follows directly from Claim 16 since $\overline{y}_{ij} = 0$ for all $j \in \mathcal{C}$ and $i \notin P_j$. ◀

VC dimension bound. We use the following fact about the VC-dimension of a range space $(\mathscr{F}, \mathcal{P})$ where $\mathscr{F}$ is a finite set of points in $\mathbb{R}^d$ and $\mathcal{P}$ consists of all positive scaling and translations of a fixed polytope $Q \subseteq \mathbb{R}^d$ with $q \geq d$ facets.

▶ **Lemma 18.** *The VC-dimension of $(\mathscr{F}, \mathcal{P})$ is at most q.*

Proof. This may be a known result; in any case we give a short proof here. Let polytope $Q = \{x \in \mathbb{R}^d : \alpha_r^t x \leq \beta_r, \forall r \in [q]\}$ where each $\alpha_r \in \mathbb{R}^d$ and $\beta_r \in \mathbb{R}$.

The VC-dimension is the size of the largest subset $A \subseteq \mathscr{F}$ such that $\{A \cap P : P \in \mathcal{P}\} = 2^A$. Consider any such set A. Suppose (for contradiction) that $|A| > q$, then we will show a subset $A' \subseteq A$ such that there is no $P \in \mathcal{P}$ with $A \cap P = A'$. This would prove the claim.

For each constraint $r \in [q]$ let $a_r \in A$ denote a point that maximizes $\{\alpha_r^t x : x \in A\}$. Set $A' = \{a_r\}_{r=1}^q$. Note that there is some $a' \in A \setminus A'$ since $|A| > q$ and $|A'| \leq q$; moreover, by the choice of a_rs, we have $\alpha_r^t a' \leq \alpha_r^t a_r$ for all $r \in [q]$.

Suppose $P \in \mathcal{P}$ is any polytope that contains all points in A'. Note that $P = \{x \in \mathbb{R}^d : \alpha_r^t x \leq \gamma_r, \forall r \in [q]\}$ for some $\{\gamma_r \in \mathbb{R}\}_{r=1}^q$ since it is a scaled translation of the fixed polytope Q. Since $a_r \in P$ for each $r \in [q]$, we have $\gamma_r \geq \alpha_r^t a_r \geq \alpha_r^t a'$. This means that $a' \in P$ as well. Hence there is no set $P \in \mathcal{P}$ with $P \cap A = A'$. ◀

Applying Lemma 18 we obtain $(\mathscr{F}, \{P_j : j \in \mathcal{C}\})$ has VC-dimension at most $d+1$. Moreover, by Claim 17 the hitting set instance $(\mathscr{F}, \{P_j : j \in \mathcal{C}\})$ has a fractional hitting set $\{\overline{x}_i : i \in \mathscr{F}\}$ of size $(1+\frac{1}{\epsilon})k$. Thus we can use the following well-known result:

▶ **Theorem 19** ([9, 6]). *Given any hitting set instance on a set-system with VC-dimension d and a fractional hitting set of size k, there is a polynomial time algorithm to compute an integral hitting set of size $O(d \log(dk)) \cdot k$.*

This completes the proof of Theorem 11.

Remark: We can also use this LP-based approach to obtain a constant-factor bicriteria approximation for the discrete container selection problem in $\mathbb{R}^2$. This is based on the ϵ-net result for "pseudo-disks" in $\mathbb{R}^2$ [14] and the observation that in dimension two the above set-system $(\mathscr{F}, \{P_j : j \in \mathcal{C}\})$ is a collection of pseudo-disks. However, the constant factor obtained via this approach is much worse than the direct approach in Section 3.1.

4 Hardness Results

In this section, we provide hardness results for the continuous and discrete container selection problems in dimension $d = 3$. All hardness results discussed here are strongly NP-hard. The reductions are based on the planar degree 3 vertex cover problem. The following restriction of this problem is also known to be NP-hard [8].

▶ **Definition 20** (Plane Degree 3 Vertex Cover (PVC)). The input is a bound k and a plane drawing of a degree 3 planar graph $G = (V, E)$ with girth at least 4, where the Euclidean distance between any pair $u, v \in V$ of vertices is exactly one if $(u, v) \in E$ and at least $\sqrt{3}$ if $(u, v) \notin E$. The decision problem is to determine whether G has a vertex cover of size at most k.

We first show that the following auxiliary problem is NP-hard.

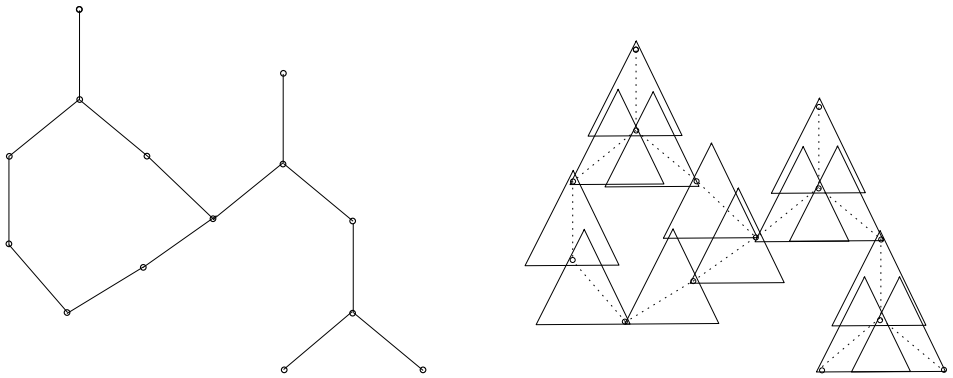
▶ **Definition 21** (Δ-hitting problem). The input is a bound k, a set V of points in the plane where each pairwise distance is at least one and a set $\{\Delta_e\}_{e \in E}$ of (possibly intersecting) equilateral triangles with side $s := \frac{2}{\sqrt{3}}$ that are all translates of each other. The goal is to find a subset $T \subseteq V$ with $|T| \leq k$ such that $T \cap \Delta_e \neq \emptyset$ for all $e \in E$.

▶ **Theorem 22.** *The Δ-hitting problem is NP-hard.*

Proof. We reduce the NP-hard PVC problem to the Δ-hitting problem (refer to Figure 3). An instance of PVC consists of a plane drawing of graph $G = (V, E)$ and bound k. We construct an instance of the Δ-hitting problem as follows. The set of points is V and the bound is k. Note that the the distance between each pair of points is at least one, by Definition 20. For each edge $e = (u, v) \in E$ we can find (in polynomial time) an equilateral triangle Δ_e with side $s = \frac{2}{\sqrt{3}}$ such that $V \cap \Delta_e = \{u, v\}$. To see this, first note that we can easily find $\Delta_e \ni u, v$ as $d(u, v) = 1$. Since the diameter of Δ_e is $\frac{2}{\sqrt{3}} < \sqrt{3}$ the vertices $V \cap \Delta_e$ form a clique in G, and as G has girth 4 we must have $|V \cap \Delta_e| = 2$. The set of triangles in the Δ-hitting problem is $\{\Delta_e\}_{e \in E}$. Moreover, we can ensure that the triangles $\{\Delta_e\}_{e \in E}$ are all translates of some canonical triangle. It is now clear that the Δ-hitting problem is a yes-instance if and only if the PVC instance has a vertex cover of size at most k. ◀

▶ **Theorem 23.** *The 3-dimensional discrete container selection problem is NP-hard.*

Proof. We reduce the Δ-hitting problem to this problem (refer to Figure 4). Consider an instance as described in Definition 21. We construct an instance of the discrete problem in $\mathbb{R}^3$ as follows. Set $A = 2|V|$ and let Π denote the plane $x + y + z = A$. We place the points V and triangles $\{\Delta_e\}_{e \in E}$ of the Δ-hitting instance on plane Π oriented so that every triangle Δ_e is



(a) Plane degree 3 vertex cover instance **(b)** Δ-hitting set instance

Figure 3 Reduction of a PVC instance to a Δ-hitting set instance: For every edge in Figure 3a, we construct an equilateral triangle, in Figure 3b, that contains the incident vertices of the edge and no other vertex. All such triangles are translates of each other. A vertex cover in the former instance is a Δ-hitting set in the latter and vice versa.

parallel[1] to the triangle $\{(A,0,0), (0,A,0), (0,0,A)\}$. We can ensure that all points in V are in the positive orthant since A is large. The potential container points are V. Observe that for each triangle Δ_e there is a unique point $p_e \in \mathbb{R}^3$ such that $\Delta_e = \Pi \cap \{x \in \mathbb{R}^3 : p_e \prec x\}$. The set of input points is $\{p_e\}_{e \in E}$. The bound k is same as for the Δ-hitting problem.

It is easy to see that the discrete container selection instance has a feasible solution with k containers if and only if the Δ-hitting instance is a yes-instance. ◀

We immediately have the following corollary of the Theorem 23, which stems from the fact that it is NP-hard to even test feasibility of the discrete container selection problem.

▶ **Corollary 24.** *It is NP-hard to approximate the 3-dimensional discrete container selection problem within any approximation guarantee.*

▶ **Theorem 25.** *The 3-dimensional continuous container selection problem is NP-hard.*

Proof. We reduce a special variant of the discrete container selection problem whose instances are defined as in Theorem 23. Let $\mathcal{I}_1 = (\mathcal{C}, \mathscr{F}, k)$ denote an instance of the discrete container selection problem from Theorem 23 where $\mathcal{C}$ are the input points and $\mathscr{F}$ denotes the potential container points. Note that all points of $\mathscr{F}$ lie on the plane $x + y + z = A$, and the distance between every pair of points in $\mathscr{F}$ is at least one. Observe that the latter property implies that the points in $\mathscr{F}$ are incomparable.

We construct an instance $\mathcal{I}_2 = (\mathcal{C}', k')$, of the continuous problem in the following way. Fix parameter $\delta < \frac{1}{2}$. For every point $c \in \mathscr{F}$ we define another point $\hat{c} := c + \delta(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$; note that $\|\hat{c}\| = \|c\| + \delta$ and $\hat{c}$ dominates c but no other point in $\mathscr{F} \setminus \{c\}$. Let $\hat{\mathscr{F}} = \{\hat{c} : c \in \mathscr{F}\}$. Observe that this is well-defined: since the distance between every pair of points in $\mathscr{F}$ is at least one, any point dominating more than one point of $\mathscr{F}$ costs at least $A + 1$.

Now, the set $\mathcal{C}'$ of input points is constructed as follows. Let $M_1 \gg |\mathcal{C}|A$ and $M_2 \gg 2(|\mathcal{C}|A + |\mathscr{F}|M_1)$ be two sufficiently large integers. For each $c \in \mathscr{F}$, we create M_1 input

[1] Two triangles Δ_1 and Δ_2 are parallel if and only if their corresponding sides are mutually parallel

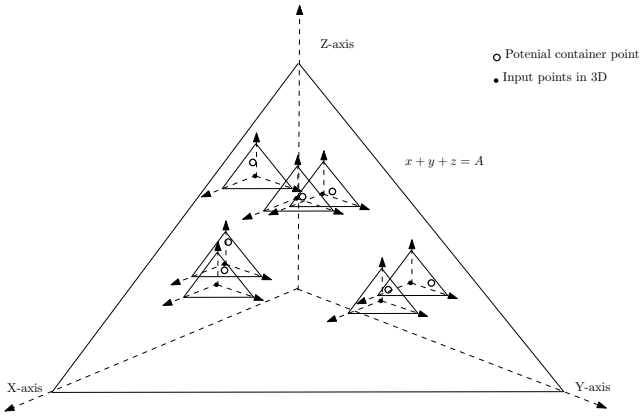


Figure 4 Reduction of the Δ-hitting set problem to the discrete container selection problem (in 3D). We consider special instances of the latter where all the potential container points are assumed to be on the plane $\Pi \equiv x+y+z=A$ and all input points lie below Π. Notice that the projections of input points onto Π form equilateral triangles as shown. Any feasible solution for the container selection problem is a Δ-hitting set in the resulting instance and vice versa.

points at c and M_2 input points at $\hat{c}$, which are added to $\mathcal{C}'$. Finally we also add the points $\mathcal{C}$ to $\mathcal{C}'$. The bound $k' := k + |\mathscr{F}|$. We claim that $\mathcal{I}_1$ is feasible if and only if $\mathcal{I}_2$ has a solution of cost at most $T := |\mathcal{C}|A + |\mathscr{F}|(M_1+M_2)(A+\delta) - kM_1\delta$.

Forward direction. Let $S = \{c_1, c_2, \ldots, c_k\}$ be the set of container points chosen by a feasible solution of $\mathcal{I}_1$. Consider the set $S' = S \cup \hat{\mathscr{F}}$. Observe that S' is a feasible solution for the instance $\mathcal{I}_2$. We now compute the assignment cost of this solution.

- The assignment cost for each point in $\mathscr{C}$ is A (it is covered by S).
- The input points at locations of S have assignment cost A (there are kM_1 such points).
- The remaining $(|\mathscr{F}| - k)M_1 + |\mathscr{F}|M_2$ input points have assignment cost $A+\delta$ each.

Therefore the total cost of this solution is exactly T.

Backward direction. Let S' with $|S'| = k + |\mathscr{F}|$ be a feasible solution to $\mathcal{I}_2$ of cost at most T. We first argue that $\hat{\mathscr{F}} \subseteq S'$. Indeed, assume that it is not true. Observe that, in this case, the input points at $\hat{\mathscr{F}}$ should be dominated by $< |\mathscr{F}|$ container points. So some container point $s \in S'$ should dominate input points at two distinct locations $\hat{c_i}$ and $\hat{c_j}$. Note that $|s| \geq A+1$ since $c_i, c_j \prec s$ (using the distance one separation between points of $\mathscr{F}$). Hence any such solution has assignment cost at least $AM_1|\mathscr{F}| + (A+\delta)M_2|\mathscr{F}| + (1-\delta)M_2 > T$ using the definition of M_2. We now assume $\hat{\mathscr{F}} \subseteq S'$. Next we show that each of the remaining k container points in S' dominates at most one point of $\mathscr{F}$. If $s \in S'$ dominates two distinct locations c_i and c_j, its cost $|s| \geq A+1$ as noted above. However, any input point can be assigned to one of the container points in $\hat{\mathscr{F}}$ at cost $A+\delta < A+1$, which makes point s redundant.

Now we show that each of the k container points $S' \setminus \hat{\mathscr{F}}$ dominates some point of $\mathscr{F}$. If not, consider a container point $s' \in S'$ that does not dominate any $\mathscr{F}$ point. Let $f \in \mathscr{F}$ be some point which is not dominated by any $S' \setminus \hat{\mathscr{F}}$; note that this must exist since each $S' \setminus \hat{\mathscr{F}}$ dominates at most one $\mathscr{F}$-point and $|S' \setminus \hat{\mathscr{F}}| = k \leq |\mathscr{F}|$. Suppose we modify the solution by removing s' and adding f: the increase in cost is at most $|\mathcal{C}|(A+\delta) + M_1 A - M_1(A+\delta) < 0$ by the definition of M_1. Thus, $\hat{\mathscr{F}} \subseteq S'$ and $S'' = S' \setminus \hat{\mathscr{F}} \subseteq \mathscr{F}$. We now claim that S'' dominates every point of $\mathcal{C}$. For a contradiction, suppose there is some point of $\mathcal{C}$ that is not dominated by S'' : then this point has assignment cost $A+\delta$. Every other points of $\mathcal{C}$ has assignment

cost at least A. The assignment cost of points at $\hat{\mathscr{F}} \cup \mathscr{F}$ is $|\mathscr{F}|(M_1 + M_2)(A + \delta) - kM_1\delta$. So the total assignment cost is at least $T + \delta$, a contradiction. Hence S'' is a feasible solution for $\mathcal{I}_1$. ◀

References

1 Amazon EC2. In `http://aws.amazon.com/ec2/`.
2 Private cloud. In `http://wikipedia.org/wiki/Cloud_computing#Private_cloud`.
3 Marcel R Ackermann, Johannes Blömer, and Christian Sohler. Clustering for metric and nonmetric distance measures. *ACM Transactions on Algorithms (TALG)*, 6(4):59, 2010.
4 Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. In *IPCO*, pages 52–63, 2014.
5 Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *STOC*, pages 106–113, 1998.
6 Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.
7 Jaroslaw Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An Improved Approximation for k-median, and Positive Correlation in Budgeted Optimization. In *SODA*, 2015.
8 Tomás Feder and Daniel H. Greene. Optimal algorithms for approximate clustering. In *STOC*, pages 434–444, 1988.
9 David Haussler and Emo Welzl. ϵ-Nets and Simplex Range Queries. *Discrete &Computational Geometry*, 2:127–151, 1987.
10 Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony Joseph, Scott Shenker, and Ion Stoica. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In *NSDI*, 2011.
11 A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3), September 1999.
12 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *STOC*, pages 901–910, 2013.
13 Jyh-Han Lin and Jeffrey Scott Vitter. epsilon-approximations with minimum packing constraint violation (extended abstract). In *STOC*, pages 771–782, 1992.
14 Evangelia Pyrga and Saurabh Ray. New existence proofs epsilon-nets. In *SOCG*, pages 199–207, 2008.
15 Baruch Schieber. Computing a Minimum Weight k-Link Path in Graphs with the Concave Monge Property. *Journal of Algorithms*, 29(2):204–222, 1998.
16 V. Vavilapalli, A. Murthy, C. Douglis, A. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwiele. Apache Hadoop YARN: Yet Another Resource Negotiator. In *SoCC*, 2013.
17 Joel Wolf, Zubair Nabi, Viswanath Nagarajan, Robert Saccone, Rohit Wagle, Kirsten Hildrum, Edward Ping, and Kanthi Sarpatwar. The X-Flex Cross-Platform Scheduler: Who's The Fairest Of Them All?. In *Middleware*, 2014.