Data Driven Resource Allocation for Distributed Learning

Travis Dick
Carnegie Mellon University

Mu Li
Carnegie Mellon University

Venkata Krishna Pillutla
University of Washington

Colin White
Carnegie Mellon University

Maria Florina Balcan
Carnegie Mellon University

Alex Smola
Carnegie Mellon University
and AWS Deep Learning

Abstract

In distributed machine learning, data is dispatched to multiple machines for processing. Motivated by the fact that similar data points often belong to the same or similar classes, and more generally, classification rules of high accuracy tend to be "locally simple but globally complex" (Vapnik and Bottou, 1993), we propose data dependent dispatching that takes advantage of such structure. We present an in-depth analysis of this model, providing new algorithms with provable worst-case guarantees, analysis proving existing scalable heuristics perform well in natural non worst-case conditions, and techniques for extending a dispatching rule from a small sample to the entire distribution. We overcome novel technical challenges to satisfy important conditions for accurate distributed learning, including fault tolerance and balancedness. We empirically compare our approach with baselines based on random partitioning, balanced partition trees, and locality sensitive hashing, showing that we achieve significantly higher accuracy on both synthetic and real world image and advertising datasets. We also demonstrate that our technique strongly scales with the available computing power.

1 INTRODUCTION

Motivation and Overview: Distributed computation is playing a major role in modern large-scale machine learning practice with a lot of work in this direction in the last few years (Balcan et al., 2012b, 2013b, 2014; Li et al., 2014; Zhang et al., 2013, 2012). This tends to take two high-level forms. The first is when the data itself is collected in a distributed manner, whether from geographically-distributed experiments, distributed sensors, distributed click data, etc., and the goal is to take advantage of all this data without incurring the substantial overhead of first communicating it all to some central location. The second high-level form is where massive amounts of data are collected centrally, and for space and efficiency reasons this data must be dispatched to distributed machines in order to perform the processing needed (Li et al., 2014; Zhang et al., 2012). It is this latter form that we address here.

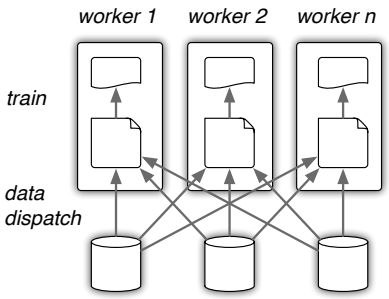


Figure 1: Data is partitioned and dispatched into multiple workers. Each worker then trains a local model using its local data. There is no communication between workers during training.

When data is dispatched to distributed machines, the simplest approach and what past work (both theoretical and empirical) has focused on is to perform the dispatching randomly (Zhang et al., 2012, 2013). Random dispatching has the advantage that dispatching is easy, and because each machine receives data from the same distribution, it is rather clean to analyze theoretically. However, since the distributions of the data on each machine are identical statistically, such techniques could lead to sub-optimal results in practice in terms of the accuracy of the resulting learning rule. Motivated by the fact that in practice, similar data points tend to have the same or similar classification, and more generally, classification rules of high accuracy tend to be "locally simple but globally complex" (Vapnik and Bottou, 1993), we propose a new paradigm for performing *data-dependent dispatching* that takes advantage of such structure by sending similar datapoints to similar machines. For example, a

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

globally accurate classification rule may be complicated, but each machine can accurately classify its *local* region with a simple classifier.

We introduce and analyze dispatching techniques that partition a set of points such that similar examples end up on the same machine/worker, while satisfying key constraints present in a real world distributed system including balancedness and fault-tolerance. Such techniques can then be used within a simple, but highly efficient distributed system that first partitions a small initial segment of data into a number of sets equal to the number of machines. Then each machine locally and independently applies a learning algorithm, with no communication between workers at training. In other words, the learning is embarrassingly parallel. See Figure 1 for a schematic representation. At the prediction time, we use a super-fast sublinear algorithm for directing new data points to the most appropriate machine.

Our Contributions: We propose a novel scheme for partitioning data which leads to better accuracy in distributed machine learning tasks, and we give a theoretical and experimental analysis of this approach. We present new algorithms with provable worst-case guarantees, analysis proving existing scalable heuristics perform well in natural non worst-case conditions, techniques for extending a dispatching rule from a small sample to the entire distribution, and an experimental evaluation of our proposed algorithms and several baselines on both synthetic and real-world image and advertising data. We empirically show that our method strongly scales and that we achieve significantly higher accuracy over baselines based on random partitioning, balanced partition trees, and locality-sensitive hashing.

In our framework, a central machine starts by clustering a small sample of data into roughly equal-sized clusters, where the number of clusters is equal to the number of available machines. Next, we extend this clustering into an efficient dispatch rule that can be applied to new points. This dispatch rule is used to send the remaining training data to the appropriate machines and to direct new points at prediction time. In this way, similar datapoints wind up on the same machine. Finally, each machine independently learns a classifier using its own data (in an embarrassingly parallel manner). To perform the initial clustering used for dispatch, we use classic clustering objectives (k-means, k-median, and k-center). However, we need to add novel constraints to ensure that the clusters give a data partition that respects the constraints of real distributed learning systems:

Balancedness: We need to ensure our dispatching procedure balances the data across the different machines. If a machine receives much more data than other machines, then it will be the bottleneck of the algorithm. If any machine receives very little data, then its processing power is wasted. Thus, enforcing upper and lower bound constraints on the cluster sizes leads to a faster, more efficient setup.

Fault-Tolerance: In order to ensure that our system is robust to machine failures, we assign each point to multiple distinct clusters. This way, even if a machine fails, the data on that machine is still present on other machines. Moreover, this has the added benefit that our algorithms behave well on points near the boundaries of the clusters. We say a clustering algorithm satisfies p-replication if each point is assigned to p distinct clusters.

Efficiency: To improve efficiency, we apply our clustering algorithms to a small sample of data. Therefore, we need to be able to extend the clustering to new examples from the same distribution while maintaining a good objective value and satisfying all constraints. It is important that the extension technique be efficient for both the initial partitioning and when we dispatch examples at prediction time.

When designing clustering algorithms, adding balancedness and fault tolerance makes the task significantly harder. Prior work has considered upper bounds on the cluster sizes (Li, 2014b; Byrka et al., 2015b; Li, 2014a; An et al., 2014; Khuller and Sussmann, 1996; Cygan et al., 2012) [1] and lower bounds (Aggarwal et al., 2006; Ahmadian and Swamy, 2016), but no prior work has shown provable guarantees with upper and lower bounds on the cluster sizes simultaneously. While capacitated clustering objective functions are nondecreasing as the number of clusters k increases, with lower bounds on the cluster sizes, we show the objective function can oscillate arbitrarily with respect to k. This makes the problem especially challenging from a combinatorial optimization perspective. Existing capacitated clustering algorithms work by rounding a fractional linear program solution, but the erratic nature of the objective function makes this task more difficult for us.

The balance constraints also introduce challenges when extending a clustering-based partitioning from a small sample to unseen data. The simple rule that assigns a new point to the cluster with the nearest center provides the best objective value on new data, but it can severely violate the balance constraints. Therefore, any balanced extension rule must take into account the distribution of data.

We overcome these challenges, presenting a variety of complementary results, which together provide strong justification for our distributed learning framework. We summarize each of our main results below.

• **Balanced fault-tolerant clustering:** We provide the first algorithmic results with provable guarantees that simultaneously handle upper and lower bounds on the cluster sizes, as well as fault tolerance. Clustering is NP-hard and adding more constraints makes it significantly harder, as we will see in Section 2. For this reason, we first devise approximation

[1] Note that enforcing only upper (resp. lower) bounds implies a weak lower (resp. upper) bound on the cluster sizes, but this is only nontrivial if the upper (resp. lower) bounds are extremely tight or the number of clusters is a small constant.

algorithms with strong worst-case guarantees, demonstrating this problem is tractable. Specifically, in Section 2 we provide an algorithm that produces a fault-tolerant clustering that approximately optimizes k-means, k-median, and k-center objectives while also roughly satisfying the given upper and lower bound constraints. At a high level, our algorithm proceeds by first solving a linear program, followed by a careful balance and replication aware rounding scheme. We use a novel min-cost flow technique to finish off rounding the LP solution into a valid clustering solution.

• **k-means++ under stability:** In addition to these algorithms which give provably strong guarantees in the worst-case, we give complementary results which show that for 'typical' problem instances, it is possible to achieve better guarantees with simpler, more scalable algorithms. Specifically, in Section 3 we show the popular k-means++ algorithm outputs a balanced clustering with stronger theoretical guarantees, provided the data satisfies a natural notion of stability. We make nontrivial extensions of previous work to ensure the upper and lower size constraints on the clusters are satisfied. No previous work gives provable guarantees while satisfying both upper and lower bounds on the cluster sizes, and Sections 2 and 3 may be of independent interest beyond distributed learning.

• **Efficient clustering by subsampling:** For datasets large enough to require distributed processing, clustering the entire dataset is prohibitively expensive. A natural way to avoid this cost is to only cluster a small subset of the data and then efficiently extend this clustering to the entire dataset. The simple extension that assigns each new point to the p clusters with the closest centers does not satisfy the balance constraints. Instead, in Section 4 we show that assigning a new example to the same p clusters as its nearest neighbor in the clustered subsample approximately preserves both the objective value and all constraints. We also use this technique at prediction time to send new examples to the most appropriate machines.

• **Experimental results:** We conduct experiments with both our LP rounding algorithms and k-means++ together with our nearest neighbor extension technique. We include empirical (and theoretical) comparisons which show the effectiveness of both algorithms in different situations. The k-means++ algorithm is competitive on real world image and advertising datasets, complementing the results of Section 3 by showing empirically that k-means++ produces high-quality balanced clusterings for 'typical' datasets. We then compare the performance of our framework (using k-means++ with nearest neighbor extension) against three baseline methods (random partitioning, balanced partition trees, and locality sensitive hashing) in large scale learning experiments where each machine trains an SVM classifier. We find that for all datasets and across a wide range of k values, our algorithm achieves higher accuracy than any of the baselines. Finally, we show that our technique strongly scales, meaning that doubling the available computational power while keeping the workload fixed reduces the running time by a constant factor, demonstrating that our method can scale to very large datasets.

Related Work: Currently, the most popular method of dispatch in distributed learning is random dispatch (Zhang et al., 2013, 2012). This may not produce optimal results because each machine must learn a global model. Previous work has studied partitioning for distributed machine learning (Wei et al., 2015; You et al., 2015; Delling et al., 2011; Bourse et al., 2014; Aydin et al., 2016), but none simultaneously achieve load-balancing guarantees and approximation guarantees for k-median, k-means, or k-center.

Previous work in theoretical computer science has considered capacitated clustering, or clustering with upper bounds (Li, 2014b; Byrka et al., 2015b; Li, 2014a; Cygan et al., 2012), and lower bounds (Aggarwal et al., 2006; Ahmadian and Swamy, 2016), but our algorithm is the first to solve a more general and challenging question of simultaneously handling upper and lower bounds on the cluster sizes, and p-replication. See Section 7 in the supplementary material for a more detailed discussion about related work.

2 FAULT TOLERANT BALANCED CLUSTERING

In this section, we give an algorithm to cluster a small initial sample of data to create a dispatch rule that sends similar points to the same machine. There are many ways to measure the similarity of points in the same cluster. We consider three classic clustering objectives, k-means, k-median, and k-center clustering while imposing upper and lower bounds on the cluster sizes and replication constraints. This is the first algorithm with provable guarantees to simultaneously handle both upper and lower bounds on the cluster sizes.

A clustering instance consists of a set V of n points, and a distance metric d. Given two points i and j in V, denote the distance between i and j by $d(i,j)$. The task is to find a set of k centers $C = \{c_1, \ldots, c_k\} \subset V$ and assignments of each point to p of the centers $f : V \to \binom{C}{p}$, where $\binom{C}{p}$ represents the subset of C^p with no duplicates. In this paper, we study three popular clustering objectives:

(1) k-median: $\min_{C,f} \sum_{i \in V} \sum_{j \in f(i)} d(i,j)$

(2) k-means: $\min_{C,f} \sum_{i \in V} \sum_{j \in f(i)} d(i,j)^2$

(3) k-center: $\min_{C,f} \max_{i \in V} \max_{j \in f(i)} d(i,j)$

We add size constraints $0 < \ell \leq L < 1$, also known as capacity constraints, so each cluster must have a size between $n\ell$ and nL. For simplicity, we assume these values are integral (or replace them by $\lceil n\ell \rceil$ and $\lfloor nL \rfloor$ respectively). Before we present our approximation algorithm, we discuss the challenges introduced by these size constraints.

Structure of Balanced Clustering: It is well-known that

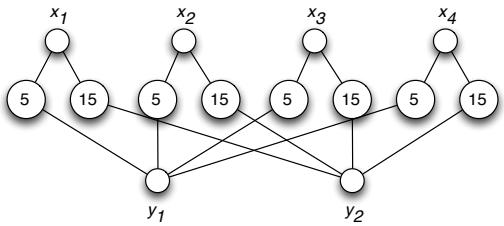


Figure 2: Each edge signifies distance 1, and all other distances are 2. The middle points are replicated as many times as their label suggests (but each pair of replicated points are still distance 2 away). Finally, add length 1 edges between all pairs in $\{x_1, x_2, x_3, x_4\}, \{y_1, y_2\}$.

solving the objectives optimally are NP-hard (even without the capacity and fault tolerance generalizations) (Jain et al., 2003). In fact, with the addition of lower bounds, the value of the optimal clustering objective $\mathcal{OPT}$ as a function of k behaves erratically. In uncapacitated clustering and clustering with upper bounds only, given a problem instance, the cost of the optimal solution always decreases as k increases. This is easy to see: given a set of optimal centers, if we add another center v, at the very least v is now distance 0 from a center, which decreases the cost.

However, when there are lower bounds on the cluster sizes, there are simple examples in which the value of the optimal solution as a function of k contains a local minimum. For instance, the star graph has this property (see Section 11 in the supplementary material). A much more subtle question is whether there exists a clustering instance with a local *maximum*. We confirm such clusterings do exist; see Figure 2. We give the idea here and defer the formal proof to Section 8 in the supplementary material.

Lemma 1. *There exists a balanced clustering instance with $p = 1$ for which the k-center, k-median, and k-means objectives contain a local maximum with respect to k.*

Proof sketch. Consider Figure 2, where $n = 86$, and set $n\ell = 21$. Since the distances are all 1 or 2, this construction is trivially a valid distance metric. From Figure 2, we see that $k = 2$ and $k = 4$ have valid clusterings using only length 1 edges, using centers $\{y_1, y_2\}$ and $\{x_1, x_2, x_3, x_4\}$, respectively. But now consider $k = 3$. The crucial property is that by construction, y_1 and any x_i cannot simultaneously be centers and each satisfy the capacity to distance 1 points, because the union of their distance 1 neighborhoods is less than $2n\ell$. In the supplementary material, we carefully check all other sets of 3 centers do not achieve a clustering with distance 1 edges, which completes the proof. □

In fact, with a more intricate clustering instance, we are able to show (in Theorem 8 in the supplementary material) that *any number* of local maxima may exist!

Approximation Algorithm: In light of these difficulties, one might ask whether any approximation algorithm exists for this problem. We answer affirmatively, by extending previous work (Li, 2014a) to fit our more challenging constrained optimization problem. Our algorithm returns a clustering whose cost is at most a constant factor multiple of the optimal solution, while violating the capacity and replication constraints by a small constant factor.

Theorem 2. *Algorithm 1 returns a constant factor approximate solution for the balanced k-clustering with p-replication problem for $p > 1$, where the upper capacity constraints are violated by at most a factor of $\frac{p+2}{p}$, and each point can be assigned to each center at most twice.*

At a high level, our algorithm proceeds by first solving a linear program, followed by careful rounding. The key insight is that p-replication helps to mitigate the capacity violation in the rounding phase. Together with a novel min-cost flow technique, this allows us to simultaneously handle upper and lower bounds on the cluster sizes. The procedure is summarized in Algorithm 1, and below we provide details, together with the key ideas behind its correctness (see Section 9 in the supplementary material for the full details).

Step 1: Linear Program The first step is to solve an linear program (LP) relaxation of the integer program (IP) formulation of our constrained clustering problem. The variables are as follows: for each $i \in V$, let y_i be an indicator for whether i is opened as a center. For $i, j \in V$, let x_{ij} be an indicator for whether point j is assigned to center i. In the LP, the variables may be fractional, so y_i represents the fraction to which a center is opened (we will refer to this as the "opening" of i), and x_{ij} represents the fractional assignment of j to i. One can use an LP solver to get a fractional solution which must then be rounded (i.e., the LP may open up $2k$ 'half' centers). Let (x, y) denote an optimal solution to the LP. For any points i and j, let c_{ij} be the cost of assigning point j to center i. That is, for k-median, $c_{ij} = d(i, j)$, and for k-means $c_{ij} = d(i, j)^2$ (we discuss k-center in the supplementary material). Define $C_j = \sum_i c_{ij} x_{ij}$, the average cost from point j to its centers in the LP solution (x, y).

It is well-known that the LP in Algorithm 1 has an unbounded integrality gap (the ratio of the optimal LP solution over the optimal integral LP solution), even when the capacities are violated by a factor of $2 - \epsilon$ (Li, 2014a). However, with fault tolerance, the integrality is only unbounded when the capacities are violated by a factor of $\frac{p}{p-1}$ (see the supplementary material for the integrality gap). Intuitively, this is because the p centers can 'share' this violation.

Step 2: Monarch Procedure Next, partition the points into "empires" such that every point is $\leq 4C_j$ from the center of its empire (the "monarch") by using a greedy procedure from Charikar et al. (1999) (for an informal description, see step 2 of Algorithm 1). By Markov's inequality, every empire has total opening $\geq p/2$, which is crucially ≥ 1 for $p \geq 2$ under our model of fault tolerance.

1. Find a solution to the following linear program:

$$\min_{x,y} \sum_{i,j \in V} c_{ij} x_{ij} \quad \text{s.t.}$$

$$\textbf{(a)}\ \forall j \in V : \sum_{i \in V} x_{ij} = p; \quad \textbf{(b)}\ \sum_{i \in V} y_i \leq k;$$

$$\textbf{(c)}\ \forall i \in V : \ell y_i \leq \sum_{j \in V} \frac{x_{ij}}{n} \leq L y_i;$$

$$\textbf{(d)}\ \forall i, j \in V : 0 \leq x_{ij} \leq y_i \leq 1.$$

2. Greedily place points into a set $\mathcal{M}$ from lowest C_j to highest (called "monarchs"), adding point j to $\mathcal{M}$ if it is not within distance $4C_j$ of any monarch. For each monarch u, let $\mathcal{E}_u$ be the points closest to u, called u's empire.
3. For empire $\mathcal{E}_u$ with total fractional opening $Y_u \triangleq \sum_{i \in \mathcal{E}_u} y_i$, give opening $Y_u / \lfloor Y_u \rfloor$ to the $\lfloor Y_u \rfloor$ closest points to u and all other points opening 0.
4. Round the x_{ij}'s by constructing a minimum cost flow problem on a bipartite graph of centers and points, setting up demands and capacities to handle the bounds on cluster sizes.

Algorithm 1: Balanced clustering with fault tolerance

Step 3: Aggregation The point of this step is to end up with $\leq k$ centers total. Since each empire has total opening at least 1, we can aggregate openings within each empire. For each empire $\mathcal{E}_u$, we move the openings to the $\lfloor Y_u \rfloor$ innermost points of $\mathcal{E}_u$, where $Y_u = \sum_{i \in \mathcal{E}_u} y_i$. We accomplish this using an iterative greedy procedure, similar to (Li, 2014a) (we give details in the supplementary material). We preserve all LP constraints, except we may incur a factor $\frac{p+2}{p}$ increase to the capacity constraints. At the end of the procedure, there are $\leq k$ points with nonzero opening, so we can set them all to 1 to round the y's. The cost incurred in this step can be bounded using the triangle inequality.

Step 4: Min cost flow Now we must round the x's. We set up a min cost flow problem, where an integral solution corresponds to an assignment of points to centers. We create a bipartite graph with V on the left (each with supply p) and the k centers on the right (each with demand $n\ell$), and a sink vertex with demand $np - kn\ell$. We carefully set the edge weights so that the minimum cost flow that satisfies the capacities corresponds to an optimal clustering assignment. Then using the Integral Flow Theorem, we are guaranteed there is an *integral* assignment that achieves the same optimal cost (and finding the min cost flow is a well-studied polynomial time problem (Papadimitriou and Steiglitz, 1998)). Thus, we can round the x's without incurring any additional cost to the approximation factor. This is the first time this technique has been used in the setting of clustering.

In Section 10 of the supplementary material, we show a more involved algorithm specifically for k-center which achieves a 6-approximation with *no violation* to the capacity or replication constraints.

3 BALANCED CLUSTERING UNDER STABILITY

In the previous section, we showed an LP-based algorithm which provides theoretical guarantees even on adversarially chosen data. Often real-world data has inherent structure that allows us to use more scalable algorithms and achieve even better clusters (Balcan et al., 2013a; Ostrovsky et al., 2006). In our distributed ML framework, this translates to being able to use a larger initial sample for the same computational power (Section 4 analyzes the effect of sample size). In this section, we prove the popular k-means++ algorithm as well as a greedy thresholding algorithm output clusters very close to the optimal solution, provided the data satisfies a natural notion of stability called *approximation stability* (Balcan et al., 2013a; Agarwal et al., 2015; Balcan and Braverman, 2010; Balcan et al., 2016; Gupta et al., 2014).

Specifically, we show that given a balanced clustering instance in which clusterings close in *value* to $\mathcal{OPT}$ are also close in terms of the clusters themselves, assuming $L \in O(\ell)$, then k-means++ with a simple pruning step (Ostrovsky et al., 2006) outputs a solution close to optimal. We overcome key challenges that arise when we add upper and lower bounds to the cluster sizes. We include the details in Section 11 of the supplementary material.

Approximation Stability: Given a clustering instance (S, d) and inputs ℓ and L, and let $\mathcal{OPT}$ denote the cost of the optimal balanced clustering. Two clusterings $\mathcal{C}$ and $\mathcal{C}'$ are ϵ-*close*, if only an ϵ-fraction of the input points are clustered differently in the two clusterings, i.e., $\min_\sigma \sum_{i=1}^k |C_i \setminus C'_{\sigma(i)}| \leq \epsilon n$, where σ is a permutation of $[k]$.

Definition 1 (Balcan et al. (2013a))**.** *A clustering instance (S, d) satisfies $(1 + \alpha, \epsilon)$*-approximation stability *with respect to balanced clustering if all clusterings $\mathcal{C}$ with cost($\mathcal{C}$) $\leq (1 + \alpha) \cdot \mathcal{OPT}$ are ϵ-close to $\mathcal{C}$.*

k-means: We show that sampling $k \log k$ centers using k-means++, followed by a greedy center-pruning step, (introduced by Ostrovsky et al. (2006)) is sufficient to cluster well with high probability, assuming (α, ϵ)-approximation stability for balanced clustering. Our results improve over Agarwal et al. (2015), who showed this algorithm outputs a good clustering with probability $\Omega(\frac{1}{k})$ for standard (unbalanced) clustering under approximation stability. Formally, our result is the following.

Theorem 3. *Given $\frac{\epsilon \cdot k}{\alpha} < \rho < 1$, k-means++ seeding with a greedy pruning step outputs a solution that is $\frac{1}{1-\rho}$ close to the optimal solution with probability $> 1 - O(\rho)$, for clustering instances satisfying $(1+\alpha, \epsilon)$-approximation stability for the balanced k-means objective, with $\frac{L}{\ell} \in O(1)$.*

Intuitively, (α, ϵ)-approximation stability forces the clusters to become "spread out", i.e., the radius of any cluster is much smaller than the inter-cluster distances. This allows us to show for 2-means clustering, the k-means++ seeding procedure will pick one point from each cluster with high probability. However, if we induct on the number of clusters, the probability of success becomes exponentially small in k. We circumvent this issue in a manner similar to Ostrovsky et al. (2006), by sampling $k \log k$ centers, and carefully deleting centers greedily, until we are left with one center per cluster with high probability.

k-median and k-center: We show that the greedy thresholding algorithm of Balcan et al. (2013a) is sufficient to give a good clustering even for the balanced k-median or k-means objective, under approximation stability. At a high level, their algorithm works by first creating a threshold graph for a specific distance, and then iteratively picking the node with the highest degree in the threshold graph and removing its neighborhood. We show balanced clustering instances where the analysis in Balcan et al. (2013a) is not sufficient to guarantee good clusterings are outputted. We provide a new technique which overcomes the difficulties in adding upper and lower balance constraints. The technique involves showing there cannot be too many distinct pairs of points from different clusters which are close together, otherwise swapping these points between clusters would conserve the balance constraints and contradict approximation stability. We obtain the following theorem.

Theorem 4. *(1) There is an efficient algorithm which returns a valid clustering that is $O(\frac{\epsilon}{\alpha})$-close to the optimal, for balanced k-median or k-means clustering under $(1+\alpha, \epsilon)$-approximation stability, provided all clusters are size $\geq 3\epsilon n(1+\frac{3}{\alpha})$. (2) There is an efficient algorithm which returns the optimal clustering for balanced k-center under $(2, 0)$-approximation stability. (3) For $\epsilon > 0$, there does not exist an efficient algorithm which returns the optimal clustering for balanced k-center under $(2-\epsilon, 0)$-approximation stability, unless $NP = RP$.*

4 EFFICIENT CLUSTERING BY SUBSAMPLING

For datasets large enough to require a distributed learning system, it is expensive to apply a clustering algorithm to the entire dataset. In this section, we show that we can first cluster a small subsample of data and then efficiently extend this clustering to the remaining data. In our technique, each point in the dataset is assigned to the same p clusters as its nearest neighbor in the clustered subsample. In fact, this dispatch rule extends the clustering to the entire space $\mathcal{X}$ (not just to the unused portion of the training set), so at prediction time it can be used to send query points to the appropriate machines. We show that the clustering induced over $\mathcal{X}$ approximately inherits all of the desirable properties of the clustered subsample: good objective value, balanced clusters, and replication.

We measure the quality of a clustering of $\mathcal{X}$ as follows: given a data distribution μ over $\mathcal{X}$, our goal is to find a clustering with centers $C = \{c_1, \ldots, c_k\}$ and an assignment function $f : \mathcal{X} \to \binom{C}{p}$ for the entire space that minimizes $Q(f, C) = \mathbb{E}_{x\sim\mu}[\sum_{c_j \in f(x)} d(x, c_j)]$ subject to the balance constraints $\mathbb{P}_{x\sim\mu}(c_j \in f(x)) \in [\ell, L]$ for all j. In this section we focus on the k-median objective, but similar results for k-means are given in Section 12 of the supplementary material.

The simplest approach to extend a clustering of small subsample of data is to assign a new example x to the p clusters with the closest centers. This strategy incurs the lowest cost for new examples, but it may severely violate the balance constraints if the distribution is concentrated near one center.

Instead, given a clustering of the subsample S, our technique assigns a new example x to the same p clusters as its nearest neighbor in the set S, denoted by $NN_S(x)$. We use the fact that the clustering of S is balanced to show that the extended clustering is also balanced. Some points in S will represent more probability mass of μ than others, so we use a second independent sample S' to estimate weights for each point in S, which are used in a weighted version of the objective and balance constraints. Pseudocode is given in Algorithm 2. We obtain the following guarantees for k-median.

Theorem 5. *For any $\epsilon, \delta > 0$, let $(\bar{g}_S, C_S)$ be the output of Algorithm 2 with parameters k, p, ℓ, L and second sample size $n' = O\big(\frac{1}{\epsilon^2}(n + \log\frac{1}{\delta})\big)$. Let (f^*, C^*) be any clustering of $\mathcal{X}$ and (g_S^*, C_S^*) be an optimal clustering of S under Q_S satisfying the weighted balance constraints (ℓ, L). Suppose that $Q_S(g_S, C_S) \leq r \cdot Q_S(g_S^*, C_S^*) + s$. Then w.p. $\geq 1-\delta$ over the second sample, the output $(\bar{g}_S, C_S)$ satisfies the balance constraints with $\ell' = \ell - \epsilon$ and $L' = L + \epsilon$ and*

$$Q(\bar{g}_S, C_S) \leq r \cdot Q(f^*, C^*) + s + 2(r+1)pD\epsilon + p(r+1)\alpha(S) + r\beta(S, \ell+\epsilon, L-\epsilon),$$

where D is the diameter of $\mathcal{X}$, the quantity $\alpha(S) = \mathbb{E}_{x\sim\mu}[d(x, \mathrm{NN}_S(x))]$ measures how well μ is approximated by S, and $\beta(S, \ell, L) = \min_{\bar{h}, C}\left\{Q(\bar{h}, C) - Q(f^*, c^*)\right\}$ measures the loss incurred by restricting to clusterings that are constant over the Voronoi tiles of S.

The terms $\alpha(S)$, $\beta(S)$ can be bounded in terms of the size of S under natural conditions on the distribution μ. For example, when the distribution has doubling dimension d_0 and the optimal clustering of $\mathcal{X}$ is ϕ-probabilistically Lipschitz (Urner et al., 2011, 2013) (intuitively requiring that the probability mass close to the cluster boundaries is small) then for $n = \tilde{O}((\frac{1}{\epsilon\phi^{-1}(\epsilon)})^{d_0} d_0)$ we will have $\alpha(S) < D\epsilon$ and $\beta(S) < pD\epsilon$ with high probability. See Section 12 in the supplementary material for details.

5 EXPERIMENTS

In this section, we present an empirical study of the accuracy and scalability of our technique using both the LP rounding algorithms and k-means++ together with the nearest neighbor extension. We compare against three baselines: random

Input: Dataset $S = \{x_1, \ldots, x_n\}$, cluster parameters (k, p, ℓ, L), second sample size n'.

1. Draw second sample S' of size n' iid from μ.
2. For each point x_i, set $\hat{w}_i = |S'_i|/n'$, where $S'_i = \{x' \in S' : NN_S(x') = x_i\}$
3. Let $C_S = (c_1, \ldots, c_k)$ and $g_S : S \to \binom{C}{p}$ be a clustering of S obtained by minimizing

$$Q_S(g, C) = \sum_{i=1}^{n} \hat{w}_i \sum_{c_j \in g(x)} d(x_i, c_j)$$

subject to $\sum_{i: c_j \in g_n(x_i)} \hat{w}_i \in [\ell, L]$ for all $j = 1, \ldots, k$.

4. Return $\bar{g}_S(x) = g_S(NN_S(x))$ and centers C_S.

Algorithm 2: Nearest neighbor clustering extension.

partitioning, balanced partition trees, and locality sensitive hashing (LSH) on both synthetic and real world image and advertising datasets. Our findings are summarized below:

• Using small samples of the given datasets, we compare the clusterings produced by our LP rounding algorithms[2] and k-means++ (with balancing heuristics described shortly). We find that clusterings produced by k-means++ and the LP rounding algorithms have similar objective values and correlate well with the underlying class labels. These results complement the results of Section 3, showing that k-means++ produces high quality balanced clusterings for 'typical' data. This comparison is detailed in Sections 13 and 14 of the supplementary material. Based on this observation, our further empirical studies use k-means++.

• We compare the accuracy of our technique (using k-means++ and the nearest neighbor extension) to the three baselines for a wide range of values of k in large-scale learning tasks where each machine learns a local SVM classifier. For all values of k and all datasets, our algorithm achieves higher accuracy than all our baselines.

• We show that our framework exhibits strong scaling, meaning that if we double the available computing power, the total running time reduces by a constant fraction.

Experimental Setup: In each run of our experiment, one of the partitioning algorithms produces a dispatch rule from $10,000$ randomly sampled training points. This dispatch rule is then used to distribute the training data among the available worker machines. If the parameter k exceeds the number of machines, we allow each machine to process multiple partitions independently. Next we train a one-vs-all linear separator for each partition in parallel by minimizing the L2-regularized L2-loss SVM objective. This objective is minimized using Liblinear (Fan et al., 2008) when the data is small enough to fit in the each worker's memory, and L-BFGS otherwise (note that both solvers will converge to the same model). The regularization parameter is chosen via 5-fold cross validation. To predict the label of a new example, we use the dispatch rule to send it to the machine with the most appropriate model. All experimental results are averaged over 10 independent runs.

Details for our technique: Our method builds a dispatch rule by clustering a small sample of data using k-means++ and uses the nearest neighbor dispatch rule in order to dispatch both the training and testing data. To ensure a balanced partitioning, we apply the following simple balancing heuristics: while there is any cluster smaller than ℓn points, pick any such cluster and merge it with the cluster whose center is nearest. Then each cluster that is larger than Ln points is randomly partitioned into evenly sized clusters that satisfy the upper capacity constraint. This guarantees every cluster satisfies the capacity constraints, but the number of output clusters may differ from k. For the nearest neighbor dispatch, we use the random partition tree algorithm of Dasgupta and Sinha (2015) for efficient approximate nearest neighbor search. We set $\ell = 1/(2k)$ and $L = 2/k$ and $p = 1$, since our baselines do not support replication.

Baselines: We compare against the following baselines.[3]

Random Partitioning: Points are dispatched uniformly at random. This baseline produces balanced partitions but does not send similar examples to the same machine.

Balanced Partition Trees: Similarly to a kd-tree, this partitioning rule recursively divides the dataset by splitting it at the median point along a randomly chosen dimension. This is repeated until the tree has k leaves (where we assume k is a power of 2). This baseline produces balanced partitions and improves over random partitioning because each machine learns a local model for a different subset of the space. The drawback is that the partitioning may result in subsets that do not contain similar data points.

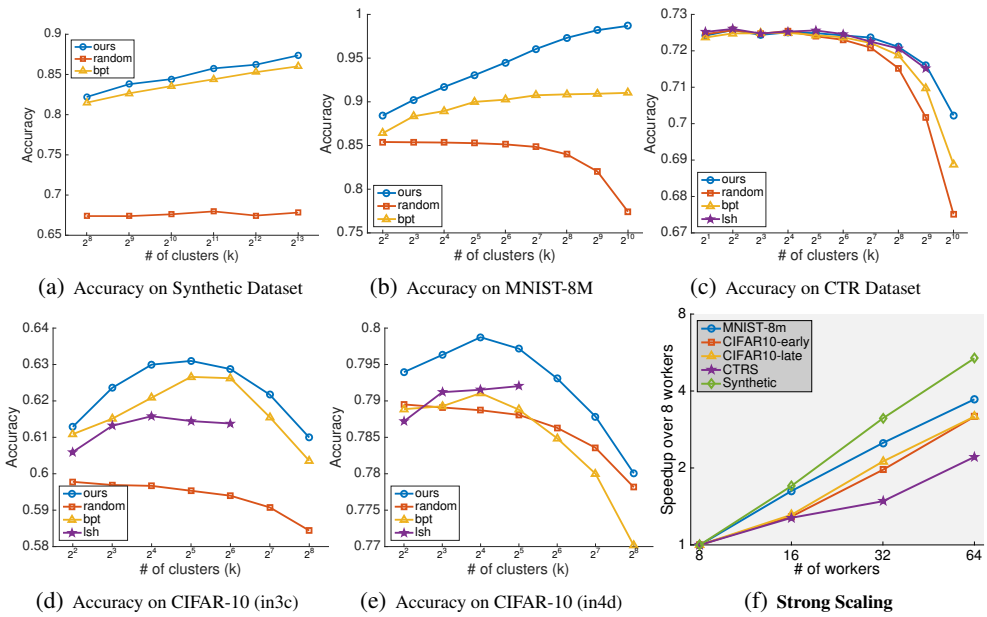
LSH Partitioning: This baseline uses locality sensitive hash functions (Andoni and Indyk, 2006) to dispatch similar points to the same machine. Given an LSH family H, we pick a random hash $h : \mathbb{R}^d \to \mathbb{Z}$. Then a point x is assigned to cluster $h(x) \mod k$. In our experiments, we use the concatenation of 10 random projections followed by binning (Datar et al., 2004). See Section 13 for details of the construction. This baseline sends similar examples to the same machine, but does not balance the cluster sizes (which is essential for practical data distribution).

Datasets: We use the following datasets:

Synthetic: We use a 128 GB synthetic dataset with 30 classes and 20 features. The data distribution is a mixture of 200 Gaussians with uniformly random centers in $[0, 1]^{20}$ with covariance $0.09I$. Labels are assigned so that nearby Gaus-

[2] We can run the LP rounding algorithm for small n, even though there are $O(n^2)$ variables.

[3] Since our framework does not communicate during training, we do not compare against algorithms that do, e.g. boosting (Balcan et al., 2012a).



(a) Accuracy on Synthetic Dataset (b) Accuracy on MNIST-8M (c) Accuracy on CTR Dataset

(d) Accuracy on CIFAR-10 (in3c) (e) Accuracy on CIFAR-10 (in4d) (f) **Strong Scaling**

Figure 3: Figures (a) through (e) show the effect of k on the classification accuracy. Figure (f) shows the speedup factor as we increase the number of workers from 8 to 64 for each dataset.

sians have the same label.

MNIST-8M: We use the raw pixels of the MNIST-8M dataset (Loosli et al., 2007). It has $8M$ examples and 784 features.

CIFAR-10: The CIFAR-10 dataset (Krizhevsky, 2009) is an image classification task with 10 classes. Following Krizhevsky et al. (2012) we include 50 randomly rotated and cropped copies of each training example to get a training set of 2.5 million examples. We extract the features from the Google Inception network (Szegedy et al., 2015) by using the output of an early layer (in3c) and a later layer (in4d).

CTR: The CTR dataset contains ad impressions from a commercial search engine where the label indicates whether the ad was clicked. It has 860K examples with 232 features.

Results: Our empirical results are shown in Figure 3. We do not report accuracies when the partitioning is imbalanced, specifically when the largest $k/2$ clusters contain more than 98% of the data. For all values of k and all datasets, our method has higher accuracy than all three baselines. The balanced partition tree is the most competitive baseline, but in Section 13 we present an additional synthetic distribution for which our algorithm drastically outperforms the balanced partition tree. For all datasets except CTR, the accuracy of our method increases as a function of k, until k is so large that each cluster becomes data starved. Our method combines the good aspects of both the balanced partition tree and LSH baselines by simultaneously sending similar examples to the same machines and ensuring that every machine gets roughly the same amount of data.

Figure 3(f) shows the speedup obtained when running our system using 16, 32, or 64 workers compared to using 8. We clock the time taken for the entire experiment: the time for clustering a subsample, dispatch, training and testing. In all cases, doubling the number of workers reduces the total time by a constant factor, showing that our framework strongly scales and can be applied to very large datasets.

6 CONCLUSION

In this work, we propose and analyze a new framework for distributed learning. Given that similar points tend to have similar classes, we partition the data so that similar examples go to the same machine. We cast the dispatching step as a clustering problem combined with novel fault tolerance and balance constraints necessary for distributed systems. We show the added constraints make the objective highly nontrivial, yet we provide LP rounding algorithms with provable guarantees. This is complemented by our results showing that the k-means++ algorithm is competitive on 'typical' datasets. These are the first algorithms with provable guarantees under both upper and lower capacity constraints, and may be of interest beyond distributed learning. We show that it is sufficient to cluster a small subsample of data and use a nearest neighbor extension technique to efficiently dispatch the remaining data. Finally, we conduct experiments for all our algorithms that support our theoretical claims, show that our framework outperforms several baselines and strongly scales.

Acknowledgements

This work was supported in part by NSF grants CCF-1451177, CCF-1422910, CCF-1535967, IIS-1618714, IIS-1409802, a Sloan Research Fellowship, a Microsoft Research Faculty Fellowship, a Google Research Award, Intel Research, Microsoft Research, and a National Defense Science & Engineering Graduate (NDSEG) fellowship.

References

Karen Aardal, Pieter L van den Berg, Dion Gijswijt, and Shanfei Li. Approximation algorithms for hard capacitated k-facility location problems. *European Journal of Operational Research*, (2):358–368, 2015.

Manu Agarwal, Ragesh Jaiswal, and Arindam Pal. k-means++ under approximation stability. *Theoretical Computer Science*, 588:37–51, 2015.

Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. In *Proceedings of the twenty-fifth ACM symposium on Principles of database systems*, pages 153–162, 2006.

Sara Ahmadian and Chaitanya Swamy. Approximation algorithms for clustering problems with lower bounds and outliers. In *Proceedings of the 43rd annual International Colloquium on Automata, Languages, and Programming*, 2016.

Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. In *Integer Programming and Combinatorial Optimization*, pages 52–63. Springer, 2014.

Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 459–468. IEEE, 2006.

Kevin Aydin, MohammadHossein Bateni, and Vahab Mirrokni. Distributed balanced partitioning via linear embedding. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 387–396. ACM, 2016.

Maria-Florina Balcan and Mark Braverman. Approximate nash equilibria under stability conditions. Technical report, 2010.

Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. *arXiv preprint arXiv:1204.3514*, 2012a.

Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity, and privacy. In *Conference on Learning Theory*, 2012b.

Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Clustering under approximation stability. *J. ACM*, 60(2):8:1–8:34, May 2013a. ISSN 0004-5411. doi: 10.1145/2450142.2450144. URL `http://doi.acm.org/10.1145/2450142.2450144`.

Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general communication topologies. In *Advances in Neural Information Processing Systems*, 2013b.

Maria-Florina Balcan, Vandana Kanchanapally, Yingyu Liang, and David Woodruff. Improved distributed principal component analysis. In *Advances in Neural Information Processing Systems*, 2014.

Maria-Florina Balcan, Nika Haghtalab, and Colin White. k-center clustering under perturbation resilience. In *Proceedings of the 43rd annual International Colloquium on Automata, Languages, and Programming*, 2016.

J. Barilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *Journal of Algorithms*, 15(3):385 – 415, 1993. ISSN 0196-6774. doi: http://dx.doi.org/10.1006/jagm.1993.1047. URL `http://www.sciencedirect.com/science/article/pii/S0196677483710473`.

Florian Bourse, Marc Lelarge, and Milan Vojnovic. Balanced graph edge partition. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1456–1465. ACM, 2014.

Jarosław Byrka, Krzysztof Fleszar, Bartosz Rybicki, and Joachim Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 722–736. SIAM, 2015a.

Jarosław Byrka, Krzysztof Fleszar, Bartosz Rybicki, and Joachim Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 722–736. SIAM, 2015b.

Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1999.

Brian F Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver, and Ramana Yerneni. Pnuts: Yahoo!'s hosted data serving platform. *Proceedings of the VLDB Endowment*, 1 (2):1277–1288, 2008.

Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. Lp rounding for k-centers with non-uniform hard capacities. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 273–282. IEEE, 2012.

Sanjoy Dasgupta and Kaushik Sinha. Randomized partition trees for exact nearest neighbor search. *Algorithmica*, 72(1):237–263, 2015.

M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

Daniel Delling, Andrew V Goldberg, Ilya Razenshteyn, and Renato F Werneck. Graph partitioning with natural cuts. In *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 1135–1146. IEEE, 2011.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Hierarchical placement and network design problems. In *FOCS*, pages 603–612. IEEE Computer Society, 2000. ISBN 0-7695-0850-2. URL `http://dblp.uni-trier.de/db/conf/focs/focs2000.html#GuhaMM00`.

Rishi Gupta, Tim Roughgarden, and C Seshadhri. Decompositions of triangle-dense graphs. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 471–482. ACM, 2014.

Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *Journal of the ACM (JACM)*, 50(6):795–824, 2003.

David R. Karger and Maria Minkoff. Building steiner trees with incomplete global knowledge. In *FOCS*, pages 613–623. IEEE Computer Society, 2000. ISBN 0-7695-0850-2. URL `http://dblp.uni-trier.de/db/conf/focs/focs2000.html#KargerM00`.

Samir Khuller and Yoram J. Sussmann. The capacitated k-center problem. In *In Proceedings of the 4th Annual European Symposium on Algorithms, Lecture Notes in Computer Science 1136*, pages 152–166. Springer, 1996.

Samory Kpotufe. The curse of dimension in nonparametric regression. 2010.

Robert Krauthgamer and James R Lee. Navigating nets: simple algorithms for proximity search. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807. Society for Industrial and Applied Mathematics, 2004.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Mu Li, David G Andersen, Alex J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pages 19–27, 2014.

Shanfei Li. An improved approximation algorithm for the hard uniform capacitated k-median problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 325–338, 2014a. doi: 10.4230/LIPIcs.APPROX-RANDOM.2014.325. URL `http://dx.doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.325`.

Shi Li. Approximating capacitated k-median with $(1+\epsilon)k$ open facilities. *arXiv preprint arXiv:1411.5630*, 2014b.

Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. *Large scale kernel machines*, pages 301–320, 2007.

Mohammad Mahdian and Martin Pál. Universal facility location. In *Algorithms-ESA 2003*, pages 409–421. Springer, 2003.

Rafail Ostrovsky, Yuval Rabani, Leonard J Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 165–176. IEEE, 2006.

Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Ruth Urner, Shai Shalev-Shwartz, and Shai Ben-David. Access to unlabeled data can speed up prediction time. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 641–648, 2011.

Ruth Urner, Sharon Wulff, and Shai Ben-David. Plal: Cluster-based active learning. In *Conference on Learning Theory*, pages 376–397, 2013.

Vladimir N. Vapnik and Leon Bottou. Local algorithms for pattern recognition and dependencies estimation. *Neural Computation*, 1993.

Kai Wei, Rishabh K Iyer, Shengjie Wang, Wenruo Bai, and Jeff A Bilmes. Mixed robust/average submodular partitioning: Fast algorithms, guarantees, and applications. In *Advances in Neural Information Processing Systems*, pages 2233–2241, 2015.

Y. You, J. Demmel, K. Czechowski, L. Song, and R. Vuduc. CA-SVM: Communication-avoiding support vector machines on clusters. In *IEEE International Parallel and Distributed Processing Symposium*, 2015.

Yuchen Zhang, John C. Duchi, and Martin Wainwright. Communication-efficient algorithms for statistical optimization. In *Neural Information Processing Systems*, 2012.

Yuchen Zhang, John Duchi, Michael Jordan, and Martin Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Neural Information Processing Systems*, 2013.

Appendix

7 RELATED WORK CONTINUED

7.0.1 Distributed Machine Learning

Currently, the most popular method of dispatch in distributed learning is random dispatch (Zhang et al., 2013, 2012). This may not produce optimal results because each machine must learn a global model. Another notion is to dispatch the data to pre-determined locations e.g., Yahoo!'s geographically distributed database, PNUTS (Cooper et al., 2008). However, it does not look at any properties of the data other than physical location.

In a recent paper, Wei et al. (2015) study partitioning for distributed machine learning, however, they give no formal guarantees on balancing the data each machine receives. You et al. (2015) use k-means clustering to distribute data for parallel training of support vector machines, but their clustering algorithms do not have approximation guarantees and are applied to the entire dataset, so their clustering step is much slower than ours. There is also work on distributed graph partitioning (Delling et al., 2011; Bourse et al., 2014; Aydin et al., 2016), in which the data points are set up in a graph structure, and must be distributed to different machines, minimizing the number of edges across machines. These techniques do not apply more generally for non graph-based objectives, e.g. k-means, k-median, or k-center.

7.0.2 Capacitated Clustering

Previous work in theoretical computer science has considered capacitated clustering, or clustering with upper bounds (Li, 2014b; Byrka et al., 2015b; Li, 2014a; An et al., 2014; Khuller and Sussmann, 1996; Cygan et al., 2012), and lower bounds (Aggarwal et al., 2006; Ahmadian and Swamy, 2016), but our algorithm is the first to solve a more general and challenging question of simultaneously handling upper and lower bounds on the cluster sizes, and p-replication.

• **k-center** The (uniform) capacitated k-center problem is to minimize the maximum distance between a cluster center and any point in its cluster subject to the constraint that the maximum size of a cluster is L. It is NP-Hard, so research has focused on finding approximation algorithms. Bar-Ilan et al. (Barilan et al., 1993) introduced the problem and presented the first constant factor polynomial time algorithm achieving a factor of 10, using a combinatorial algorithm which moves around clients until the capacities are satisfied, and the objective is approximately satisfied. The approximation factor was improved by Khuller et al. (Khuller and Sussmann, 1996). Cygan et al. (Cygan et al., 2012) give the first algorithm for capacitated k-center with non-uniform capacities by using an LP rounding algorithm. The appoximation factor is not explicitly computed, although it is mentioned to be in the order of hundreds. (An et al., 2014) follows a similar procedure but with a dynamic rounding procedure, and they improve to an approximation factor of 8. Further, for the special case of uniform capacities, they show a 6-approximation.

• **k-median** k-median with capacities is a notoriously difficult problem in clustering. It is much less understood than k-center with capacities, and uncapacitated k-median, both of which have constant factor approximations. Despite numerous attempts by various researchers, still there is no known constant factor approximation for capacitated k-median (even though there is no better lower bound for the problem than the one for uncapacitated k-median). As stated earlier, there is a well-known unbounded integrality gap for the standard LP even when violating the capacity or center constraints by a factor of $2 - \epsilon$ (Aardal et al., 2015).

Charikar et al. gave a 16-approximation when constraints are violated by a factor of 3 (Charikar et al., 1999). Byrka et al. improved this violation to $2+\epsilon$, while maintaining an $O(\frac{1}{\epsilon^2})$ approximation (Byrka et al., 2015a). Recently, Li improved the latter to $O(\frac{1}{\epsilon})$, specifically, when constraints are violated by $2 + \frac{2}{\alpha}$ for $\alpha \geq 4$, they give a $6 + 10\alpha$ approximation (Li, 2014a). These results are all for the *hard* capacitated k-median problem. In the *soft* capacities variant, we can open a point more than once to achieve more capacity, although each extra opening counts toward the budget of k centers. In hard capacities, each center can only be opened once. The hard capacitated version is more general, as each center can be replicated enough times so that the soft capacitated case reduces to the hard capacitated case. Therefore, we will only discuss the hard capacitated case.

All of the algorithms for capacitated k-median mentioned above share the same high-level LP rounding and aggregation idea but with different refinements in the algorithm and analysis.

Universal and load balanced facility location

In the facility location problem, we are given a set of demands and a set of possible locations for facilities. We should open facilities at some of these locations, and connect each demand point to an open facility so as to minimize the total cost of opening facilities and connecting demands to facilities. Capacitated facility location is a variant where each facility can supply only a limited amount of the commodity. This and other special cases are captured by the Universal Facility Location problem where the facility costs are general concave functions. Local search techniques (Mahdian and Pál, 2003) have been proposed and applied successfully. Also, LP rounding techniques suffer from unbounded integrality gap for capacitated facility location (Mahdian and Pál, 2003).

Load-balanced facility location (Karger and Minkoff, 2000), (Guha et al., 2000), is yet another variant where every open facility must cater to a minimum amount of demand. An

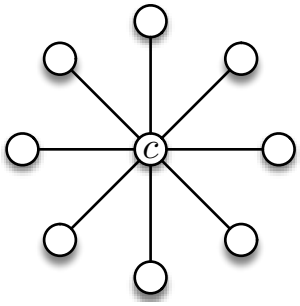


Figure 4: A graph in which the objective function strictly increases with k.

unconstrained facility location problem with modified costs is constructed and solved. Every open facility that does not satisfy the capacity constraint is closed and the demand is rerouted to nearby centers. The modified problem is constructed so as to keep this increase in cost bounded.

8 STRUCTURE OF BALANCED CLUSTERING

In this section, we show that adding lower bounds to clustering makes the problem highly nontrivial. Specifically, our main result is that the k-means, k-median, and k-center objective values may oscillate arbitrarily with respect to k (Theorem 8). in light of this structure, our results from Sections 2 and 3 are more surprising, since it is not obvious that algorithms with constant-factor guarantees exist.

We give a variety of clustering instances which do not have monotone cost functions with respect to k. For readability and intuition, these examples start out simple, and grow in complexity until we eventually prove Theorem 8.

First, consider a star graph with n points and lower bound ℓ, such that $n\ell \geq 3$ (see Figure 4).

The center c is at distance 1 to the $10n\ell$ leaves, and the leaves are at distance 2 from each other. When $k = 1$, each point is distance 1 to the center c. However as we increase k, the new centers must be leaves, distance 2 from all the other points, so $n\ell - 1$ points must pay 2 instead of 1 for each extra center. It is also easy to achieve an objective that strictly decreases up to a local minimum k', and then strictly increases onward, by adding k' copies of the center of the star.

Lemma 6. *Given a star graph with parameters n and ℓ such that $n\ell \geq 3$, then the cost of the k-means and k-median objectives strictly increase in k.*

Proof. Let the size of the star graph be n. Clearly, the optimal center for $k = 1$ is c. Then $\mathcal{OPT}_1 = n - 1$. Then for $k = 2$, we must choose another center p that is not c. p is distance 2 to all points other than c, so the optimal clustering is for p's cluster to have the minimum of $n\ell$ points, and c's cluster has the rest. Therefore, $\mathcal{OPT}_2 = n + n\ell - 2$.

This process continues; every time a new center is added, the new center pays 0 instead of 1, but $n\ell - 1$ new points must pay 2 instead of 1. This increases the objective by $n\ell - 2$. As long as $n\ell \geq 3$, this ensures the objective function is strictly increasing in k. □

Note for this example, the problem goes away if we are allowed to place multiple centers on a single point (in the literature, this is called "soft capacities", as opposed to enforcing one center per point, called "hard capacities"). The next lemma shows there can be a local minimum for hard capacities.

Lemma 7. *For all k', there exists a balanced clustering instance in which the k-means or k-median objective as a function of k has a local minimum at k'.*

Proof. Given $l \geq 3$, we create a clustering instance as follows. Define k' sets of points $G_1, \ldots, G_{k'}$, each of size $2n\ell - 1$. For any two points in some G_i, set their distance to 0. For any two points in different sets, set their distance to 1. Then for $1 \leq k \leq k'$, the objective value is equal to $(k' - k)(2n\ell - 1)$, since we can put k centers into k distinct groups, but $(k' - k)$ groups will not have centers, incurring cost $2n\ell - 1$. When $k > k'$, we cannot put each center in a distinct group, so there is some group G_i with two centers. Since $|G_i| = 2n\ell - 1$, the two centers cannot satisfy the capacity constraint with points only from G_i, so the objective value increases. □

Local maxima: So far, we have seen examples in which the objective decreases with k, until it hits a minimum (where capacities start to become violated), and then the objective strictly increases. The next natural question to ask, is whether the objective can also have a local maximum. As we saw in Section 2, this is possible. Here we present the formal proof. For convenience, we restate the statement of the lemma.

Lemma 1 (restated). *There exists a balanced clustering instance in which the k-center, k-median, and k-means objectives contain a local maximum with respect to k.*

Proof. Consider the graph in Figure 2, and let $n\ell = 21$. We claim that there exist valid clusterings using only length 1 edges for $k = 2$ and $k = 4$, but not $k = 3$. For $k = 2$, let the centers be y_1 and y_2. Then y_1 grabs the 20 red points (and itself) to hit the capacity of 21. y_2 grabs all the rest of the points. For $k = 4$, let the centers be x_1, x_2, x_3, x_4. Then each have edges to 20 middle points, non-overlapping, and WLOG x_1 grabs y_1 and y_2.

Now consider $k = 3$. The crucial property is that by construction, y_1 and any x_i cannot simultaneously be centers

and each satisfy the capacity to distance 1 points. This is because all x_i and y_1 are at minimum capacity, but y_1's neighbors overlap with neighbors from each x_i. So we cannot just take the centers from $k = 2$ and add a center from $k = 4$. The rest of the proof is checking that no other case works.

Case 1: the set of centers includes a point p not in $\{x_1, x_2, x_3, x_4, y_1, y_2\}$. The rest of the points are only distance 1 from exactly two points, so p cannot hit the lower bound of 21 using only distance 1 assignments.

Case 2: the set of centers is a subset of $\{x_1, x_2, x_3, x_4\}$. Then there are clearly 20 points which are not distance 1 from the three centers.

Case 3: the set of centers includes both y_1 and y_2. Then we need to pick one more center, x_i. x_i is distance 1 from 20 middle points, plus $\{x_1, x_2, x_3, x_4, y_1, y_2\}$, so 26 total. y_1 is also distance 1 from 20 middle points and $\{x_1, x_2, x_3, x_4, y_1, y_2\}$. y_1 and x_i share exactly 5 neighbors from the middle points, plus $\{x_1, x_2, x_3, x_4, y_1, y_2\}$ as neighbors. Then the union of points that x_i and y_1 are distance 1 from, is $26 + 26 - 11 = 41$, which implies that x_i and y_1 cannot simultaneously reach the lower bound of 21 with only distance 1 points.

Case 4: the set of centers does not include x_i nor y_j. By construction, for each pair x_i and y_j, there exists some middle points which are only distance 1 from x_i and y_j.

These cases are exhaustive, so we conclude $\mathcal{OPT}_3$ must be strictly larger than $\mathcal{OPT}_2$ and $\mathcal{OPT}_4$ (no matter what objective we use). □

The previous example does not work for the case of soft capacities, since the set of centers $\{x_1, y_2, y_2\}$ allows every point to have an edge to its center.

Now we prove our main theorem. Note, this theorem holds even for soft capacities.

Theorem 8. *For all $m \in \mathbb{N}$, there exists a balanced clustering instance in which the k-center, k-median, and k-means objectives contain m local maxima, even for soft capacities.*

Proof. We start with a proof sketch, and then we present the formal details. As in the previous lemma, we will construct a set of points in which each pair of points are either distance 1 or 2. It is convenient to define a graph on the set of points, in which an edge signifies a distance of 1, and all non-edges denote distance 2. We will construct a clustering instance where the objective value for all even values of k between $10m$ and $12m$ is low and the objective value for all odd values of k between $10m$ and $12m$ is high. The m odd values will be the local maxima. We will set the lower bound $n\ell$ to be the product of all the even integers between $10m$ and $12m$.

We start by creating a distinct set of "good" centers, X_k, for each even value of k between $10m$ and $12m$. Let X be the union of these sets. The set X_k contains k points which will be the optimal centers for a k-clustering in our instance. Then we will add an additional set of points, Y, and add edges from Y to the centers in X with the following properties.

1. For each even value of k between $10m$ and $12m$, there is an assignment of the points in Y to the centers in X_k so that points in Y are only assigned to adjacent centers and the capacity constraints are satisfied.
2. Each of the good centers in X is adjacent to no more than $\frac{6}{5} \cdot n\ell$ points in Y.
3. For each good center x in X_k, there is at least one point x' in every other set X'_k (for $k' \neq k$) so that the number of points in Y adjacent to both x and x' is at least $\frac{2}{5} \cdot n\ell$.
4. Any subset of the centers in X that does not contain any complete set of good centers X_k for some even k is non-adjacent to at least one point in Y.

Whenever we add a point to Y, we give it an edge to exactly one point from each X_k. This ensures that each X_k partitions Y. We first create connected components as in Figure 5 that each share $\frac{2}{5} \cdot n\ell$ points from Y, to satisfy Property 3.

For property 4, we add one additional point to Y for every combination of picking one point from each X_k. This ensures that any set which does not contain at least one point from each X_k will not be a valid partition for Y. Note that in the previous two steps, we did not give a single center more than $\frac{6}{5} \cdot n\ell$ edges, satisfying property 2. Then we add "filler" points to bring every center's capacity up to at least $n\ell$, which satisfies property 1.

Now we explain why properties 1-4 are sufficient to finish off the proof. Property 1 guarantees that the for each even value of k there is a clustering where the cost of each point in Y is one, which results in a good clustering objective.

Properties 2 and 3 guarantee that any set including a full X_k and a point from a different $X_{k'}$ cannot achieve cost 1 for each point without violating the capacities. Property 4 guarantees that any set without a full X_k cannot achieve cost 1 for each point. This completes the proof sketch.

Now we present the details in their entirety.

Setup. Set $k_{min} = 10 \cdot m$, and $k_{max} = 12m$. Define $K_{good} = \{k \mid k_{min} \leq k \leq k_{max} \text{ and } 2 \mid k\}$. Similarly, let $K_{bad} = \{k \mid k_{min} \leq k \leq k_{max} \text{ and } 2 \nmid k\}$. Note $|K_{bad}| = m$ and $|K_{good}| = m + 1$. For all $k \in K_{good}$, define $X_k = \{x_1^{(k)}, \ldots, x_{k'}^{(k)}\}$. Let $X = \bigcup_k X_k$.

Define $G = (V, E)$, $V = X \cup Y$, $X \cap Y = \emptyset$. Just like in the last proof, the edges later correspond to a distance of

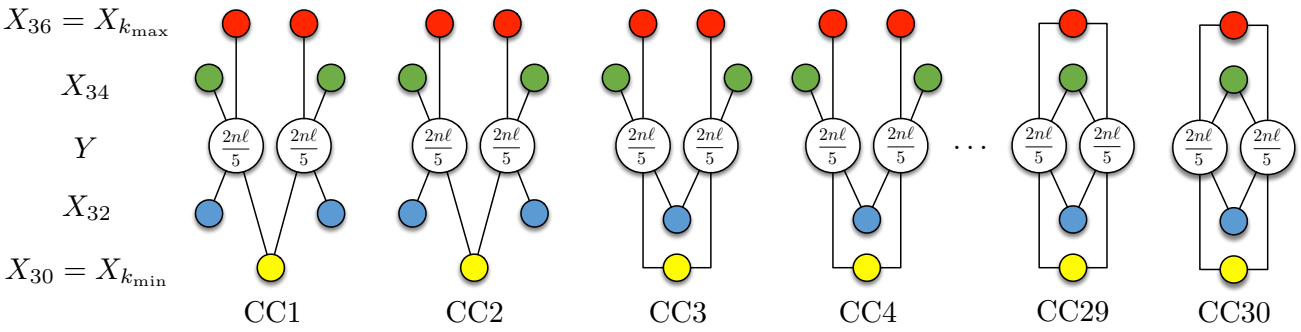


Figure 5: An example when $m = 3$. Each X_k is a different color. The white circles represent $2n\ell/5$ points from Y each.

1, and all other distances are 2. We will construct Y and E such that for all $k \in K_{good}$, all the neighbors of X_k form a partition of Y, i.e. $\forall k \in K_{good}$, $\bigcup_i N(x_i^{(k)}) = Y$ and $N(x_i^{(k)}) \cap N(x_j^{(k)}) = \emptyset$ for all $i \neq j$. So taking X_k as the centers corresponds to a k-clustering in which all points are distance 1 from their center. We will also show that for all $k \in K_{bad}$, it is not possible to find a valid set of centers for which every point has an edge to its center, unless the capacities are violated. This implies that all m points in K_{bad} are local maxima.

For all $k \in K_{good}$, $X_{k'}$ will have exactly $\frac{k_{max}}{k'} l$ edges in Y Thus, set $n\ell = \prod_{k \in K_{good}} k$ to make all of these values integral. Note that some points (those in $X_{k_{max}}$) have exactly $n\ell$ edges, and all points have $\leq \frac{6}{5} n\ell$ edges (which is tight for the points in $X_{k_{min}}$).

Now we define the main property which drives the proof. We say $x_{i_1}^{(j_1)}$ *overlaps* with $x_{i_2}^{(j_2)}$ if $N(x_{i_1}^{(j_1)}) \cup N(x_{i_2}^{(j_2)}) > \frac{2}{5} n\ell$. Note this immediately implies it is not possible to include them in the same set of centers such that each point has an edge to its center, since $N(x_{i_1}^{(j_1)}) \cup N(x_{i_2}^{(j_2)}) \leq N(x_{i_1}^{(j_1)}) + N(x_{i_2}^{(j_2)}) - N(x_{i_1}^{(j_1)}) \cap N(x_{i_2}^{(j_2)}) < 2 \cdot \frac{6}{5} n\ell - \frac{2}{5} n\ell = 2n\ell$.

Outline. We will construct Y in three phases. First, we add edges to ensure that for all $x_{i_1}^{(j_1)}$, for all $j_2 \neq j_1$, there exists an i_2 such that $x_{i_1}^{(j_1)}$ overlaps with $x_{i_2}^{(j_2)}$. It follows that if we are trying to construct a set of centers from X for $k' \in K_{bad}$, we will not be able to use any complete $X_{k'}$ as a subset. These are called the *backbone* edges.

The next phase is to add enough edges among points in different X_k's so that no subset of X (other than the $X_{k'}$'s) is a complete partition of Y. We will accomplish this by adding a bunch of points to Y shared by various $x \in X$, so that each x has edges to k_{max} points in Y. These are called the *dispersion* edges.

The final phase is merely to add edges so that all points reach their assigned capacity. We do this arbitrarily. These are called the *filler* edges.

Note whenever we add a point to Y, for all $k \in K_{good}$, we need to add an edge to exactly one $x \in X_k$, which will ensure that all X_k's form a partition of Y.

Phase 1: Backbone edges. Recall that for $k, k' \in K_{good}$, we want $\forall i$, $\exists j$ such that $x_i^{(k)}$ overlaps with $x_j^{(k')}$. Since $k_{max} = \frac{6}{5} k_{min}$, some x's will be forced to overlap with two points from the same X_k. However, we can ensure no point overlaps with three points from the same X_k.

We satisfy all overlappings naturally by creating k_{min} components, CC_1 to $CC_{k_{min}}$. Each component CC_i contains point $x_i^{(k_{min})}$. The rest of the sets X_k are divided so that one or two points are in each component, as shown in Figure 5. Formally, in component CC_i, sets $X_{k_{min}}$ to $X_{k_{min}+\lceil \frac{i}{2} \rceil}$ have one point in the component, and all other sets have two points in the component.

For each component CC_i, we add $\frac{4}{5} n\ell$ points to Y, split into two groups of $\frac{2}{5} n\ell$. The points from sets $X_{k_{min}+\lceil \frac{i}{2} \rceil}$ have edges to all $\frac{4}{5} n\ell$ points, and the points from the rest of the sets (since there are two from each set) have edges to one group of $\frac{2}{5} n\ell$ points. Therefore, for all $k, k' \in K_{good}$, each point $x \in X_k$ belongs to some component CC_i, and overlaps with some $x' \in X_{k'}$, so all of the overlapping requirements are satisfied (only using points within the same component).

This completes phase 1. Each point in X had at most $\frac{4}{5} n\ell$ edges added, so every point can still take at least $\frac{n\ell}{5}$ more edges in subsequent phases.

Phase 2: Dispersion edges. Now we want to add points to Y to ensure that no set of at most k_{max} points from X create a partition of Y, except sets that completely contain some X_k.

We have a simple way of achieving this. For every $(x_1, x_2, \ldots, x_{m+1}) \in X_{k_{min}} \times X_{k_{min}+2} \times \cdots \times X_{k_{max}}$, add one point to Y with edges to $x_1, x_2, \ldots, x_{m+1}$. Then we have added $\prod_{k \in K_{good}} k$ total points to Y in this phase.

This completes phase 2.

Phase 3: Filler edges. The final step is just to fill in the

leftover points, since we want every point $x_i^{(k)}$ to have $\frac{k_{min}}{k}l$ points total. All of the mechanisms for the proof have been set up in phases 1 and 2, so these final points can be arbitrary.

We greedily assign points. Give each point $x_i^{(k)} \in X$ a number $t_{x_i^{(k)}} = \frac{k_{min}}{k}n\ell - N(x_i^{(k)})$, i.e., the number of extra points it needs. Take the point $x \in X_k$ with the minimum t, and create t points in Y with x. For each layer other than X_k, add edges to the point with the smallest number. Continue this process until $t = 0$ for all points.

Final Proof. Now we are ready to prove that G has m local maxima. By construction, for all $k \in K_{good}$, X_k is a set of centers which satisfy the capacity constraints, and every point has an edge to its center. Now, consider a set C of centers of size $k' \in K_{bad}$. We show in every case, C cannot satisfy the capacity constraints with all points having edges to their centers.

Case 1: C contains a point $y \in Y$. y only has m edges, which is much smaller than $n\ell$.

Case 2: There exists $k \in K_{good}$ such that $X_k \subseteq C$. Then since $|C| \notin K_{good}$, $\exists x \in C \setminus X_k$. By construction, there exists $x_i^{(k)} \in X_k$ such that x and $x_i^{(k)}$ are overlapping. Therefore, both centers cannot satisfy the capacity constraints with points they have an edge to.

Case 3: For all $k \in K_{good}$, there exists $x \in X_k$ such that $x \notin C$. Take the set of all of these points, $x_1, x_2, \ldots, x_{m+1}$. By construction, there is a point $y \in Y$ with edges to only these points. Therefore, y will not have an edge to its center in this case.

This completes the proof.

□

9 DETAILS FROM SECTION 2

In this section, we provide the formal details for the bicriteria algorithm presented in Section 2.

For convenience, we restate theorem 2 here.

Theorem 2 (restated). *Algorithm 1 returns a constant factor approximate solution for the balanced k-clustering with p-replication problem for $p > 1$, where the upper capacity constraints are violated by at most a factor of $\frac{p+2}{p}$, and each point can be assigned to each cluster at most twice.*

Step 1 details:

We restate the LP for k-means and k-median for completeness, labeling each constraint for the proofs later.

$$\min \sum_{i,j \in V} c_{ij} x_{ij} \tag{LP.1}$$

$$\text{subject to: } \sum_{i \in V} x_{ij} = p, \ \forall j \in V \tag{LP.2}$$

$$\ell y_i \leq \sum_{j \in V} \frac{x_{ij}}{n} \leq L y_i, \ \forall i \in V \tag{LP.3}$$

$$\sum_{i \in V} y_i \leq k; \tag{LP.4}$$

$$0 \leq x_{ij} \leq y_i \leq 1, \ \forall i, j \in V. \tag{LP.5}$$

As mentioned in Section 2, it is well-known that the standard capacitated k-median LP (this LP, without the lower bound constraint and with $p = 1$) has an unbounded integrality gap, even when the capacities are violated by a factor of $2 - \epsilon$ (Aardal et al., 2015). The integrality gap is as follows. $k = 2nL - 1$, and there are nL groups of size $2nL - 1$. Points in the same group are distance 0, and points in different groups are distance 1. Fractionally, we can open $2 - \frac{1}{nL}$ facilities in each group to achieve cost 0. But integrally, some group contains at most 1 facility, and thus the capacity violation must be $2 - \frac{1}{nL}$.

However, with p replication, there must be p centers per group, so the balance violation can be split among the p centers. Therefore, the integrality is only unbounded when the capacities are violated by a factor of $\frac{p}{p-1}$

The k-center LP is a little different from the k-median/means LP. As in prior work (An et al., 2014; Cygan et al., 2012; Khuller and Sussmann, 1996), we guess the optimal radius, t. Since there are a polynomial number of choices for t, we can try all of them to find the minimum possible t for which the following program is feasible. Here is the LP for k-center.

$$\sum_{i \in V} x_{ij} = p, \qquad \forall j \in V \tag{2a}$$

$$n\ell y_i \leq \sum_{j \in V} x_{ij} \leq nL y_i, \qquad \forall i \in V \tag{2b}$$

$$\sum_{i \in V} y_i \leq k; \tag{2c}$$

$$0 \leq x_{ij} \leq y_i, \qquad \forall i, j \in V \tag{2d}$$

$$x_{ij} = 0 \qquad \text{if } d(i,j) > t. \tag{2e}$$

For k-median and k-means, let C_{LP} denote the objective value. For k-center, C_{LP} would be the smallest threshold t at which the LP is feasible, however we scale it as $C_{LP} = tnp$ for consistency with the other objectives. For all $j \in V$, define the connection cost C_j as the average contribution of a point to the objective. For k-median and k-means, it is $C_j = \frac{1}{p}\sum_{i \in V} c_{ij} x_{ij}$. That is, for k-median, it is the

average distance of a point to its fractional centers while for k-means, it is the average squared distance of a point to its fractional centers. For k-center, C_j is simply the threshold $C_j = t$. Therefore, $C_{LP} = \sum_{j \in V} pC_j$ in all cases.

The notation is summarized in table 1.

Step 2 details: Let $\mathcal{M}$ be the set of monarchs, and for each $u \in \mathcal{M}$, denote $\mathcal{E}_u$ as the empire of monarch u. Recall that the contribution of an assignment to the objective c_{ij} is $d(i,j)$ for k-median, $d(i,j)^2$ for k-means, and t for k-center. We also define a parameter $\rho = 1$ for k-center, $\rho = 2$ for k-median, and $\rho = 4$ for k-means, for convenience.

Initially set $\mathcal{M} = \emptyset$. Order all points in nondecreasing order of C_i. For each point i, if $\exists j \in \mathcal{M}$ such that $c_{ij} \leq 2tC_i$, continue. Else, set $\mathcal{M} = \mathcal{M} \cup \{i\}$. At the end of the for loop, assign each point i to cluster $\mathcal{E}_u$ such that u is the closest point in $\mathcal{M}$ to i. See Algorithm 3.

Input: V and fractional (x, y)
Output: Set of monarchs, $\mathcal{M}$, and empire $\mathcal{E}_j$ for each monarch $j \in \mathcal{M}$

$\mathcal{M} \leftarrow \emptyset$
Order all points in non-decreasing order of C_i
// Identify Monarchs
foreach $i \in V$ **do**
  if $\nexists j \in \mathcal{M}$ such that $c_{ij} \leq 2\rho C_i$ **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{i\}$
// Assign Empires as Voronoi partitions around monarchs
foreach $j \in V$ **do**
  Let $u \in \mathcal{M}$ be the closest monarch to j
  $\mathcal{E}_u \leftarrow \mathcal{E}_u \cup \{j\}$

Algorithm 3: Monarch procedure for coarse clustering: Greedy algorithm to create monarchs and assign empires

We obtain the following guarantees.

Lemma 9. *Let ρ be a parameter such that $\rho = 1$ for k-center, $\rho = 2$ for k-median, and $\rho = 4$ for k-means. The output of the monarch procedure satisfies the following properties:*

(1a) The clusters partition the point set;

(1b) Each point is close to its monarch: $\forall j \in \mathcal{E}_u, u \in \mathcal{M}, c_{uj} \leq 2\rho C_j$;

(1c) Any two monarchs are far apart: $\forall u, u' \in \mathcal{M}$ s.t. $u \neq u', c_{uu'} > 4\max\{C_u, C_{u'}\}$;

(1d) Each empire has a minimum total opening: $\forall u \in \mathcal{M}, \sum_{j \in \mathcal{E}_u} y_j \geq \frac{p}{2}$ (or for k-center, $\sum_{j \in \mathcal{E}_u} y_j \geq p$).

Proof of Lemma 9. The first three properties follow easily from construction (for the third property, recall we ordered the points at the start of the monarch procedure). Here is the proof of the final property, depending on the objective function.

For k-center and k-median, it is clear that for some $u \in \mathcal{M}$, if $d(i,u) \leq \rho C_u$, then $i \in \mathcal{E}_u$ (from the triangle inequality and Property (1c)). For k-means, however: if $d(i,u)^2 \leq 2C_u$, then $i \in \mathcal{E}_u$. Note that the factor is $\rho/2$ for k-means. This is because of the triangle inequality is a little different for squared distances.

To see why this is true for k-means, assume towards contradiction that $\exists i \in V$, $u, u' \in \mathcal{M}$, $u \neq u'$ such that $u \in \mathcal{E}_{u'}$ and $d(i,u)^2 \leq 2C_u$. Then $d(i,u') \leq d(i,u)$ by construction. Therefore, $d(u,u')^2 \leq (d(u,i)+d(i,u'))^2 \leq 4d(i,u)^2 \leq 8C_u$, and we have reached a contradiction by Property (1c).

Now, to prove property (1d):

k-center: From the LP constraints, for every u, $\sum_{j \in V} x_{ju} = p$. But x_{ju} is non-zero only they are separated by at most t, the threshold. Combining this with the fact that if $d(j,u) \leq C_u = t$, then $j \in \mathcal{E}_u$, we get, for each $u \in \mathcal{M}$:

$$\sum_{j \in \mathcal{E}_u} y_j \geq \sum_{j \in \mathcal{E}_u} x_{ju} = p$$

k-median and k-means: Note that C_u is a weighted average of costs c_{iu} with weights x_{iu}/p, i.e., $C_u = \sum_i c_{iu} {}^{x_{iu}}/_{p}$. By Markov's inequality,

$$\sum_{j : c_{ju} > 2C_u} \frac{x_{ju}}{p} < \frac{C_u}{2C_u} = \frac{1}{2}$$

Combining this with the fact that if $c_{ju} \leq 2C_u$, then $j \in \mathcal{E}_u$ for both k-median and k-means , we get, for each $u \in \mathcal{M}$:

$$\sum_{j \in \mathcal{E}_u} y_j \geq \sum_{j : c_{ju} \leq 2C_u} y_j \geq \sum_{j : c_{ju} \leq 2C_u} x_{ju} \geq \frac{p}{2}.$$

□

Step 3 Details: First we define a suboperation called Move, which is the standard way to transfer openings between points to maintain all LP constraints (Li, 2014a):

Definition 2 (Operation "Move")**.** *The operation "Move" moves a certain opening δ from a to b. Let (x', y') be the updated (x, y) after a movement of $\delta \leq y_a$ from a to b. Define*

$$\begin{aligned} y'_a &= y_a - \delta \\ y'_b &= y_b + \delta \\ \forall u \in V, x'_{au} &= x_{au}(1 - {}^{\delta}/_{y_a}) \\ \forall u \in V, x'_{bu} &= x_{bu} + x_{au} \cdot {}^{\delta}/_{y_a} \end{aligned}$$

It has been proven in previous work that the move operation does not violate any of the LP constraints except the constraint that $y_i \leq 1$ (Li, 2014a). We provide a proof below for completeness. Should we require $\delta \leq \min(y_a, 1 - y_b)$,

Table 1: Notation table

Symbol	Description	k-median	k-means	k-center
y_i	Fractional opening at center i	-		
x_{ij}	Fractional assignment of point j to center i	-		
c_{ij}	Cost of assigning j to center i	$d(i,j)$	$d(i,j)^2$	t
C_j	Avg cost of assignment of point j to all its centers	$\sum_i c_{ij}x_{ij}/p$		t
C_{LP}	Cost of LP	$\sum_j pC_j$		
ρ	parameter for monarch procedure	2	4	1

the constraint $y_i \leq 1$ would not be violated. But to get a bi-criteria approximation, we allow this violation. The amount by which the objective gets worse can then be bounded by the triangle inequality.

Lemma 10. *The operation Move does not violate any of the LP constraints except possibly the constraint $y_i \leq 1$ and the threshold constraint 2e of k-center.*

Proof. To show that the *Move* operation satisfies all the LP constraints, first note that the only quantities that change are $y_a, y_b, x_{au}, x_{bu},\ \forall u \in V$. Further, x, y satisfy all the constraints of the LP. Using this,

- Constraint LP.1: For every u, $\sum_i x'_{iu} = \sum_i x_{iu} = p$.
- Constraint LP.2 (1):

$$\sum_u x'_{au} = \sum_u x_{au}(1 - \delta/y_a) \leq nLy_a(1 - \delta/y_a) = nLy'_a$$
$$\sum_u x'_{bu} = \sum_u x_{bu} + \sum_u x_{au} \cdot \delta/y_a \leq nLy_b + nLy_a \cdot \delta/y_a = nLy'_b$$

- Constraint LP.2 (2):

$$\sum_u x'_{au} = \sum_u x_{au}(1 - \delta/y_a) \geq n\ell y_a(1 - \delta/y_a) = n\ell y'_a$$
$$\sum_u x'_{bu} = \sum_u x_{bu} + \sum_u x_{au} \cdot \delta/y_a \geq n\ell y_b + n\ell y_a \cdot \delta/y_a = n\ell y'_b$$

- Constraint LP.3: $\sum_i y'_i = \sum_i y_i \leq k$
- Constraint LP.4 (1):

$$x'_{au} = x_{au}(1 - \delta/y_a) \leq y_a(1 - \delta/y_a) = y'_a$$
$$x'_{bu} = x_{bu} + x_{au} \cdot \delta/y_a \leq y_b + y_a \cdot \delta/y_a = y'_b.$$

- Non-negative constraint: this is true since $\delta \leq y_a$.

□

Input: V, fractional (x, y), empires $\{\mathcal{E}_j\}$
Output: updated (x, y)
foreach $\mathcal{E}_u$ **do**
  Define $Y_u = \sum_{i \in \mathcal{E}_u} y_i$, $z_u = \frac{Y_u}{\lfloor Y_u \rfloor}$.
  while $\exists v$ *s.t.* $y_v \neq z_u$ **do**
    Let v be the point farthest from u with nonzero y_v.
    Let v' be the point closest to j with $y_{v'} \neq z_u$.
    Move $\min\{y_v, z_u - y_{v'}\}$ units of opening from y_v to $y_{v'}$.

Algorithm 4: Aggregation procedure

See Algorithm 4 for the aggregation procedure.

Note that by Property (1d), we have $Y_u \geq 1$ (whenever $p \geq 2$). Then by construction, $z_u \geq 1$. In each empire $\mathcal{E}_u$, start with the point i with nonzero y_i that is farthest away from the monarch u. Move its opening to the monarch u. Continue this process until u has opening exactly z_u, and then start moving the farthest openings to the point j closest to the monarch u. Continue this until the $\lfloor Y_u \rfloor$ closest points to u all have opening z_u. Call the new variables (x', y'). They have the following properties.

Lemma 11. *The aggregated solution (x', y') satisfies the following constraints:*

(2a) The opening of each point is either zero or in $[1, \frac{p+2}{2}]$: $\forall i \in V,\ 1 \leq y'_i < \frac{p+2}{p}$ or $y'_i = 0$;

(2b) Each cluster satisfies the capacity constraints: $i \in V,\ \ell y'_i \leq \sum_{j \in V} \frac{x'_{ij}}{n} \leq Ly'_i$;

(2c) The total fractional opening is k: $\sum_{i \in V} y'_i = k$;

(2d) Points are only assigned to open centers: $\forall i, j \in V,\ x'_{ij} \leq y'_i$;

(2e) Each point is assigned to p centers: $\forall i \in V,\ \sum_j x'_{ji} = p$;

(2f) The number of points with non-zero opening is at most k: $|\{i \mid y'_i > 0\}| \leq k$.

Proof. For the first property, recall that each cluster $\mathcal{E}_u$ has total opening $\geq \frac{p}{2}$, so by construction, all i with nonzero

y'_i has $y'_i \geq 1$. We also have $\frac{Y_u}{\lfloor Y_u \rfloor} \leq \frac{\lfloor Y_u \rfloor + 1}{\lfloor Y_u \rfloor} \leq \frac{p+2}{p}$, which gives the desired bound.

The next four properties are checking that the LP constraints are still satisfied (except for $y'_i \leq 1$). These follow from the fact that *Move* does not violate the constraints. The last property is a direct result of Properties (2a) and (2c). □

We obtain the following guarantee on the moving costs.

Lemma 12. *$\forall j \in V$ whose opening moved from i' to i,*

- *k-center: $d(i,j) \leq 5t$,*
- *k-median: $d(i,j) \leq 3d(i',j) + 8C_j$,*
- *k-means: $d(i,j)^2 \leq 15d(i',j)^2 + 80C_j$.*

Proof of Lemma 12. **k-center.** Use the fact that all $C_j = t$, and $x_{ij} > 0 \implies d(i,j) \leq t$ with property (1b) to get:

$$\begin{aligned} d(i,j) &\leq d(i,u) + d(u,i') + d(i',j) \\ &\leq 2C_i + 2C_{i'} + d(i',j) \leq 5t. \end{aligned}$$

k-median. By construction, if the demand of point j moved from i' to i, then $\exists u \in \mathcal{M}$ s.t. $i, i' \in \mathcal{E}_u$ and $d(u,i) \leq d(u,i')$. Denote j' as the closest point in $\mathcal{M}$ to j. Then $d(u,i') \leq d(j',i')$ because $i' \in \mathcal{E}_u$. Then,

$$\begin{aligned} d(i,j) &\leq d(i,u) + d(u,i') + d(i',j) \\ &\leq 2d(u,i') + d(i',j) \\ &\leq 2d(j',i') + d(i',j) \\ &\leq 2(d(j',j) + d(j,i')) + d(i',j) \\ &\leq 8C_j + 3d(i',j). \end{aligned}$$

k-means: The argument is similar to k-median, but with a bigger constant factor because of the squared triangle inequality.

$$\begin{aligned} d(i,j)^2 &\leq (d(i,u) + d(u,i') + d(i',j))^2 \\ &\leq (2d(u,i') + d(i',j))^2 \\ &\leq 4d(u,i')^2 + d(i',j)^2 + 4d(u,i')d(i',j) \\ &\leq 4d(u,i')^2 + d(i',j)^2 + 4d(u,i')d(i',j) \\ &\quad + (2d(i',j) - d(u,i))^2 \\ &\leq 5d(u,i')^2 + 5d(i',j)^2 \\ &\leq 5d(j',i')^2 + 5d(i',j)^2 \\ &\leq 5(d(j',j) + d(j,i'))^2 + 5d(i',j)^2 \\ &\leq 5d(j',j)^2 + 10d(i',j)^2 + 10d(j',j)d(i',j) \\ &\leq 5d(j',j)^2 + 10d(i',j)^2 + 10d(j',j)d(i',j) \\ &\quad + 5(d(j',j) - d(i',j))^2 \\ &\leq 10d(j',j)^2 + 15d(i',j)^2 \\ &\leq 80C_j + 15d(i',j)^2. \end{aligned}$$

□

Step 4 details: Set $\{i \mid y_i \neq 0\} = Y$. We show details of the min cost flow network in Algorithm 5.

Input: V, (x,y), y are integral
Output: updated (x,y) with integral x's and y's
Create a flow graph $G = (V', E)$ as follows.
Add each $i \in V$ to V', and give i supply p.
Add each $i \in Y$ to V', and give i demand $n\ell$.
Add a directed edge (i,j) for each $i \in V$, $j \in Y$, with capacity 2 and cost c_{ij} (for k-center, make the edge weight $5t$ if $d(i,j) \leq 5t$ and $+\infty$ otherwise.
Add a sink vertex v to V', with demand $np - kn\ell$.
Add a directed edge (i,v) for each $i \in Y$, with capacity $\lceil \frac{p+2}{p} nL \rceil - n\ell$ and cost 0.
Run an min cost integral flow solver on G.
Update x by setting x_{ij} to 0, 1, or 2 based on the amount of flow going from i to j.

Algorithm 5: Min cost flow procedure: Set up flow problem to round x's

Lemma 13. *There exists an integral assignment of the x'_{ij}'s such that $\forall i, j \in V$, $x'_{ij} \leq 2$ and it can be found in polynomial time.*

Proof. See Algorithm 5 and Figure 6 for the details of the flow construction.

In this graph, there exists a feasible flow: $\forall i, j \in V$, send x'_{ij} units of flow along the edge from i to j, and send $\sum_{j \in V} x_{ij}$ units of flow along the edge from i to v. Therefore, by the integral flow theorem, there exists a maximal integral flow which we can find in polynomial time. Also, by construction, this flow corresponds to an integral assignment of the x'_{ij}'s such that $x'_{ij} \leq 2$. □

Now we are ready to prove Theorem 2. The approximation ratios are 5, 11, and 95 for k-center, k-median, and k-means, respectively.

Proof of Theorem 2.
k-center: Recall that we defined $C_{LP} = tnp$, where t is the threshold for the k-center LP. From Lemma 12, when we reassign the demand of point j from i' to i, $d(i,j) \leq 5t$. In other words, the y-rounded solution is feasible at threshold $5t$. Then the k-center cost of the new y's is $np(5t) = 5C_{LP}$. From Lemma 13, we can also round the x's at no additional cost.

k-median: From Property 12, when we reassign the demand of point j from i' to i, $d(i,j) \leq 3d(i',j) + 8C_j$. Then we can bound the cost of the new assignments with respect to

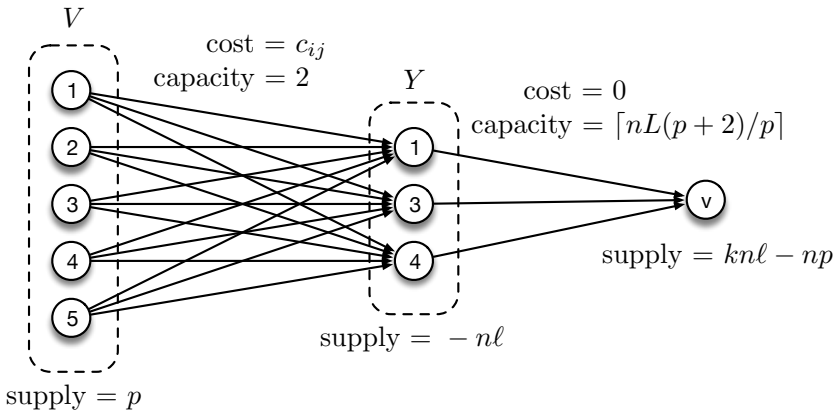


Figure 6: Flow network for rounding the x's: The nodes in each group all have the same supply, which is indicated below each group. The edge costs and capacities are shown above each group. The y-rounded solution gives a feasible flow in this network. By the Integral Flow Theorem, there exists a minimum cost flow which is integral and we can find it in polynomial time.

the original LP solution as follows.

$$\begin{aligned}\sum_{i\in V}\sum_{j\in V} d(i,j)x'_{ij} &\le \sum_{i\in V}\sum_{j\in V}(8C_j + 3d(i,j))x_{ij}\\ &\le \sum_{i\in V}\sum_{j\in V} 8C_j x_{ij}\\ &\quad + \sum_{i\in V}\sum_{j\in V} 3d(i,j)x_{ij}\\ &\le \sum_{j\in V} 8C_j \sum_{i\in V} x_{ij} + 3C_{LP}\\ &\le \sum_{j\in V} 8pC_j + 3C_{LP} \le 11C_{LP}.\end{aligned}$$

Then from Lemma 13, we get a solution of cost at most $11C_{LP}$, which also has integral x's.

k-means: The proof is similar to the k-median proof. From lemma 12, when we reassign the demand of point j from i' to i, $d(i,j)^2 \le 15d(i',j)^2 + 80C_j$. Then we can bound the cost of the new assignments with respect to the original LP solution as follows.

$$\begin{aligned}\sum_{i\in V}\sum_{j\in V} d(i,j)^2 x'_{ij} &\le \sum_{i\in V}\sum_{j\in V}(80C_j + 15d(i',j)^2)x_{ij}\\ &\le \sum_{i\in V}\sum_{j\in V} 80C_j x_{ij} + \sum_{i\in V}\sum_{j\in V} 15d(i,j)^2 x_{ij}\\ &\le \sum_{j\in V} 80C_j \sum_{i\in V} x_{ij} + 15C_{LP}\\ &\le \sum_{j\in V} 80pC_j + 15C_{LP} \le 95C_{LP}.\end{aligned}$$

Then from Lemma 13, we get a solution of cost at most $95C_{LP}$, which also has integral x's. □

See Algorithm 6 for the final algorithm.

Input: V
Output: Integral (x, y) corresponding to bicriteria clustering solution
1 Run a solver for the LP relaxation for k-median, k-means, or k-center, output (x, y).
2 Run Algorithm 3 with V, (x, y), output set of empires $\{\mathcal{E}_j\}$.
3 Run Algorithm 4 with V, $\{\mathcal{E}_j\}$, (x, y), output updated (x, y).
4 Run Algorithm 5 with V, (x, y), output updated (x, y).

Algorithm 6: Bicriteria approximation Algorithm for k-median, k-means, and k-center

10 k-CENTER

In this section, we present a more complicated algorithm that is specific to k-center, which achieves a true approximation algorithm - the capacity and replication constraints are no longer violated.

Approach

As in the previous section and in prior work (An et al., 2014; Cygan et al., 2012; Khuller and Sussmann, 1996), we start off by guessing the optimal distance t. Since there are a polynomial number of possibilities, it is still only polynomially expensive. We then construct the threshold graph $G_t = (V, E_t)$, with j being the set of all points, and $(x, y) \in E_t$ iff $d(x, y) \le t$.

A high-level overview of the rounding algorithm that follows is given in Algorithm 7.

Connection to the previous section: The algorithm here is similar to the bicriteria algorithm presented previously. There are, however, two differences. Firstly, we work only

with connected components of the threshold graph. This is necessary to circumvent the unbounded integrality gap of the LP (Cygan et al., 2012). Secondly, the rounding procedure of the y's can now move opening across different empires. Since the threshold graph is connected, the distance between any two adjacent monarchs is bounded and turns out to exactly be thrice the threshold. This enables us to get a constant factor approximation without violating any constraints.

Input: V: the set of points, k: the number of clusters, (ℓ, L): min and max allowed cluster size
Output: A k-clustering of V respecting cluster size constraints, p: replication factor
Procedure `balanced-k-center` (V, k, p, ℓ, L)
  foreach threshold t **do**
    Construct the threshold graph G_t
    foreach connected component $G^{(c)}$ of G_t **do**
      foreach k' in $1, ...k$ **do**
        // Solve balanced k'-clustering on $G^{(c)}$
        Solve `LPRound` $(G^{(c)}, k', p, \ell, L)$
    Find a solution for each $G^{(c)}$ with k_c centers such that $\sum_c k_c = k$ by linear search; call is s
    if no such a solution exists **then return** "No Solution Found"
    else return solution s
Procedure `LPRound` (G, k, p, ℓ, L)
  $(x, y) \leftarrow$ relaxed solution of LP in equation 3
  $(x', y') \leftarrow$ `yRound`(G, x, y)
  Round x' to get x'' from theorem 20
  return (x'', y')
Procedure `yRound` (G, x, y)
  Construct coarse clustering to get a tree of clusters from algorithm 8
  Round clusters in a bottom up manner in the tree, moving mass around to nodes within a distance of 5 away (algorithm 9)
  return rounded solution with integral y

Algorithm 7: Algorithm overview

The Algorithm

Intuition

The approach is to guess the optimal threshold, construct the threshold graph at this threshold, write and round several LPs for each connected component of this graph for different values of k. The intuition behind why this works is that at the optimal threshold, each cluster is fully contained within a connected component (by definition of the threshold graph).

We the round the opening variables, but this time, open exactly k centers. Most of the work goes into rounding the openings, and showing that it is correct. Then, we simply round the assignments using a minimum cost flow again.

Linear Program

As earlier, let y_i be an indicator variable to denote whether vertex i is a center, and x_{ij} be indicators for whether j belongs to the cluster centered at i. By convention, i is called a facility and j is called a client.

Consider the following LP relaxation for the IP for each connected component of G. Note that it is exactly the same as the one from the previous section, except it is described in terms of the threshold graph G. Let us call it `LP-`k`-center`(G):

$$\sum_{i \in V} y_i = k \tag{3a}$$

$$x_{ij} \leq y_i \qquad \forall i, j \in V \tag{3b}$$

$$\sum_{j:ij \in E} x_{ij} \leq nLy_i \qquad \forall i \in V \tag{3c}$$

$$\sum_{j:ij \in E} x_{ij} \geq n\ell y_i \qquad \forall i \in V \tag{3d}$$

$$\sum_{i:ij \in E} x_{ij} = p \qquad \forall j \in V \tag{3e}$$

$$x_{ij} = 0 \qquad \forall ij \notin E \tag{3f}$$

$$0 \leq x, y \leq 1 \tag{3g}$$

Once we have the threshold graph, for the purpose of k-center, all distances can now be measured in terms of the length of the shortest path in the threshold graph. Let $d_G(i, j)$ represent the distance between i and j measured by the length of the shortest path between i and j in G.

Connected Components

It is well known (Cygan et al., 2012) that even without lower bounds and replication, the LP has unbounded integrality gap for general graphs. However, for connected components of the threshold graph, this is not the case.

To begin with, we show that it suffices to be able to do the LP rounding procedure for only connected threshold graphs, even in our generalization.

Theorem 14. *If there exists an algorithm that takes as input a connected graph G, capacities ℓ, L, replication p, and k for which* `LP-`k`-center`(G_t) *is feasible, and computes a set of k centers to open and an assignment of every vertex j to p centers i such that $d_G(i, j) \leq r$ satisfying the capacity constraints, then we can obtain a r-approximation algorithm to the balanced k-centers problem with p-replication.*

Proof. Let connected component i have k_i clusters. For each connected component, do a linear search on the range

$[1, \ldots, k]$ to find values of k_i for which the problem is feasible. These feasible values will form a range, if size constraints are to be satisfied. To see why this is the case, note that if (x_1, y_1) and (x_2, y_2) are fractional solutions for $k = k_1$ and $k = k_2$ respectively, then $((x_1 + x_2)/2, (y_1 + y_2)/2)$ is a valid fractional soluion for $k = (k_1 + k_2)/2$.

Suppose the feasible values of k_i are $m_i \leq k_i \leq M_i$. If $\sum_i m_i > k$ or $\sum_i M_i < k$, return NO (at this threshold t). Otherwise, start with each k_i equal to m_i. Increase them one by one up to M_i until $\sum_i k_i = k$. This process takes polynomial time. □

From now on, the focus is entirely on a single connected component.

Rounding y

Given an integer feasible point to the IP for each connected component, we can obtain the desired clustering. Hence, we must find a way to obtain an integer feasible point from any feasible point of `LP-`k`-center`.

To round the y, we follow the approach of An et al.(An et al., 2014). The basic idea is to create a coarse clustering of vertices, and have the cluster centers form a tree. The radius of each cluster will be at most 2, and the length of any edge in the tree will exactly be three, by construction.

Now, to round the y, we first start from the leaves of the tree, moving opening around in each coarse cluster such that at most one node (which we pick to be the center, also called the monarch). In subsequent steps, this fractional opening is passed to the parent cluster, where the same process happens. The key to getting a constant factor approximation is to ensure that fractional openings that transferred from a child cluster to a parent cluster are not propagated further. Note that the bicriteria algorithm did not move opening from one coarse cluster (empire) to another because we didn't have an upper bound of the cost incurred by making this shift.

Preliminaries.: We start with some definitions.

Definition 3 (δ-feasible solution (Cygan et al., 2012))**.** *A solution (x, y) feasible on G^δ, the graph obtained by connecting all nodes within δ hops away from each other.*

Next, we introduce the notion of a distance-r shift. Intuitively, a distance-r shift is a series of movements of openings, none of which traverses a distance more than r in the threshold graph. Note that the definition is similar to what is used in An et al.(An et al., 2014).

Definition 4 (Distance-r shift)**.** *Given a graph $G = (V, E)$ and $y, y' \in \mathbb{R}^{|V|}_{\geq 0}$, y' is a distance-r shift of y if y' can be obtained from y via a series of disjoint movements of the form "Move δ from i to i'" where $\delta \leq \min(y_i, 1 - y_{i'})$ and every i and i' are at most a distance r apart in the threshold graph G. Further, if all y' are zero or one, it is called an integral distance-r shift.*

Note that, by the definition of a distance-r shift, each unit of y moves only once and if it moves more than once, all the movements are put together as a single, big movement, and this distance still does not exceed r.

Lemma 15 (Realizing distance-r shift)**.** *For every distance-r shift y' of y such that $0 \leq y'_i \leq 1 \; \forall i \in V$, we can find x' in polynomial time such that (x', y') is $(r+1)$-feasible.*

Proof. We can use the Move operation described earlier and in Cygan et al. (Cygan et al., 2012) to change the corresponding x for each such a movement to ensure that the resulting (x', y') are $(r+1)$-feasible. The additional restriction $\delta \leq 1 - y_b$ ensures that $y \leq 1$. Since each unit of y moves only once, all the movements put together will also lead a solution feasible in G^{r+1}, i.e. we get a $(r+1)$-feasible solution.

□

From here on, we assume that $x_{ij}, x_{i'j}$ are adjusted as described above for every movement between i and i'.

The algorithm to round y (An et al., 2014) proceeds in two phases. In the first phase, we cluster points into a tree of coarse clusters (monarchs) such that nearby clusters are connected using the monarch procedure of Khuller et al (Khuller and Sussmann, 1996). In the second phase, fractional opening are aggregated to get an integral distance-5 shift.

Monarch Procedure.: The monarch procedure presented a little differently but is very similar to the monarch procedure presented earlier. Since the threshold graph is connected, we can get guarantees on how big the distance between two monarchs is.

Algorithm 8 describes the first phase where we construct a tree of monarchs and assign empires to each monarch. Let $\mathcal{M}$ be the set of all monarchs. For some monarch, $u \in \mathcal{M}$, let $\mathcal{E}_u$ denote its empire. For each vertex i, let $m(i)$ denote the the monarch u to whose empire $\mathcal{E}_u$, i belongs.

The guarantees now translate to the following (Lemma 16):

- Empires partition the point set.
- The empire includes *all* immediate neighbors of a monarch and additionally, some other nodes of distance two (called dependents).
- Adjacent monarchs are exactly distance 3 from each other.

Lemma 16. *Algorithm 8, the monarch procedure is well-defined and its output satisfies the following:*

- $\mathcal{E}_u \cap \mathcal{E}_{u'} = \emptyset$.
- $\forall u \in \mathcal{M} : \mathcal{E}_u = N^+(u) \cup (\bigcup_{j \in N^+(u)} \texttt{Dependents}(j))$.

Input: $G = (V, E)$
Output: Tree of monarchs, $T = (\mathcal{M}, E')$, and empires for each monarch

```
Marked ← ∅
foreach j ∈ V do
    initialize ChildMonarchs(j) and Dependents(j) to ∅
Pick any vertex u and make it a monarch
E_u ← N^+(u); Initialize T to be a singleton node u
Marked ← Marked ∪ E_u
while ∃w ∈ (V \ Marked) such that d_G(w, Marked) ≥ 2 do
    Let u ∈ (V \ Marked) and v ∈ Marked such that d_G(u, v) = 2
    Make u a monarch and assign its empire to be E_u ← N^+(u)
    Marked ← Marked ∪ E_u
    Make u a child of m(v) in T
    ChildMonarchs(v) ← ChildMonarchs(v) ∪ {u}
foreach v ∈ (V \ Marked) do
    Let u ∈ Marked be such that d_G(u, v) = 1
    Dependents(u) ← Dependents(u) ∪ {v}
    E_{m(u)} ← E_{m(u)} ∪ {v}
```

Algorithm 8: Monarch Procedure: Algorithm to construct tree of monarchs and assign empires

- *The distance between a monarch and any node in its empire is at most 2.*
- *Distance between any two monarchs adjacent in T is exactly 3.*
- *If* $\texttt{ChildMonarchs}(j) \neq \emptyset$ *or* $\texttt{Dependents}(j) \neq \emptyset$*, then j is at distance one from some monarch.*

Proof. Note that the whole graph is connected and $V \neq \emptyset$. For the while loop, if there exists w such that $d_G(w, \texttt{Marked}) \geq 2$, there exists u such that $d_G(u, \texttt{Marked}) = 2$ because the graph is connected. By the end of the while loop, there are no vertices at a distance 2 or more from Marked. Hence, vertices not in Marked, if any, should be at a distance 1 from Marked. Thus, the algorithm is well defined.

Each time a new monarch u is created, $N^+(u)$ is added to its empire. This shows the first statement. The only other vertices added to any empire are the dependents in the foreach loop. Each dependent j is directly connected to i, a marked vertex. Hence, i has to be a neighbor of a monarch. If i were a monarch, j would have been marked in the while loop. Thus, $d_G(j, m(i)) = 2$.

If the first statement of the while loop, v is a marked vertex, and has to be a neighbor of some monarch $m(v)$. New monarch u is chosen such that $d_G(u, v) = 2$. The parent monarch of u is $m(v)$ and $d_G(u, m(v)) = d_G(u, v) + d_G(v, m(v)) = 3$.

□

Initial Aggregation.: Now, we shall turn to the rounding algorithm of An et al (An et al., 2014). The algorithm begins with changing y_u of every monarch $u \in \mathcal{M}$ to 1. Call this the initial aggregation. It requires transfer of at most distance one because the neighbors of the monarchs has enough opening.

Lemma 17. *The initial aggregation can be implemented by a distance-1 shift.*

Proof. For every vertex $u \in V$, we have $\sum_{j \in N(u)} y_j \geq \sum_{j \in N(u)} x_{uj} = p \geq 1$. Hence, there is enough y-mass within a distance of one from u. The actual transfer can happen by letting $\delta = \min(1 - y_u, y_j)$ for some neighbor j of u and then transferring δ from j to u. That is, $y_j = y_j - \delta$ and $y_u = y_u + \delta$.

□

Rounding.: The rounding procedure now proceeds in a bottom-up manner on the tree of monarchs, rounding all y using movements of distance 5 or smaller. After rounding the leaf empires, all fractional opening, if any is at the monarch. For internal empires, the centers of child monarch (remnants of previous rounding steps) and dependents are first rounded. Then the neighbors of the monarch are rounded to leave the entire cluster integral except the monarch. The two step procedure is adopted so that the opening propagated from this monarch to its parent originates entirely from the 1-neighborhood of the monarch.

Formally, at the end of each run of round on $u \in \mathcal{M}$, all the vertices of the set I_u are integral, where $I_u := (\mathcal{E}_u \setminus u) \cup (\bigcup_{j \in N(u)} \texttt{ChildMonarchs}(j))$.

The rounding procedure is described in detail in Algorithm 9. The following lemma states and proves that algorithm 9 rounds all points and doesn't move opening very far.

Lemma 18 (Adaptation of Lemma 19 of An et al (An et al., 2014))**.** *Let* $I_u := (\mathcal{E}_u \setminus u) \cup (\bigcup_{j \in N(u)} \texttt{ChildMonarchs}(j))$.

- $\texttt{Round}(u)$ *makes the vertices of I_u integral with a set of opening movements within $I_u \cup \{u\}$.*
- *This happens with no incoming movements to the monarch u after the initial aggregation.*
- *The maximum distance of these movements is five, taking the initial aggregation into account.*

Input: Tree of monarchs, T, and empires for each monarch after the initial aggregation
Output: y', an integral distance-5 shift of y

1 **Procedure** `Round` (Monarch u)
2 //Recursive call
3 **foreach** child w of u in T **do** `Round`(w)
4 //Phase 1
5 **foreach** $j \in N(u)$ **do**
6 $X_j \leftarrow \{j\} \cup$ `ChildMonarchs`$(j) \cup$ `Dependents`(j)
7 $W_j \leftarrow \{\lfloor y(X_j) \rfloor$ nodes from $X_j\}$; (Avoid picking j if possible)
8 `LocalRound` $(W_j, X_j, \emptyset)$
9 `LocalRound` $(\{j\}, X_j \setminus W_j, \emptyset)$
10 //Phase 2
11 $F = \{j | j \in N(u)$ and $0 < y_j < 1\}$
12 $W_F \leftarrow \{$ any $\lfloor y(F) \rfloor$ nodes from $F\}$
13 `LocalRound` $(W_F, F, \emptyset)$
14 //Residual
15 **if** $y(F \setminus W_F) > 0$ **then**
16 Choose $w^* \in F \setminus W_F$
17 `LocalRound` $(\{w^*\}, F \setminus W_F, u)$
18 **Procedure** `LocalRound` (V_1, V_2, V_3)
19 **while** $\exists i \in V_1$ such that $y_i < 1$ **do**
20 Choose a vertex w with non-zero opening from $V_2 \setminus V_1$
21 if there exists none, choose j from $V_3 \setminus V_1$
22 $\delta \leftarrow \min(1 - y_i, y_j)$
23 `Move` δ from j to i

Algorithm 9: Algorithm to round y

Proof. **Integrality.** From lemma 16, it can be seen that $X_j, j \in N(u)$ above form a partition of I_u. Hence, it suffices to verify that each node of every X_j is integral.

At the end of line 8, the total non-integral opening in X_j is $y(X_j) - \lfloor y(X_j) \rfloor$, and is hence smaller than one. Line 9 moves all these fractional openings to j. By now, all openings of $X_j \setminus \{j\}$ are integral.

Now, F is the set of all non-integral $j \in N(u)$. So, by the end of line 13, the total non-integral opening in $N(u)$ (and hence in all of I_u) is $y(F \setminus W_F) = y(F) - \lfloor y(F) \rfloor$, and is again smaller than one. If this is zero, we are done.

Otherwise, we choose a node w^*, shift this amount to w^* in line 17. To make this integral, this operations also transfers the *remaining amount, i.e.* $1 - y(F \setminus W_F)$ *from the monarch* u. If this happens, the monarch u's opening is no longer integral, but I_u's is.

This shows the first bullet. For the second one, notice that after the initial aggregation, this last operation is the only one involving the monarch u and hence, there are no other incoming movements into u.

Distance. In the first set of transfers in line 8 the distance of the transfer is at most 4. This is because dependents are a distance one away from j and child monarchs are at a distance two away. The maximum distance is when the transfer happens from one child monarch to another, and this distance is 4 (recall that there are no incoming movements into monarchs).

The transfers in line 9 moves openings from a child monarch or a dependent to j. The distances are 2 and 1 respectively. Accounting for the initial aggregation, this is at most 3.

The rounding on line 13 moves openings between neighbors of the monarch, i.e. from some j to j' where $j, j' \in N(u)$. So, the distance between j and j' is at most 2. From the preceding transfers, the openings at j moved a distance of at most three to get there, and thus, we conclude that openings have moved at most a distance of 5 so far.

The first step of rounding on line 17 moves openings from some j to w^*, where $j, w^* \in N(u)$. As above, the maximum distance in this case is 5. The second step of rounding on line 17 moves opening from the monarch u to its neighbor w^*. This distance is one, and after accounting for the initial aggregation, is 2.

From this, we see that the maximum distance any opening has to move is 5. □

The algorithms, their properties in conjunction with lemma 15 leads to the following theorem, which also summarizes this subsection.

Theorem 19. *There exists a polynomial time algorithm to find a 6-feasible solution with all y integral.*

Rounding x

Once we have integral y, rounding the x is fairly straightforward, without making the approximation factor any worse. Exactly the same procedure used in bicriteria algorithms works here too. But, we can have an easier construction since for k-center since we can use distances in the threshold graph instead.

Theorem 20. *There exists a polynomial time algorithm that given a δ-feasible solution (x, y) with all y integral, finds a δ-feasible solution (x', y) with all x' integral.*

Proof. We shall use a minimum cost flow network to this. Consider a directed bipartite graph (S, T, E'), where $S = V$ and $T = \{i : y_i = 1\}$ and $j \to i \in E'$ iff $x_{ij} > 0$. Add a dummy vertex s, with edges to every vertex in S, and t with edges from every vertex in T. In this network, let every edge of the bipartite graph have capacity 1. Further, all the $s \to S$ edges have capacity p. s supplies a flow of np units, while each $u \in T$ has a demand of l units. To ensure no excess demand or supply, t has a demand of $np - kl$. All the $t \to T$ edges have a capacity of $(L - \ell)$.

All the $s \to S$ edges have a cost of -1 and every other edge has a cost of zero. See figure 7.

Clearly, a feasible assignment (x, y) to $\texttt{LP}$-k-$\texttt{center}(G^\delta)$ with integral y is a feasible flow in this network. In fact, it is a minimum-cost flow in this network. This can be verified by the absence of negative cost cycles in the residual graph (because all negative cost edges are at full capacities).

Since, the edge capacities are all integers, there exists a minimum cost integral flow by the Integral Flow Theorem. This flow can be used to fix the cluster assignments.

□

Piecing together theorems 19 and 20, we have the following theorem:

Theorem 21. *Given an instance of the k-centers problem with p-replication and for a connected graph G, and a fractional feasible solution to $\texttt{LP}$-k-$\texttt{center}(G)$, there exists a polynomial time algorithm to obtain a 6-feasible integral solution. That is, for every i, j such that $x_{ij} \neq 0$, we have $d_G(i, j) \leq 6$.*

11 PROOFS FROM SECTION 3

k-means++:

In this section, we show that adding lower bounds to clustering is a high

we provide the full details for Theorem 3. We start by showing it is true for $k = 2$, and then we generalize to any k. First we need the following lemma. Given a point set S, let $\Delta_k(S)$ denote the optimal k-means cost of S.

Lemma 22 ((Ostrovsky et al., 2006))**.** *Given a set $S \subseteq \mathbb{R}^d$ and any partition $S_1 \cup S_2$ of S with $S_1 \neq \emptyset$. Let s, s_1, s_2 denote the centers of mass of S, S_1, and S_2, respectively. Then*

1. $\Delta_1^2(S) = \Delta_1^2(S_1) + \Delta_1^2(S_2) + \frac{|S_1||S_2|}{|S|} \cdot d(s_1, s_2)^2$
2. $d(s_1, s)^2 \leq \frac{\Delta_1^2(S)}{|S|} \cdot \frac{|S_2|}{|S_1|}$.

We define r_i as the radius of cluster C_i, i.e. $r_i = \frac{1}{|C_i|} \sum_{x \in C_i} d(x, c_i)$. Given a clustering instance S satisfying (α, ϵ)-approximation stability, with upper and lower bounds L and ℓ on the cluster sizes. We assume that $L \in O(\ell)$. For convenience, let $|C_i| = n_i$ for all i.

Lemma 23. $max(r_1^2, r_2^2) \leq O(\frac{\epsilon}{\alpha} \cdot \frac{L}{\ell}) d(c_1, c_2)^2$.

Proof. From part 2 of Lemma 22, we have $\Delta_1^2(X) = \Delta_2^2(X) + \frac{n_1 n_2}{n} \cdot d(c_1, c_2)^2$, which implies that $\frac{n}{n_1 n_2} \cdot \Delta_2^2(X) = d(c_1, c_2)^2 \frac{\Delta_2^2(X)}{\Delta_1^2(X) - \Delta_2^2(X)}$. Let c denote the center of mass of X. Then $\Delta_1^2(X) = \sum_{x \in X} d(c, x)^2 = n_1 d(c, c_1)^2 + n_2 d(c, c_2)^2 + \Delta_2^2(X) > \min(n_1, n_2)(d(c, c_1)^2 + d(c, c_2)^2) + \Delta_2^2(X) \geq \min(n_1, n_2) d(c_1, c_2)^2 + \Delta_2^2(X)$. Therefore, $\frac{n}{n_1} r_1^2 + \frac{n}{n_2} r_2^2 \leq \frac{\Delta_2^2(X)}{\min_i n_i \cdot d(c_1, c_2)} \leq \frac{n}{\min_i n_i} \cdot \frac{w_{avg}^2}{d(c_1, c_2)^2} = \frac{n}{\min_i n_i}$, and it follows that $\max(r_1^2, r_2^2) \leq O(\frac{\epsilon}{\alpha} \cdot \frac{L}{\ell}) d(c_1, c_2)^2$. □

Let $\rho = \frac{100\epsilon}{\alpha} < 1$. Now we define the core of a cluster C_i as $X_i = \{x \in C_i \mid d(x, c_i)^2 \leq \frac{r_i^2}{\rho}\}$. Then by a Markov inequality, $|X_i| \geq (1 - \rho) n_i$ for all i. This concentration inequality, along with Lemma 23 are the key structures needed to show k-means++ produces a good clustering. Recall in k-means++, we pick seeds $\{\hat{c}_1, \ldots, \hat{c}_k\}$ so that we pick point x for $\hat{c}_{i+1}$ with probability proportional to $d(x, \{\hat{c}_1, \ldots, \hat{c}_i\})^2$.

Lemma 24. *Assume $k = 2$. For sufficiently small ϵ, $Pr[(\hat{c}_1 \in X_1 \,\&\&\, \hat{c}_2 \in X_2) \,||\, (\hat{c}_2 \in X_1 \,\&\&\, \hat{c}_1 \in X_2)] = 1 - O(\rho)$.*

Proof. Wlog, we set $d(c_1, c_2)^2 = 1$ for ease of computation (scaling all distances does not affect the optimal clustering). Let $A = \sum_{x \in X_1, y \in X_2} d(x, y)^2$ and $B = \sum_{x, y \in X} d(x, y)^2$. Then the probability is A/B. Let c_1' and c_2' denote the centers of mass of X_1 and X_2, respectively. By Lemma 22, $d(c_1, c_1') \leq \frac{\Delta_1(C_i)}{|C_i|} \cdot \frac{10\epsilon/\alpha |C_i|}{(1 - 10\epsilon/\alpha)|C_i|} \leq r_i^2 O(\frac{\rho}{1-\rho}) \leq \frac{L}{\ell} \cdot O(\frac{\epsilon}{\alpha})(1 - O(\frac{\epsilon}{\alpha}))^{-2} d(c_1, c_2)$, so $d(c_1', c_2') \geq d(c_1, c_2) - d(c_1, c_1') - d(c_2, c_2') \geq 1 - O(\frac{L}{\ell} \cdot \frac{\epsilon}{\alpha})$.

Therefore, $A = \sum_{x \in X_1, y \in X_2} d(x, y) = |X_1| \Delta_1(X_2) + |X_2| \Delta_1(X_1) + |X_1||X_2| d(c_1', c_2')^2 \geq |X_1||X_2| d(c_1', c_2') \geq n_1 n_2 (1 - O(\frac{\epsilon}{\alpha}))^2 (1 - O(\frac{L}{\ell} \cdot \frac{\epsilon}{\alpha}))^2 \geq n_1 n_2 (1 - O(\frac{L}{\ell} \cdot \frac{\epsilon}{\alpha}))$.

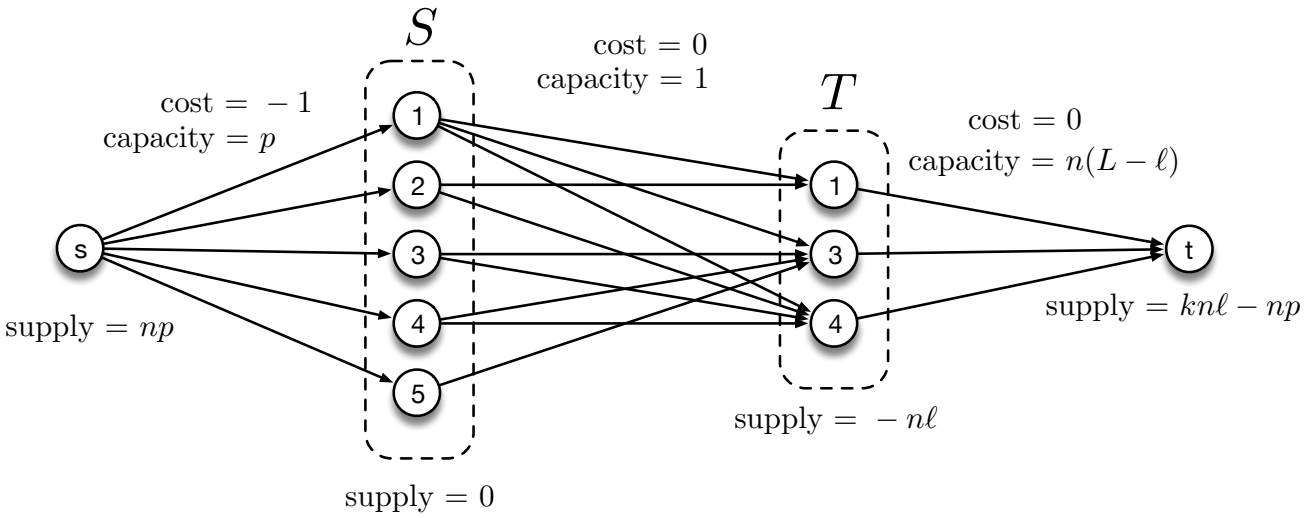


Figure 7: Minimum cost flow network to round x's. Each node in a group has the same supply, which is indicated below. The cost and capacity of each edge is indicated above.

$B = \sum_{x,y \in X} d(x,y)^2 = n^2 w_{avg}^2 + n_1 n_2 \leq n_1 n_2 (\frac{5}{4} \cdot \frac{\epsilon}{\alpha} \cdot \frac{n^2}{n_1 n_2} + 1) \leq n_1 n_2 (1 + \rho \frac{L^2}{\ell^2})$. Therefore, $\frac{A}{B} \geq \frac{1 - O(\frac{L}{\ell}\rho)}{1 + O(\frac{L}{\ell} \cdot \rho)}$. □

This lemma, combined with Lemma 3.3 from (Ostrovsky et al., 2006), immediately gives us the following theorem.

Theorem 25. *k-means++ seeding with a Lloyd step outputs a solution that is $\frac{1}{1-\rho}$ close to the optimal solution with probability $> 1 - O(\rho)$, for clustering instances satisfying $(1 + \alpha, \epsilon)$-approximation stability for the balanced 2-means objective, with $\frac{L}{\ell} \in O(1)$.*

Now that we have $k = 2$, we move to the case where $k \leq \frac{\epsilon}{\alpha}$. This assumption is reasonable for real-world data which does not have too many clusters and is sufficiently stable. We need this assumption so that we can set $\rho > \frac{\epsilon \cdot k}{\alpha}$. We still assume that $L \in O(\ell)$.

Assume that we sample k points $\hat{c}_1, \ldots \hat{c}_k$. We start with a lemma similar to Lemma 24.

Lemma 26. *The probability that $\hat{c}_1$ and $\hat{c}_2$ lie in the cores of different clusters is $1 - O(\rho)$.*

Proof. $A = \sum_{x \in X_i,\, y \in X_j} d(x,y)^2 = |X_i| \Delta_1(X_2) + |X_2| \Delta_1(X_1) + |X_1||X_2| d(c_i', c_j')^2 \geq n_1 n_2 (1 - O(\rho))^2$. $B = \sum_{x \in C_i, !y \in C_j} d(x,y)^2 = (n_i + n_j) \Delta_1(C_i \cup C_j) = (n_i + n_j)^2 (\frac{w_i^2 n_i + w_j^2 n_j}{n_i + n_j}) + n_i n_j d(c_i, c_j)^2 = n_i n_j (\frac{(n_i + n_j)^2}{n_i n_j} (\frac{w_i^2 n_i + w_j^2 n_j}{n_i + n_j}) + d(c_i, c_j)^2)$. Summing over all i, j, we have the following.

$$A/B = \frac{\sum_{i,j} n_i n_j}{\sum_{i,j} n_i n_j (1 + \frac{(w_i n_i + w_j n_j)(n_i + n_j)}{n_i n_j})}.$$

When $L \in O(\ell)$, we simplify the denominator: $\sum_{i,j} n_i n_j (1 + \frac{(w_i n_i + w_j n_j)(n_i + n_j)}{n_i n_j}) = \sum_{i,j} n_i n_j + \sum_{i,j} (n_i + n_j)(w_i n_i + w_j n_j) = \sum_{i,j} n_i n_j + O(\frac{L}{\ell}) \cdot \frac{n}{k} \sum_{i,j} (w_i n_i + w_j n_j) = \sum_{i,j} n_i n_j + O(\frac{L}{\ell}) \cdot \frac{n}{k} \sum_{i,j} 2n \frac{\epsilon}{\alpha} = O(1 + \rho) \sum_{i,j} n_i n_j$ when $\rho > \frac{\epsilon}{\alpha}$.

Therefore, $A/B = 1 - O(\rho)$. □

Lemma 27.

$$Pr[\hat{c}_{i+1} \in \bigcup_{j \notin 1, \ldots, i} X_j \mid \hat{c}_1, \ldots, \hat{c}_i \text{ lie in the cores of } X_1, \ldots X_i] = 1 - O(\rho).$$

Proof. $A = \sum_{j=m+1}^{k} \sum_{x \in X_j} d(x, \hat{C})^2$, and $B = \sum_{j=1}^{k} \sum_{x \in C_j} d(x, \hat{C})$. Let

$$\phi = \max_{j \geq m+1} [\max_{x \in X_j} d(x, C_j) / d(c_j, \hat{C})],$$

From Lemma 23, we have that $\phi \leq \max_{i,j} \frac{(\sqrt{\frac{\epsilon}{\alpha}}) d(c_i, c_j)}{d(c_i, c_j) - O(\sqrt{\frac{\epsilon}{\alpha}}) d(c_i, c_j)} \leq \frac{O(\sqrt{\frac{\epsilon}{\alpha}})}{1 - O(\sqrt{\frac{\epsilon}{\alpha}})} \leq 1$. Then for all points in the core of a cluster, $d(x, \hat{C}) \geq (1 - \phi) d(c_j, \hat{C})$. Then $A \geq \sum_{j=m+1}^{k} (1 - O(\rho)) n_j (1 - \phi)^2 d(c_j, \hat{C}) \geq (1 - \rho - 2\phi) \sum_{j=m+1}^{k} n_j d(c_j, \hat{C})^2$. $B \leq \sum_{j=1}^{k} (\Delta_1(C_j) + n_j d(c_j, \hat{c}_{p_j})^2) \leq \Delta_k(V) + \sum_{j=1}^{m} n_j d(c_j, x_j)^2 + \sum_{j=m+1}^{k} n_j d(c_j, \hat{C})^2 \leq \Delta_k(V) + \frac{1}{\rho} \sum_{j=1}^{m} \Delta_1(C_j) + \sum_{j=m+1}^{k} n_j d(c_j, \hat{C}) \leq \frac{1}{\rho} \Delta_k(V) + \sum_{j=m+1}^{k} n_j d(c_j, \hat{C})$ If we set $\rho = \Omega(\frac{L}{\ell} \cdot \epsilon k \alpha)$, then $A/B = 1 - O(\rho)$. □

Lemma 28. *Given we have sampled points $\{\hat{x}_1, \ldots, \hat{x}_i\}$, and let $C_1, \ldots C_m$ be all the clusters whose outer core contains some point $\hat{x}_j$. Then Pr$[\hat{x}_{i+1} \in \cup_{j=m+1}^{k} X_j] \geq 1 - 5\rho$.*

Proof. This follows from the previous lemma. □

Now we analyze the case where we sample more than k points. Let $N = \frac{2k}{1-5\rho}$.

Lemma 29. *Say we sample N points. The probability that for all $i \in [k]$, there is some sampled point in X_i', is $\geq 1-\rho$.*

Proof. The proof follows from Lemma 28. □

Finally, we perform a greedy removal phase. We refer the reader to (Ostrovsky et al., 2006) since the analysis is the same. This finishes the proof of Theorem 3.

k-median: Given a clustering instance (S, d) which satisfies $(1+\alpha, \epsilon)$-approximation stability with respect to balanced k-median, for some (l, L, k). We denote $\mathcal{C} = \{C_1, \ldots, C_k\}$ as the optimal partition and $c_1, \ldots, c_k$ as the optimal centers. Denote w_{avg} as the average distance from a point to its center in the optimal solution. Given a point x, define $w(x)$ as the distance to its center, and in general, for all i, denote $w_i(x)$ as the distance to the center c_i of cluster C_i in the optimal solution. Note, we will discuss the p-replication variant at the end of this section.

Lemma 30. *Assume the size of the optimal clusters are $> 4\epsilon n(1 + \frac{6}{\alpha})$. Then,*

- *For $< \frac{\epsilon n}{2}$ points x, there exists a set of $\geq \frac{\epsilon n}{2}$ points y from different clusters such that $d(x, y) \leq \frac{\alpha w_{avg}^2}{\epsilon}$.*
- *$< \frac{6\epsilon n}{\alpha}$ points x have $w(x) > \frac{\alpha w_{avg}^2}{6\epsilon}$.*

Proof.
- Assume the statement is false. Then there exist $\frac{\epsilon n}{2}$ pairs of distinct points (x, y) such that $d(x, y) \leq \frac{\alpha w_{avg}^2}{\epsilon}$ (for example, because of Hall's theorem). Now we create a new partition $\mathcal{C}'$ by swapping the points in each pair, i.e., for a pair $x \in C_i$ and $y \in C_j$, put x into C_j and y into C_i. Then for a single swap, the increase in cost is $w_j(x) - w_i(x) + w_i(y) - w_j(y) \leq w_j(y) + \frac{\alpha w_{avg}^2}{\epsilon} - w_i(x) + w_i(x) + \frac{\alpha w_{avg}^2}{\epsilon} - w_j(y) = \frac{2\alpha w_{avg}^2}{\epsilon}$. Therefore, the total difference in cost from $\mathcal{OPT}_\Phi$ to $\Phi(\mathcal{C}')$ is then $\alpha\mathcal{OPT}_\Phi$. Note that since we only performed swaps, $\mathcal{C}'$ still satisfies all the capacity constraints of Φ. Furthermore, $(C)'$ is ϵ-close to $\mathcal{OPT}_\Phi$ (since all clusters are size $> 2\epsilon n$, the optimal bijection between $(C)'$ and $\mathcal{OPT}_\Phi$ is the identity). This contradicts the approximation stability property.
- This follows simply from Markov's inequality.

□

Define $x \in C_i$ as a good point if there are less than $\frac{\epsilon}{2}$ points $y \notin C_i$ such that $d(x, y) \leq \frac{\alpha w_{avg}^2}{\epsilon}$, and $w(x) < \frac{\alpha w_{avg}^2}{6\epsilon}$. If a point is not good, then call it bad. Denote the set of bad points by B. For all i, denote $X_i = \{x \in C_i \mid x \text{ is good}\}$. Then Lemma 30 implies that if the optimal clusters are size $> 2\epsilon n$, $|B| < \epsilon n(1 + \frac{6}{\alpha})$.

Now we show that the optimal centers are sufficiently far apart.

Lemma 31. *Assume for all i, $|C_i| > \frac{7\epsilon n}{\alpha}$. Then for all i, j, $d(c_i, c_j) > \frac{2}{3} \cdot \frac{\alpha w_{avg}^2}{\epsilon}$.*

Proof. Given a cluster C_i. Since there are $< \frac{\epsilon n}{2}$ and $< \frac{6\epsilon n}{\alpha}$ points which do not satisfy properties 1 and 2 from Lemma 30, then at least one point from each cluster satisfies both properties, i.e., is a good point. Then given i, j, let $x_i \in C_i$ and $x_j \in C_j$ be good points. Then $d(c_i, c_j) > d(x_i, x_j) - d(x_i, c_i) - d(x_j, c_j) \geq \frac{\alpha w_{avg}^2}{\epsilon} - 2 \cdot \frac{\alpha w_{avg}^2}{6\epsilon} = \frac{2}{3} \cdot \frac{\alpha w_{avg}^2}{\epsilon}$. □

Combining Lemmas 30 an 31 implies that the threshold graph with distance $\frac{2}{3} \cdot \frac{\alpha w_{avg}^2}{\epsilon}$ will contain mostly "good" edges between good points from the same cluster. The rest of the argument for showing correctness of the algorithm is similar to the analysis in (Balcan et al., 2013a). We include it in Appendix 11 for completeness.

The following lemma is similar to Lemma 3.5 from (Balcan et al., 2013a). We need to assume the clusters are larger, but our proof generalizes to *capacitated* k-median.

Lemma 32. *Assume the optimal cluster sizes are $\geq \frac{\epsilon n}{2}(1 + \frac{3}{\alpha})$. For $\tau = \frac{\alpha w_{avg}^2}{3\epsilon}$, the threshold graph G_τ has the following properties:*

- *There is an edge between any two good points x, y in the same cluster C_i,*
- *There is not an edge between any two good points x, y in different clusters; furthermore, these points do not even share a neighbor in G_τ.*

Proof.
- Since x and y are both good, $d(x, y) \leq w(x) + w(y) \leq \frac{\alpha w_{avg}^2}{6\epsilon} + \frac{\alpha w_{avg}^2}{6\epsilon} \leq \frac{\alpha w_{avg}^2}{3\epsilon}$ by the triangle inequality.
- Assume x and y have a common neighbor z. Consider a point y_2 from y's cluster such that $w(y_2) \leq \frac{\alpha w_{avg}^2}{6\epsilon}$. By assumption, there are at least $\frac{\epsilon n}{2}$ such points. Furthermore, $d(x, y_2) \leq d(x, z) + d(z, y) + d(y, y_2) \leq 2\tau + w(y) + w(y_2) \leq \frac{2\alpha w_{avg}^2}{3\epsilon} + \frac{\alpha w_{avg}^2}{6\epsilon} + \frac{\alpha w_{avg}^2}{6\epsilon} = \frac{\alpha w_{avg}^2}{\epsilon}$. Since x is close to at least $\frac{\epsilon n}{2}$ points from different clusters, x cannot be a good point, so we have reached a contradiction.

□

Then the threshold graph is as follows. Each X_i forms a clique, the neighborhood of X_i is exactly $X_i \cup B$, and for all

$i \neq j, N(X_i) \cup N(X_j) = \emptyset$. This facilitates an algorithm for clustering in this setting, following analysis that is similar to (Balcan et al., 2013a).

Lemma 33 ((Balcan et al., 2013a)). *There is an efficient algorithm such that, given a graph G satisfying the properties of Lemma 32, and given $b \geq |B|$ such that each $|X_i| \geq b+2$, outputs a k-clustering with each X_i in a distinct cluster.*

The proof of this theorem, given in (Balcan et al., 2013a), depends solely on the properties of Lemma 32.

Lemma 34. *There is an efficient algorithm such that if a clustering instance satisfies $(1+\alpha, \epsilon)$-approximation stability for the balanced k-median objective and all clusters are size $\geq 3\epsilon n(1+\frac{3}{\alpha})$, then the algorithm will produce a clustering that is $O(\frac{\epsilon}{\alpha})$-close to the optimal clustering.*

Proof. First we use the proof of Theorem 3.7 from (Balcan et al., 2013a) (which assumes w_{avg}^2 is known) to the point where they achieve error $O(\frac{\epsilon}{\alpha})$ by using their version of Lemma 33. They go on to lower the error to ϵ, but this part of the proof breaks down for us. For the case of unknown w_{avg}^2, we can use the same fix as in the proof of Theorem 3.8 from (Balcan et al., 2013a). □

However, there is a problem with Lemma 34: even though it returns a clustering with small error, the capacity constraints might be violated. We can fix the capacity violations if we double our error.

Lemma 35. *Given an $\epsilon' > 0$ and a clustering $\mathcal{C}'$, if $\mathcal{C}'$ is ϵ'-close to the optimal clustering, then it is possible in polynomial time to construct a* valid *clustering $\mathcal{C}''$ which is $2\epsilon'$-close to the optimal clustering.*

Proof. For each cluster C_i' in $\mathcal{C}$, let v_i be the number of points for which C_i' violates the capacity, i.e., $|C_i'| - L$ or $l - |C_i'|$ or 0. Clearly, $\sum_i v_i \leq \epsilon' n$, or else $\mathcal{C}'$ would not be ϵ'-close to the optimal. Then we can iteratively take a point from the largest cluster, and place it into the smallest cluster. In each iteration, $\sum_i v_i$ is reduced by at least 1. So in $\leq \epsilon' n$ rounds, we reach a valid clustering, and the error is only ϵ' more than $\mathcal{C}'$. □

Part 1 of Theorem 4 follows immediately from the previous lemma.

Note that all of the proofs in this section can be trivially extended to the case where there is p-replication. However, p-replication does not mesh well with stability assumptions. Although Theorem 4 works completely under p-replication, all but an ϵ-fraction of the data appears to have "trivial" replication, in which there are $\frac{k}{p}$ groups of data, each of which have nearly the same p centers. This makes the problem similar to a $\frac{k}{p}$-clustering problem, up to ϵn points. The reason for this phenomenon is the following. If good points x_i and x_j share center c in addition to having other centers c_i and c_j, then by the triangle inequality, c_i and c_j are close together. This would contradict Lemma 31 unless each pair of good points either have p centers in common, or no centers in common.

Examples of balanced approximation stable instances:

In this section, we explicitly construct clustering instances which satisfy approximation stability for balanced clustering, but not for approximation stability for unbalanced clustering.

Given $n, \alpha, \epsilon, \ell, L$, and let $k = 2$. Denote the two optimal centers as c_1 and c_2. Let $d(c_1, c_2) = 1.9$. We place $x \leq \frac{\epsilon n}{\alpha}$ points at distance 1 from c_1, and .9 from c_2. Call this set of points A. Then we place $\ell n - x$ points at distance 0 from c_1 (denote by B_1), and we place the rest of the points at distance 0 from c_2 (denote by B_2). We need to assume that $\ell n - x \geq 0$.

Then for balanced clustering, $C_1 = A \cup B_1$, and $C_2 = B_2$, because C_1 must contain at least ℓn points. The optimal cost is x. For standard clustering, $C_1 = B_1$, and $C_2 = A \cup B_2$, and the optimal cost is $.9x$. This clustering instance is not $(\frac{10}{9}, \frac{x}{n})$approximation stable for standard clustering: all points in A can move from C_2 to C_1, incurring a cost of $.1x$ to the objective.

However, this clustering instance is $(1+\alpha, \epsilon)$-approximation stable for the balanced clustering objective. Moving any point to a different cluster incurs a cost of at least 1. Given a partition with cost $(1+\alpha)x = x + \alpha \cdot x \leq x + \alpha\frac{\epsilon n}{\alpha} \leq x + \epsilon n$. Then less than ϵn points have switched clusters.

k-center:

Now we will prove the rest of Theorem 4. Given a clustering instance, denote its optimal balanced k-center radius by r^*.

Lemma 36. *Given a clustering instances satisfying $(2, 0)$-approximation stability for balanced k-center, then for all $p \in C_i$, $q \in C_j$, $i \neq j$, $d(p, q) > r^*$.*

Proof. Assume the claim is false. Then $\exists p \in C_i$, $q \in C_j$ such that $d(p, q) \leq r^*$ and $i \neq j$. Then consider the optimal clustering, except switch p for q. So the clusters are $C_i \cup \{q\} \setminus \{p\}$, $C_j \cup \{p\} \setminus \{q\}$, and the other $k-2$ clusters are the same as in $\mathcal{OPT}$. This clustering achieves a balanced k-center radius of $2r^*$. The only points with a new distance to their center are p and q. By the triangle inequality, $d(c_i, q) \leq d(c_i, p) + d(p, q) \leq 2r^*$ and $d(c_j, p) \leq d(c_j, q) + d(q, p) \leq 2r^*$. Furthermore, since the updated clusters are still the same size as C_i and C_j, all the balance constraints still hold. But this gives us a contradiction under $(2, 0)$-approximation stability, since we have found a valid clustering of cost $2r^*$ which is different from $\mathcal{OPT}$. □

From this lemma, there is a simple algorithm to optimally cluster instances satisfying $(2, 0)$-approximation stability

for balanced k-center. We merely need to create the threshold graph for threshold distance r^* and output the connected components. Since every point is distance $\leq r^*$ to their center by definition, all points in an optimal cluster will be in the same connected component. By Lemma 36, no two points from different clusters will appear in the same connected component.

The final part of Theorem 4 is to prove hardness of balanced k-center under $(2-\epsilon, 0)$-approximation stability, for all $\epsilon > 0$, unless $NP = RP$. Note that there does not exist an efficient algorithm for $(2-\epsilon)$-approximation stability, if the algorithm takes in ℓ and L as parameters. This is a corollary of Theorem 10 in (Balcan et al., 2016), by setting $\ell = 0$ and $L = 1$.

However, we can show something stronger: we can show no algorithm exists, even for the special case when $\ell = L$. We show the analysis from (Balcan et al., 2016) carries over in this case.

Given a hard balanced-3-Dimensional Matching instance X_1, X_2, X_3, T, where $|X_1| = |X_2| = |X_3| = m$, and each $t \in T$ is a triple $t = (x_1, x_2, x_3)$, $x_1 \in X_1$, $x_2 \in X_2$, $x_3 \in X_3$. We modify the reduction to balanced-perfect-dominating set as follows. We start by converting it to a graph $G = (V, E)$ in the same way as (Balcan et al., 2016). Now we make four copies of this graph, G_1, G_2, G_3, G_4. For each vertex v in G_1 corresponding to an element in T (denote this set by GT_1), we add edges from v to its other three copies in G_2, G_3 and G_4. Call the resulting graph G'. Note this reduction is still parsimonious. It has the additional property that if a dominating set of size $|T| + 3|M|$ exists, then each vertex in the dominating set hits exactly 4 vertices. First, assume a 3-matching for the 3DM instance exists. Then we can achieve a dominating set of size $|T| + 3|M|$. Pick the vertices corresponding to the 3-matching, there are $4|M|$ such vertices, each of which have edges to the 3 elements they represent. We also put in the dominating set, the $|T| - |M|$ elements in GT_1 that are not in the 3-matching. Each of these elements have edges to their 3 copies in G_2, G_3, and G_4. This makes a full dominating set of size $|T| + 3|M|$. If there is no 3-matching for the 3DM instance, then the dominating set must be strictly larger. Finally, the reduction from Unambiguous-Balanced Perfect Dominating Set to clustering is the exact same proof, but now since each vertex in the dominating set hits 4 vertices, we get that each cluster is size exactly 4.

12 PROOFS FROM SECTION 4

Each clustering of the entire space $\mathcal{X}$ can be represented as a pair (f, C) where $C = \{c_1, \ldots, c_k\}$ is a set of centers and $f : \mathcal{X} \to \binom{C}{p}$ is a function that assigns each point in $\mathcal{X}$ to p of the k centers. We measure the quality of a clustering of $\mathcal{X}$ as follows: given a data distribution μ over $\mathcal{X}$, our goal is to find a clustering with centers C and an assignment function $f : \mathcal{X} \to \binom{C}{p}$ that minimizes either the k-median objective $Q^{(1)}$ or the k-means objective $Q^{(2)}$ given by

$$Q^{(1)}(f, C) = \mathop{\mathbb{E}}_{x\sim\mu}\left[\sum_{i\in f(x)} d(x, c_i)\right]$$

and

$$Q^{(2)}(f, C) = \mathop{\mathbb{E}}_{x\sim\mu}\left[\sum_{i\in f(x)} d(x, c_i)^2\right],$$

subject to the constraint that each cluster has probability mass between ℓ and L. Specifically, we require for each cluster index i that $\mathbb{P}_{x\sim\mu}(i \in f(x)) \in [\ell, L]$. Throughout this section, we use the notation Q, Q_n, and $\hat{Q}_n$ as a placeholder for either the k-median or k-means objective.

In our algorithm, each point x in the subsample S acts as a representative for those points in $\mathcal{X}$ that are closer to it than any other sample point (i.e., its cell in the Voronoi partition induced by S). Since each sample point might represent more or less of the probability mass of μ, we consider the following weighted clustering problem of a dataset S. A clustering of the data set S is a pair (g, C) for some set of centers $C = \{c_1, \ldots, c_k\}$ and an assignment function $g : S \to \binom{C}{p}$ that assigns each point of S to p of the centers. The weight for point x_i is $w_i = \mathbb{P}_{x\sim\mu}(\mathrm{NN}_S(x) = x_i)$, where $\mathrm{NN}_S(x)$ denotes the nearest neighbor in S to the point x. The weighted k-median and k-means objectives on S are given by

$$Q_n^{(1)}(g, c) = \sum_{j=1}^{n} w_j \sum_{i\in g(x_j)} d(x_j, c(i))$$

and

$$Q_n^{(2)}(g, c) = \sum_{j=1}^{n} w_j \sum_{i\in g(x_j)} d(x_j, c(i))^2,$$

where the subscript n refers to the size of the subsample S. The weighted capacity constraints require that the total weight of each cluster i, given by $\sum_{j:i\in g(x_j)} w_j$, is between ℓ and L. Since the distribution μ is unknown, our algorithm uses a second sample drawn from μ to estimate the weights. Given estimates $\hat{w}_1, \ldots, \hat{w}_n$ of the true weights, the estimated k-median and k-means objective functions are

$$\hat{Q}_n^{(1)}(g, c) = \sum_{j=1}^{n} \hat{w}_j \sum_{i\in g(x_j)} d(x_j, c(i))$$

and

$$\hat{Q}_n^{(2)}(g, c) = \sum_{j=1}^{n} \hat{w}_j \sum_{i\in g(x_j)} d(x_j, c(i))^2,$$

and the estimated weight of a cluster is $\sum_{j:i\in g(x_j)} \hat{w}_j$. Finally, for any clustering (g, c) of S, define the nearest neighbor extension to be $(\bar{g}, c)$ where $\bar{g}(x) = g(\mathrm{NN}_S(x))$. The

assignment function $\bar{g}$ assigns each point in $\mathcal{X}$ to the same p clusters as its nearest neighbor in S.

Finally, we introduce notation for the various relevant classes of cluster assignment functions. For lower and upper bounds on cluster sizes ℓ and L, we denote the set of cluster assignments of the entire space $\mathcal{X}$ that satisfy the capacity constraints by

$$F(\ell, L) = \left\{ f : \mathcal{X} \to \binom{k}{p} \ : \ \mathop{\mathbb{P}}_{x \sim \mu}(i \in f(x)) \in [\ell, L] \, \forall i \right\}.$$

Similarly, for the samples S, for true and estimated weights, define the sets of feasible assignments respectively as:

$$G_n(\ell, L) = \left\{ g : S \to \binom{k}{p} \ : \ \sum_{j : i \in g(x)} w_j \in [\ell, L] \, \forall i \right\}$$

$$\hat{G}_n(\ell, L) = \left\{ g : S \to \binom{k}{p} \ : \ \sum_{j : i \in g(x)} \hat{w}_j \in [\ell, L] \, \forall i \right\}.$$

Before proving the main results in the paper, we need the following lemmas. We first show that if we take the second sample size n' to be $\tilde{O}(n/\epsilon^2)$, then with high probability the error in any sum of the estimated weights $\hat{w}_j$ is at most ϵ.

Lemma 37. *For any $\epsilon > 0$ and $\delta > 0$, if we set $n' = O\big(\frac{1}{\epsilon^2}(n + \log \frac{1}{\delta})\big)$ in Algorithm 2, then with probability at least $1 - \delta$ we have $\left|\sum_{i \in I}(w_i - \hat{w}_i)\right| \le \epsilon$ uniformly for all index sets $I \subset [n]$.*

Proof. Let V_i be the cell of point x_i in the Voronoi partition induced by S. For any index set $I \subset [n]$, let V_I denote the union $\bigcup_{i \in I} V_i$. Since the sets $V_1, \ldots, V_n$ are disjoint, for any index set I we have that $\mu(V_I) = \sum_{i \in I} w_i$ and $\hat{\mu}(V_I) = \sum_{i \in I} \hat{w}_i$, where $\hat{\mu}$ is the empirical measure induced by the second sample S'. Therefore it suffices to show uniform convergence of $\hat{\mu}(V_I)$ to $\mu(V_I)$ for the 2^n index sets I. Applying Hoeffding's inequality to each index set and the union bound over all 2^n index sets, we have that

$$\mathbb{P}\left(\sup_{I \subset [n]} \left| \sum_{i \in I} w_i - \hat{w}_i \right| > \epsilon \right) \le 2^n e^{-2n'\epsilon^2}.$$

Setting $n' = O\big(\frac{1}{\epsilon^2}(n + \log \frac{1}{\delta})\big)$ results in the right hand side being equal to δ. □

Next we relate the weighted capacity constraints and objective over the set S to the constraints and objective over the entire space $\mathcal{X}$.

Lemma 38. *Let (g, c) be any clustering of S that satisfies the weighted capacity constraints with parameters ℓ and L. Then the nearest neighbor extension $(\bar{g}, c)$ satisfies the capacity constraints with respect to μ with the same parameters. For the k-median objective we have*

$$|Q_n^{(1)}(g, c) - Q^{(1)}(\bar{g}, c)| \le p \mathop{\mathbb{E}}_{x \sim \mu}\big[d(x, \mathrm{NN}_S(x))\big],$$

and for the k-means objective we have

$$Q_n^{(2)}(g, c) \le 2Q^{(2)}(\bar{g}, c) + 2p \mathop{\mathbb{E}}_{x \sim \mu}\big[d(x, \mathrm{NN}_S(x))^2\big]$$

and

$$Q^{(2)}(\bar{g}, c) \le 2Q_n^{(2)}(g, c) + 2p \mathop{\mathbb{E}}_{x \sim \mu}\big[d(x, \mathrm{NN}_S(x))^2\big].$$

Proof. The fact that $\bar{g}$ satisfies the population-level capacity constraints follows immediately from the definition of the weights $w_1, \ldots, w_n$.

By the triangle inequality, k-median objective over $\mathcal{X}$ with respect to μ can be bounded as follows

$$\begin{aligned} Q^{(1)}(\bar{g}, c) &\le \mathop{\mathbb{E}}_{x \sim \mu}\left[\sum_{i \in \bar{g}(x)} d(x, \mathrm{NN}_S(x)) \right] \\ &\quad + \mathop{\mathbb{E}}_{x \sim \mu}\left[\sum_{i \in \bar{g}(x)} d(\mathrm{NN}_S(x), c(i)) \right] \\ &= p \mathop{\mathbb{E}}_{x \sim \mu}[d(x, \mathrm{NN}_S(x))] + Q_n^{(1)}(g, c). \end{aligned}$$

The reverse inequality follows from an identical argument applying the triangle inequality to $Q_n^{(1)}$.

For the k-means objective, the result follows similarly by using the following approximate triangle inequality for squared distances: $d(x, z)^2 \le 2(d(x, y)^2 + d(y, z)^2)$, □

We are now ready to prove Theorem 39. The statement below is modified to include the k-means objective.

Theorem 39. *For any $\epsilon > 0, \delta > 0$, let $(\bar{g}_n, c_n)$ be the output of Algorithm 2 with parameters k, p, ℓ, L and second sample size $n' = O\big((n + \log 1/\delta)/\epsilon^2\big)$. Let (f^*, c^*) be any clustering of $\mathcal{X}$ and (g_n^*, c_n^*) be an optimal clustering of S under $\hat{Q}_n$ satisfying the estimated weighted balance constraints (ℓ, L). Suppose the algorithm used to cluster S satisfies $\hat{Q}(g_n, c_n) \le r \cdot \hat{Q}(g_n^*, c_n^*) + s$. Then w.p. $\ge 1 - \delta$ over the second sample the output $(\bar{g}_n, c_n)$ will satisfy the balance constraints with $\ell' = \ell - \epsilon$ and $L' = L + \epsilon$. For k-median we have*

$$\begin{aligned} Q^{(1)}(\bar{g}_n, c_n) \le r \cdot Q^{(1)}(f^*, c^*) + s + 2(r+1)pD\epsilon \\ + p(r+1)\alpha_1(S) + r\beta_1(S, \ell + \epsilon, L - \epsilon), \end{aligned}$$

and for k-means we have

$$\begin{aligned} Q^{(2)}(\bar{g}_n, c_n) \le 4r \cdot Q^{(2)}(f^*, c^*) + 2s + 4(r+1)pD^2\epsilon \\ + 2p(2r+1)\alpha_2(S) + 4r\beta_2(S, \ell + \epsilon, L - \epsilon). \end{aligned}$$

Proof. Lemma 37 guarantees that when the second sample is of size $O(\frac{1}{\epsilon^2}(n + \log \frac{1}{\delta}))$ then with probability at least $1 - \delta$, for any index set $I \subset [n]$, we have $\left|\sum_{i \in I} w_i - \hat{w}_i\right| \le \epsilon$. For the remainder of the proof, assume that this high probability event holds.

First we argue that the clustering (g_n, c_n) satisfies the true weighted capacity constraints with the slightly looser parameters $\ell' = \ell - \epsilon$ and $L' = L + \epsilon$. Since the clustering (g_n, c_n) satisfies the estimated weighted capacity constraints, the high probability event guarantees that it will also satisfy the true weighted capacity constraints with the looser parameters $\ell' = \ell - \epsilon$ and $L' = L + \epsilon$. Lemma 38 then guarantees that the extension $(\bar{g}_n, c_n)$ satisfies the population-level capacity constraints with parameters ℓ' and L'.

Next we bound the difference between the estimated and true weighted objectives for any clustering (g, c) of S. For each point x_j in the set S, let $C_j = \sum_{i \in g(x_j)} d(x_j, c(i))$ be the total distance from point x_j to its p assigned centers under clustering (g, c), and let J be the set of indices j for which $\hat{w}_j > w_j$. Then by the triangle inequality, we have the following bound for the k-median objective:

$$\begin{aligned}
&|\hat{Q}_n^{(1)}(g,c) - Q_n^{(1)}(g,c)| \\
&\leq \left|\sum_{j\in J}(\hat{w}_j - w_j)C_j\right| quad + \left|\sum_{j\notin J}(w_j - \hat{w}_j)C_j\right| \\
&\leq \left(\left|\sum_{j\in J}(\hat{w}_j - w_j)\right| + \left|\sum_{j\notin J}(w_j - \hat{w}_j)\right|\right)pD \\
&\leq 2pD\epsilon, \qquad (4)
\end{aligned}$$

where the second inequality follows from the fact that $C_j \leq pD$ and the sum has been split so that $(\hat{w}_j - w_j)$ is always positive in the first sum and negative in the second. For the k-means objective, the only difference is that our upper bound on C_j is D^2 instead of D, which gives $|\hat{Q}_n^{(2)}(g,c) - Q_n^{(2)}(g,c)| \leq 2pD^2\epsilon$.

Finally, let (h_n, c_n') be the clustering of S that attains the minimum in the definition of $\beta(S, \ell + \epsilon, L - \epsilon)$. That is, the clustering of S satisfying the capacity constraints with parameters $\ell + \epsilon$ and $L - \epsilon$ whose nearest neighbor extension has the best objective over $\mathcal{X}$ with respect to μ (note that this might not be the extension of (g_n^*, c_n^*)).

Now we turn to bounding the k-median objective value of $(\bar{g}_n, c_n)$ over the entire space $\mathcal{X}$. Combining Lemma 38, equation (4), the approximation guarantees for (g_n, c_n) with respect to $\hat{Q}_n$, and the optimality of (g_n^*, c_n^*), we have the following:

$$\begin{aligned}
Q^{(1)}(\bar{g}_n, c_n) &\leq Q_n^{(1)}(g_n, c_n) + p\alpha_1(S) \\
&\leq \hat{Q}_n^{(1)}(g_n, c_n) + 2pD\epsilon + p\alpha_1(S) \\
&= Q_n^{(1)}(g_n, c_n) - r \cdot \hat{Q}_n^{(1)}(g_n^*, c_n^*) \\
&\quad + r \cdot \hat{Q}_n^{(1)}(g_n^*, c_n^*) + 2pD\epsilon + p\alpha_1(S) \\
&\leq s + 2pD\epsilon + p\alpha_1(S) \\
&\quad + r \cdot \hat{Q}_n^{(1)}(h_n, c_n') \\
&\leq s + 2(r+1)pD\epsilon + p\alpha_1(S) + r \cdot Q_n^{(1)}(h_n, c_n') \\
&\leq s + 2(r+1)pD\epsilon + p(r+1)\alpha_1(S) \\
&\quad + r \cdot Q^{(1)}(\bar{h}_n, c_n') \\
&\leq s + 2(r+1)pD\epsilon + p(r+1)\alpha_1(S) \\
&\quad + r \cdot \beta_1(S, \ell + \epsilon, L - \epsilon) + r \cdot Q^{(1)}(f^*, c^*).
\end{aligned}$$

The proof for the case of k-means is identical, except we use the corresponding result from Lemma 38 and the alternative version of equation (4). □

Bounding $\alpha(S)$:

In this section, we prove the following Lemma bounding the α term from Theorem 39:

Lemma 40. *For any $\epsilon, \delta > 0$, and $\mathcal{X} \subseteq \mathbb{R}^q$, if a randomly drawn S from μ is of size $O(q^{q/2}\epsilon^{-(q+1)}(q\log\frac{\sqrt{q}}{\epsilon} + \log\frac{1}{\delta}))$ in the general case, or $O(\epsilon^{-d_0}(d_0\log\frac{1}{\epsilon} + \log\frac{1}{\delta}))$ if μ is doubling with dimension d_0, then w.p $\geq 1 - \delta$, $\alpha_1(S) \leq \epsilon D$ and $\alpha_2(S) \leq (\epsilon D)^2$.*

First we show that when the set $\mathcal{X}$ is bounded in $\mathbb{R}^q$, then for a large enough sample S drawn from μ, every point $x \in \mathcal{X}$ will have a close neighbor uniformly with high probability.

Lemma 41. *For any $r > 0$ and any $\epsilon > 0$, there exists a subset $\mathcal{Y}$ of $\mathcal{X}$ containing at least $1 - \epsilon$ of the probability mass of μ such that, for any $\delta > 0$, if we see an iid sample S of size $n = O(\frac{1}{\epsilon}(\frac{D\sqrt{q}}{r})^q(q\log\frac{D\sqrt{q}}{r} + \log\frac{1}{\delta}))$ drawn from μ, then with probability at least $1 - \delta$ we have $\sup_{x\in\mathcal{Y}} d(x, \mathrm{NN}_S(x)) \leq r$.*

Proof. Let C be the smallest cube containing the support $\mathcal{X}$. Since the diameter of $\mathcal{X}$ is D, the side length of C is at most D. Let $s = r/\sqrt{q}$ be the side-length of a cube in $\mathbb{R}^q$ that has diameter r. Then it takes at most $m = \lceil D/s \rceil^q$ cubes of side-length s to cover the set C. Let $C_1, \ldots, C_m$ be such a covering of C, where each C_i has side length s.

Let C_i be any cube in the cover that has probability mass at least ϵ/m under the distribution μ. The probability that a sample of size S drawn from μ does not contain a sample in C_i is at most $(1 - \epsilon/m)^n$. Let I denote the index set of all those cubes with probability mass at least ϵ/m under μ. Applying the union bound over the cubes indexed by I, the probability that there exists a cube C_i with $i \in I$ that does not contain any sample from S is at most $m(1 -$

$\epsilon/m)^n \leq me^{-n\epsilon/m}$. Setting $n = \frac{m}{\epsilon}(\ln m + \log\frac{1}{\delta}) = O(\frac{1}{\epsilon}(\frac{D\sqrt{q}}{r})^q(q\log\frac{D\sqrt{q}}{r}+\log\frac{1}{\delta}))$ results in this upper bound being δ. For the remainder of the proof, suppose that this high probability event occurs.

Define $\mathcal{Y} = \bigcup_{i\in I} C_i$. Each cube from our cover not used in the construction of $\mathcal{Y}$ has probability mass at most ϵ/m and, since there are at most m such cubes, their total mass is at most ϵ. It follows that $\mathbb{P}_{x\sim\mu}(x \in \mathcal{Y}) \geq 1-\epsilon$. Moreover, every point x in $\mathcal{Y}$ belongs to one of the cubes, and every cube C_i with $i \in I$ contains at least one sample point. Since the diameter of the cubes is r, it follows that the nearest sample to x is at most r away.

Setting $r = D\epsilon$, we obtain one half of Lemma 40. □

For the remainder of this section, suppose that μ is a doubling measure of dimension d_0 with support $\mathcal{X}$ and that the diameter of $\mathcal{X}$ is $D > 0$. First, we shall prove general lemmas about doubling measures. They are quite standard, and are included here for the sake of completion. See, for example, (Krauthgamer and Lee, 2004; Kpotufe, 2010).

Lemma 42. *For any $x \in \mathcal{X}$ and any radius of the form $r = 2^{-T}D$ for some $T \in \mathbb{N}$, we have*

$$\mu(B(x,r)) \geq (r/D)^{d_0}.$$

Proof. Since $\mathcal{X}$ has diameter D, for any point $x \in \mathcal{X}$ we have that $\mathcal{X} \subset B(x,D)$, which implies that $\mu(B(x,D)) = 1$. Applying the doubling condition T times gives $\mu(B(x,r)) = \mu(B(x,2^{-T}D)) \geq 2^{-Td_0} = (r/D)^{d_0}$. □

Lemma 43. *For any radius of the form $r = 2^{-T}D$ for some $T \in \mathbb{N}$, there is a covering of $\mathcal{X}$ using balls of radius r of size no more than $(2D/r)^{d_0}$.*

Proof. Consider the following greedy procedure for covering $\mathcal{X}$ with balls of radius r: while there exists a point $x \in \mathcal{X}$ that is not covered by our current set of balls of radius r, add the ball $B(x,r)$ to the cover. Let C denote the set of centers for the balls in our cover. When this procedure terminates, every point in $\mathcal{X}$ will be covered by some ball in the cover.

We now show that this procedure terminates after adding at most $(2D/r)^{d_0}$ balls to the cover. By construction, no ball in our cover contains the center of any other, implying that the centers are at least distance r from one another. Therefore, the collection of balls $B(x,r/2)$ for $x \in C$ are pairwise disjoint. Lemma 42 tells us that $\mu(B(x,r/2)) \geq (r/2D)^{d_0}$, which gives that $1 \geq \mu\Big(\bigcup_{x\in C} B(x,r/2)\Big) = \sum_{x\in C}\mu(B(x,r/2)) \geq |C|(r/2D)^{d_0}$. Rearranging the above inequality gives $|C| \leq (2D/r)^{d_0}$. □

The next lemma tells us that we need a sample of size $O\big((\frac{D}{r})^{d_0}(d_0\log\frac{D}{r}+\log\frac{1}{\delta})\big)$ in order to ensure that there is a neighbor from the sample no more than r away from any point in the support with high probability. The second half of Lemma 40 is an easy corollary.

Lemma 44. *For any $r > 0$ and any $\delta > 0$, if we draw an iid sample S of size $n = (\frac{2D}{r})^{d_0}(d_0\log(\frac{4D}{r}) + \log(\frac{1}{\delta}))$, then with probability at least $1-\delta$ we have $\sup_{x\in\mathcal{X}} d(x, \mathrm{NN}_S(x)) \leq r$*

Proof. By Lemma 43 there is a covering of $\mathcal{X}$ with balls of radius $r/2$ of size $(4D/r)^{d_0}$. For each ball B in the cover, the probability that no sample point lands in B is $(1-\mu(B))^n \leq (1-(r/2D)^{d_0})^n \leq \exp(-n(r/2D)^{d_0})$. Let E be the event that there exists at least one ball B in our cover that does not contain one of the n sample points. Applying the union bound over the balls in the cover, we have that $\mathbb{P}(E) \leq (4D/r)^{d_0}\exp(-n(r/2D)^{d_0})$. Setting $n = (2D/r)^{d_0}(d_0\log(4D/r) + \log(1/\delta)) = O\big((\frac{D}{r})^{d_0}(d_0\log\frac{D}{r}+\log\frac{1}{\delta})\big)$, we have that $\mathbb{P}(E) < \delta$. When the bad event E does not occur, every ball in our covering contains at least one sample point. Since every point $x \in \mathcal{X}$ belongs to at least one ball in our covering and each ball has diameter r, we have $\sup_{x\in\mathcal{X}} d(x, \mathrm{NN}_S(x)) \leq r$. □

Bounding $\beta(S)$: In this section, we prove the following bound on β:

Lemma 45. *Let μ be a measure on $\mathbb{R}^q$ with support $\mathcal{X}$ of diameter D. Let f^*, some clustering of μ that satisfies capacities $(\ell+\epsilon, L-\epsilon)$, be ϕ-PL. If we see a sample S drawn iid from μ of size $O\Big(\frac{1}{\epsilon}\big(\frac{1}{\phi^{-1}(\epsilon/2)}\big)^q\big(q\log\frac{\sqrt{q}}{\phi^{-1}(\epsilon/2)}+\log\frac{1}{\delta}\big)\Big)$ in the general case or $O\Big(\big(\frac{1}{\phi^{-1}(\epsilon)}\big)^{d_0}\big(d_0\log\frac{4}{\phi^{-1}(\epsilon)}+\log\frac{1}{\delta}\big)\Big)$ when μ is a doubling measure of dimension d_0 then, w.p. at least $1-\delta$ over the draw of S, we have that $\beta_1(S,\ell,L) \leq pD\epsilon$ and $\beta_2(S,\ell,L) \leq pD^2\epsilon$.*

First, we define the Probabilistic Lipschitzness condition:

Definition 5 (Probabilistic Lipschitzness)**.** *Let $(\mathcal{X}, d())$ be some metric space of diameter D and let $\phi : [0,1] \to [0,1]$. $f : \mathcal{X} \to [k]$ is ϕ-Lipschitz with respect to some distribution μ over $\mathcal{X}$, if $\forall\lambda \in [0,1]$: $\mathbb{P}_{x\sim\mu}\big[\exists y : \mathbb{I}\{f(x) \neq f(y)\}$ and $d(x,y) \leq \lambda D\big] \leq \phi(\lambda)$*

Proof of Lemma 45. Suppose $\mathbb{P}_{x\sim\mu}(f^*(\mathrm{NN}_S(x)) \neq f^*(x)) \leq \epsilon$.

Define the restriction $f_S : S \to \binom{k}{p}$ of $f \in F(\ell, L)$ to be $f_S(x) = f(x)$ for $x \in S$. Firstly, we shall show that the cluster sizes of f^*_S can be bounded. Recall that the sizes of cluster i in a clustering f of $\mathcal{X}$ and a clustering g of the sample S are respectively is $\mathbb{P}_{x\sim\mu}(i \in f(x))$ and $\mathbb{P}_{x\sim\mu}(i \in \bar{g}(x))$. By the triangle inequality, $|\mathbb{P}_{x\sim\mu}(i \in \bar{f^*_S}(x)) - \mathbb{P}_{x\sim\mu}(i \in f^*(x))| \leq \mathbb{P}_{x\sim\mu}(\bar{f^*_S}(x) \neq f^*(x)) =$

$\mathbb{P}_{x\sim\mu}(f^*(\mathrm{NN}_S(x)) \neq f^*(x))$ and this is at most ϵ, by our assumption.

Consider $\beta(S)$. Since $f^* \in F(\ell+2\epsilon, L-2\epsilon)$, we have that $f^*_S \in G_n(\ell-\epsilon, L+\epsilon)$, we have

$$\beta(S) \leq Q(\bar{f}^*_S, c^*) - Q(f^*, c^*)$$
$$= \mathop{\mathbb{E}}_{x\sim\mu}\left[\sum_{i\in f^*(\mathrm{NN}_S(x))} \|x-c(i)\| - \sum_{i'\in f^*(x)} \|x-c(i')\|\right]$$

By the triangle inequality, $\|x-c(i)\|-\|x-c(i')\| \leq \|c(i)-c(i')\|$. Since $f^*(x)$ and $f^*_S(\mathrm{NN}_S(x))$ can differ on at most p assignments, and since any two centers are most a distance D apart, we have that $\beta(S) \leq \mathbb{E}_{x\sim\mu}(pD\cdot\mathbb{I}\{f^*(\mathrm{NN}_S(x)\} \neq f^*(x))) = pD\cdot\mathbb{P}_{x\sim\mu}(f^*(\mathrm{NN}_S(x)) \neq f^*(x)) \leq pD\epsilon$.

All that remains is to show that $\mathbb{P}_{x\sim\mu}(f^*(\mathrm{NN}_S(x)) \neq f^*(x)) \leq \epsilon$ for big enough n. Lemma 46 lists the conditions when this is true. □

We require the following lemma for nearest neighbor classification, similar in spirit to that of Urner et al (Urner et al., 2013). Note that since f is a set of p elements, this lemma holds for multi-label nearest neighbor classification.

Lemma 46. *Let μ be a measure on $\mathbb{R}^q$ with support $\mathcal{X}$ of diameter D. Let the labeling function, f be ϕ-PL. For any accuracy parameter ϵ and confidence parameter δ, if we see a sample S of size at least*

- $\frac{2}{\epsilon}\left\lceil\frac{\sqrt{q}}{\phi^{-1}(\epsilon/2))}\right\rceil^q\left(q\log\lceil\frac{\sqrt{q}}{\phi^{-1}(\epsilon/2)}\rceil+\log\frac{1}{\delta}\right)$ *in the general case*
- $\left(\frac{2}{\phi^{-1}(\epsilon)}\right)^{d_0}\left(d_0\log\frac{4}{\phi^{-1}(\epsilon)}+\log\frac{1}{\delta}\right)$ *when μ is a doubling measure of dimension d_0*

then nearest neighbor classification generalizes well. That is, with probability at least $1-\delta$ over the draw of S, the error on a randomly drawn test point, $\mathbb{P}_{x\sim\mu}(f(x) \neq f(\mathrm{NN}_S(x))) \leq \epsilon$.

Proof. Let $\lambda = \phi^{-1}(\epsilon)$. We know that most of $\mathcal{X}$ can be covered using hypercubes in the general case, as in Lemma 41 or entirely covered using balls in the case when μ is a doubling measure, as in Lemma 43, both of diameter λD. In case we have cubes in the cover, we shall use a ball of the same diameter instead. This does not change the sample complexity, since a cube is completely contained in a ball of the same diameter.

Formally, let $\mathcal{C}$ be the covering obtained from Lemma 41 or Lemma 43, depending on whether or not the measure is a doubling measure. Define $\mathcal{B}(x)$ to be the set of all the balls from $\mathcal{C}$ that contain the point x. A point will only be labeled wrongly if it falls into a ball with no point from S, or a ball that contains points of other labels. Hence,

$$\mathop{\mathbb{P}}_{x\sim\mu}(f(\mathrm{NN}_S(x)) \neq f(x)) \leq \mathop{\mathbb{P}}_{x\sim\mu}(\forall C\in\mathcal{B}(x): S\cap C = \emptyset)$$
$$+ \mathop{\mathbb{P}}_{x\sim\mu}(\exists y \in \bigcup_{C\in\mathcal{B}(x)} C : f(y) \neq f(x))$$

Since each ball is of diameter λD, the second term is at most $\mathbb{P}_{x\sim\mu}(\exists y \in B(x,\lambda D) : f(y) \neq f(x))$. By the PL assumption, this is at most $\phi(\lambda) = \epsilon$, independent of the covering used.

For the first term, our analysis will depend on which covering we use:

- From Lemma 41, we know that all but $1-\epsilon$ fraction of the space is covered by the covering $\mathcal{C}$. When the sample is of size $O(\frac{1}{\epsilon}(\frac{\sqrt{q}}{\lambda})^q(q\log(\frac{\sqrt{q}}{\lambda})+\log\frac{1}{\delta}))$, each $C\in\mathcal{C}$ sees a sample point. For a sample this large, the first term is $\leq\epsilon$. Substituting ϵ with $\epsilon/2$ completes this part of the proof.
- When μ is a doubling measure, we can do better.If every ball of the cover sees a sample point, the first term is necessarily zero. From the proof of Lemma 44, we know that if we draw a sample of size $n = (2/\lambda)^{d_0}(d_0\log(4/\lambda)+\log(1/\delta))$ samples, then every ball of the cover sees a sample point with probability at least $1-\delta$ over the draw of S. This completes the proof.

□

13 DETAILS FOR THE EXPERIMENTS

Experimental System Setup: We now describe the distributed implementation used for the experiments. We start one worker process on each of the available processing cores. First, a single worker subsamples the data, clusters the subsample into k clusters, and then builds a random partition tree for fast nearest neighbor lookup. The subsample, clustering, and random partition tree describe a dispatching rule, which is then copied to every worker. Training the system has two steps: first, the training data is dispatched to the appropriate workers, and then each worker learns a model for the clusters they are responsible for. During the deployment phase, the workers load the training data in parallel and send each example to the appropriate workers (as dictated by the dispatch rule). To minimize network overhead examples are only sent over the network in batches of 10,000. During the training phase, each worker calls either Liblinear or an L-BFGS solver to learn a one-vs-all linear classifier for each of their clusters. For testing, the testing data is loaded in parallel by the workers and the appropriate workers are queried for predictions.

Details of LSH-baseline: The LSH family used by our LSH baseline is the concatenation of t random projections followed by binning. Each random projection is constructed by sampling a direction u from the standard Gaussian distribution in $\mathbb{R}^d$. An example x is then mapped to the index $\lfloor u^\top x/w \rfloor$, where w is a scale parameter. Two points x and y map to the same bin if they agree under all t hash functions. In our experiments, the parameter t was set to 10 and w is chosen so that hashing the training data results in approximately $2k$ non-empty bins. We tried several other values of t and w but performance did not change significantly.

Details for Synthetic Data Distribution: The synthetic distribution used in Section 5 is an equal-weight mixture of 200 Gaussians in $\mathbb{R}^{20}$ with means chosen uniformly at random from the unit cube $[0, 1]^{20}$. Each Gaussian is associated with one of 30 class labels. To decide the class labels, we construct a hierarchical clustering of the Gaussian centers using complete linkage and assign labels using a simple recursive procedure: each internal node of the subtree is associated with a range of permissible labels. The permissible labels assigned to the left and right children of the parent node are a partition of the parent's assigned labels, and the number of labels they receive is proportional to the number of leaves in the subtree. If a subtree has only one permissible label, both children are given that label. Finally, each leave chooses a label uniformly at random from the set of permissible labels it was assigned (in many cases, there will only be one). This labeling strategy results in nearby Gaussians having similar labels.

Inception Network: The specific architecture for the nerual network we used when constructing the feature representations for the CIFAR-10 dataset can be found here: `https://github.com/dmlc/mxnet/blob/master/example\/notebooks/cifar10-recipe.ipynb`.

Hardware: The experiments for MNIST-8M, CIFAR10, and the CTR datasets were performed on a cluster of 15 machines, each with 8 Intel(R) Xeon(R) cores of clock rate 2.40 GHz and 32GB shared memory per machine. The experiments for the large synthetic experiment were performed on AWS. We used clusters of 8, 16, 32, and 64 m3.large EC2 instances, each with an Intel (R) Xeon E5-2670 v2 processors and 7.5 GB memory per machine.

Clustering algorithm selection: In Section 3 we showed that k-means++ will find high-quality balanced clusterings of the data whenever a natural stability condition is satisfied. Since the k-means++ algorithm is simple and scalable, it is a good candidate for implementation in real systems. In this section we present an empirical study of the quality of the clusterings produced by k-means++ for the datasets used in our experiments. For each of the datasets used in our learning experiments, we find that the clustering obtained by k-means++ is very competitive with the LP rounding techniques. We also include a synthetic dataset designed specifically so that k-means++ with balancing heuristics will not perform as well as the LP-rounding algorithms.

We compare the clustering algorithms on two metrics: (1) the k-means objective value of the resulting clustering and (2) the mean per-cluster class distribution entropy. Since the LP rounding algorithms may violate the replication by a factor of 2, we use an averaged version of the k-means objective

$$Q(f, c) = \sum_{i=1}^{n} \frac{1}{|f(x_i)|} \sum_{j \in f(x_i)} \|x - c_j\|^2,$$

which provides a fair comparison when $|f(x_i)|$ is not exactly p for all points. The second metric is the mean per-cluster class distribution entropy which measures how well the clusters are capturing information about the class labels. Each cluster has an empirical distribution over the class labels and this distribution will have low entropy when the cluster contains mostly examples from a small number of classes. Therefore, when the average class-distribution entropy per cluster is small, we expect the learning problems for each cluster to be simpler than the global learning problem, which should lead to improved accuracy. Formally, given a dataset $(x_1, y_1), \ldots, (x_n, y_n)$ with $y_i \in \{1, \ldots, M\}$ and a clustering (f, c), we compute

$$H(f, c) = -\frac{1}{k} \sum_{j=1}^{k} \sum_{y=1}^{M} p_{j,y} \log_2(p_{j,y}),$$

where $p_{j,y}$ is the fraction of the points in cluster j that belong to class y.

Results: We run the k-means++ algorithm with balancing heuristics described in Section 5, the LP-rounding algorithm for the k-means objective, and the LP-rounding algorithm for the k-median objective. For each dataset, we randomly subsample 700 points and run the algorithm for values of k in $\{8, 16, 32, 64, 128\}$ with $p = 2$ and compute the above metrics. This is averaged over 5 runs. The k-means objective values are shown in Figure 8 and the mean per-cluster class entropies are shown in Figure 9.

For every dataset, the k-means++ algorithm finds a clustering of the data that has better k-means objective value than the LP-rounding algorithms, and for all but the large values of k, the per-cluster class entropies are also smaller for k-means++. It is interesting to note that the LP-rounding algorithm for k-median achieves *better* k-means objective that the LP-rounding algorithm for k-means! This might be explained by the smaller approximation factor for k-median. The balancing heuristics for k-means++ never resulted in the algorithm outputting more than k clusters (though for $k = 128$, it output on average 16 too few clusters). Finally, the LP-rounding algorithms assigned the majority of points to only 1 center (which is allowed under our bi-criteria analysis). This reduces the cluster sizes, which explains why

the per-cluster class entropy is lower for the LP rounding algorithms when k is large. These results justify the use of k-means++ in our distributed learning experiments.

Additional Synthetic Distribution for Bounded Partition Trees: In this section we present an additional synthetic distribution for which our algorithm significantly outperforms the balanced partition tree. The data distribution is uniform on the 100-dimensional rectangle $[0, 10] \times [0, 10] \times [0, 1] \times \cdots \times [0, 1]$, where the first two dimensions have side length 10 and the rest have side length 1. The class of an example depends only on the first 2 coordinates, which are divided in to a regular 4×4 grid with one class for each grid cell, giving a total of 16 classes. Figure 10 shows a sample from this distribution projected onto the first two dimensions. We use either balanced partition trees or our algorithm using k-means++ to partition the data, and then we learn a linear one-vs-all SVM on each subset. If a subset is small enough to only intersect with one or two grid cells, then the learning problem is easy. If the subset intersects with many grid cells, there is not usually a low-error one-vs-all linear classifier.

Intuitively, balanced partition trees fail on this dataset because in order for them to produce a good partitioning, they need to repeatedly split on one of the first two dimensions. Any time the balanced partition tree splits on another dimension, the two resulting learning problems are identical but with half the data. On the other hand, clustering-based approaches will naturally divide the data into small groups, which leads to easier learning problems. The accuracies for the balanced partition trees and k-means++ are shown in Figure 11. Our method is run with parameters $p = 1$, $\ell = 1/(2k)$, and $L = 2/k$.

14 COMPARISON OF CLUSTERING ALGORITHMS

In this section, we empirically and theoretically compare the LP rounding and k-means++ algorithms on a data distribution designed specifically to show that in some cases, the LP rounding algorithm gives higher performance.

The synthetic distribution is a mixture of two Gaussians in $\mathbb{R}^2$, one centered at $(0, 0)$ and the other centered at $(10, 0)$. We set the balancing constraints to be $\ell = 1/10$ and $L = 1$ so that no cluster can contain fewer than 10% of the data. The mixing coefficient for the Gaussian at $(10, 0)$ is set to $0.8\ell = 0.08$, so in a sample this Gaussian will not contain enough data points to form a cluster on its own. In an optimal 2-clustering of this data with the given constraints, cluster centered at $(10, 0)$ will steal some additional points from the Gaussian centered at $(0, 0)$. Running the k-means++ algorithm, however, will produce a clustering that does not satisfy the capacity constraints, and the merging heuristic described in Section 5 will simply merge the points into one cluster. The following labeling function is designed so that there is no globally accurate one-vs-all linear classifier, but for which there is an accurate classifier for each cluster in the optimal clustering.

$$f(x) = \begin{cases} -1 & \text{if } x_1 \leq 0 \\ 1 & \text{if } x_1 \in (0, 5] \\ 2 & \text{otherwise.} \end{cases}$$

The LP rounding algorithm requires the replication parameter p to be at least two, so we run both algorithms with $p = 2$ and $k = 4$, in which case the above intuitions still hold but now each of the clusters is assigned two centers instead of one. Figure 12 shows a sample drawn from this distribution labeled according to the above target function.

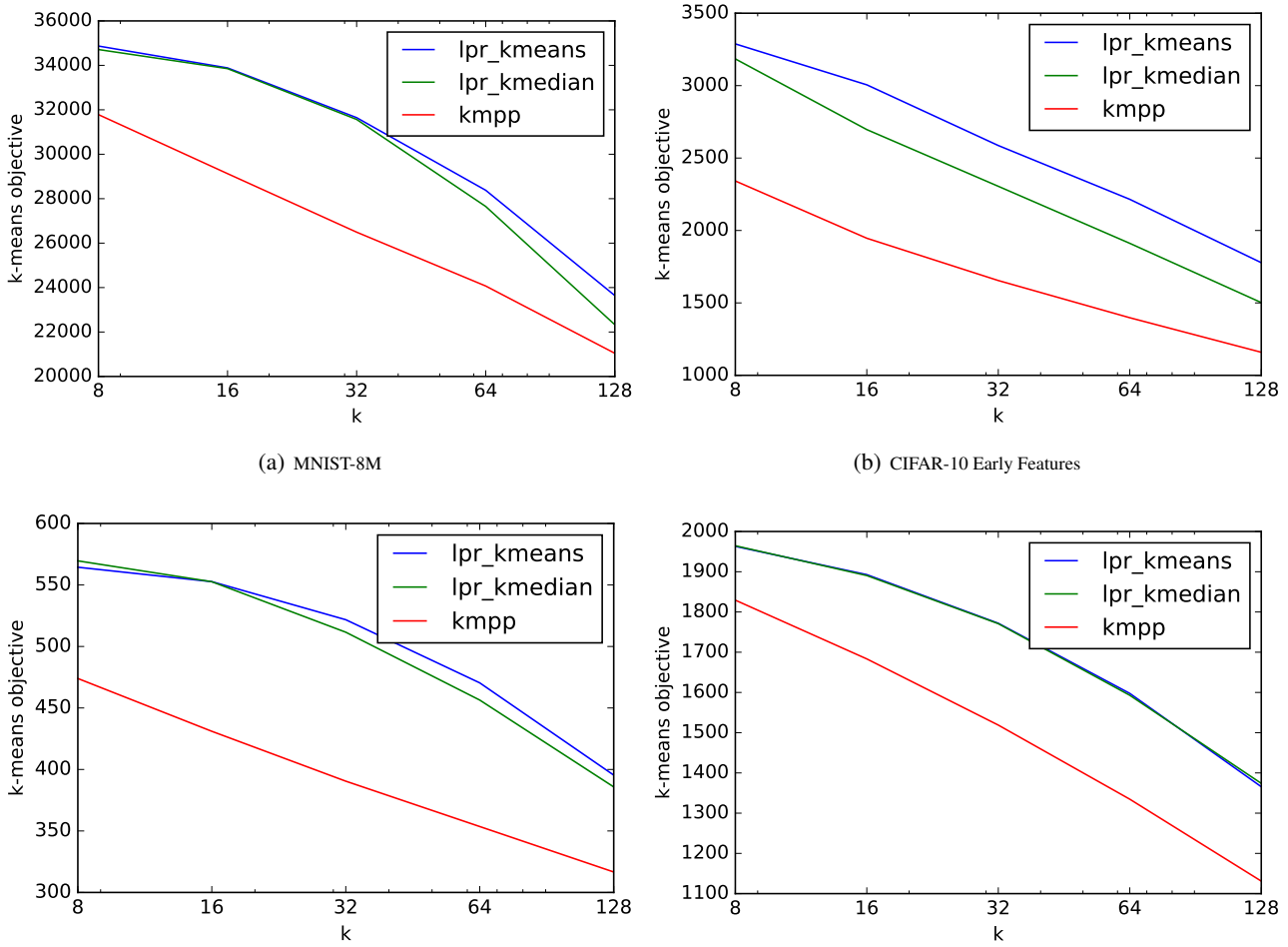
We evaluate the LP rounding algorithm, k-means++, and an algorithm that optimally solves the clustering problem by solving the corresponding integer program (this is only feasible for very small input sizes). Other aspects of the experiment are identical to the large scale learning experiments described in Section 5. In all cases, we set the parameters to be $k = 4$, $p = 2$. The training size is 10,000, the testing size 1,000, and the clustering is performed on a sample of size 200. Running k-means++ results in accuracy **0.768** (averaged over 500 runs), using the LP rounding algorithm results in accuracy **0.988**, and exactly solving the capacitated clustering IP results in accuracy **0.988**. Since the LP rounding and IP based algorithms are deterministic, we did not run the experiment multiple times. The accuracy of k-means++ does not significantly increase even if we cluster the entire sample of training 10,000 points rather than a small sample of size 200. This experiment shows that, while the k-means++ heuristic works effectively for many real-world datasets, it is possible to construct datasets where its performance is lower than the LP rounding algorithm.

With a modification to this distribution, but with the same intuition, we can prove there is a point set in which the LP rounding algorithm outperforms k-means++.

Lemma 47. *There exists a distribution such that with constant probability, classification using the k-median LP rounding algorithm as a dispatcher outperforms the k-means++ algorithm with a balancing heuristic.*

Proof. The point set is as follows. All points are on a line, and there are three groups of points, group A, B, and C. Two points in the same group are at distance zero. Group A is distance 1 from group B, and group B is distance 10. Group A is distance 101 from group C. Group A contains 112 points, group B contains 111 points, and group C contains 1 point. Set $k = 2$, $n\ell = 112$, and $L = 1$. For now, $p = 1$. All points in A, B, and C are labeled -1, 0, and 1, respectively.

Then the optimal k-median 2-means cluster is to put centers at A and B. Then the points at A and B pay zero, and the



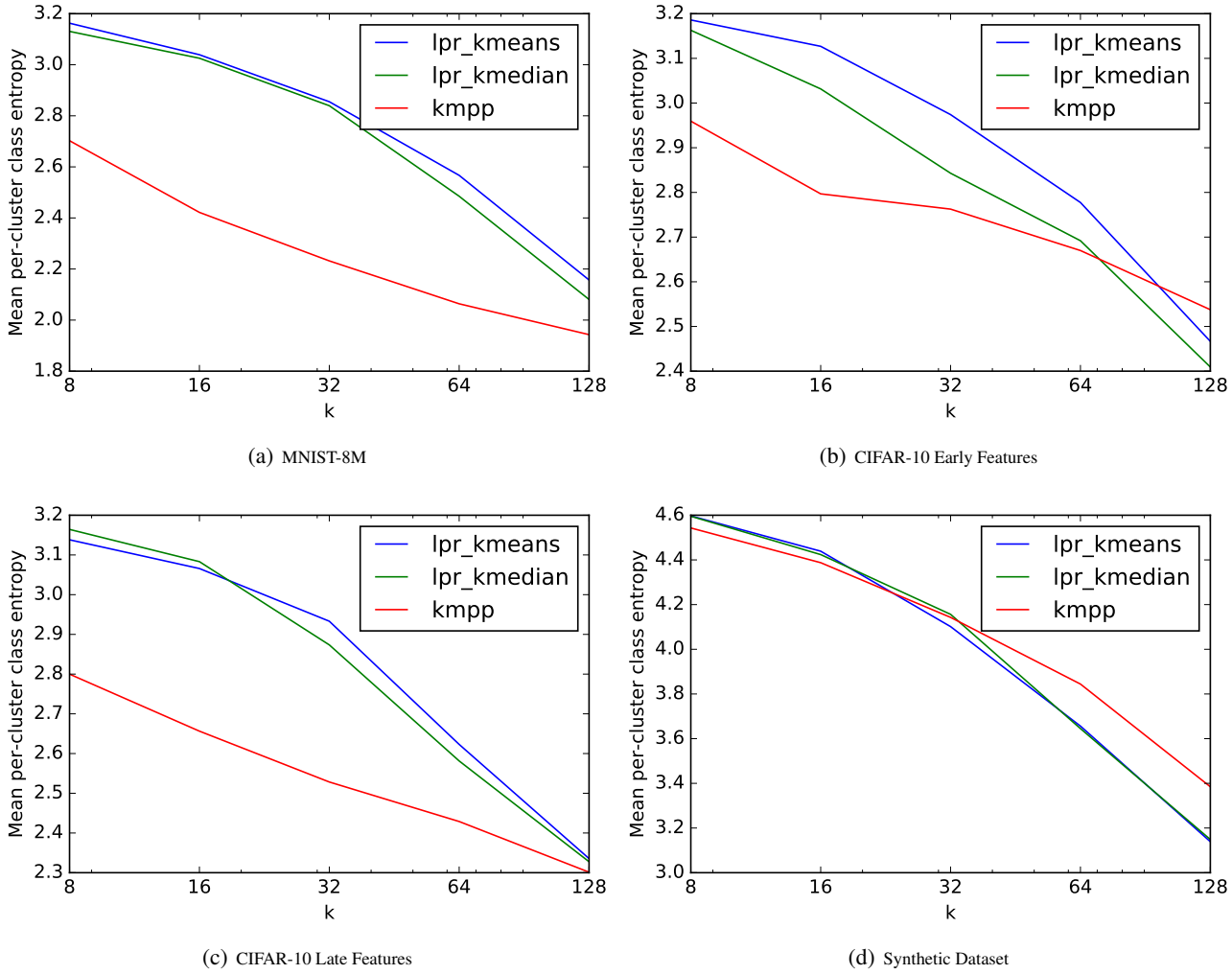
(a) MNIST-8M

(b) CIFAR-10 Early Features

(c) CIFAR-10 Late Features

(d) Synthetic Dataset

Figure 8: Comparison of the k-means objective value.



(a) MNIST-8M

(b) CIFAR-10 Early Features

(c) CIFAR-10 Late Features

(d) Synthetic Dataset

Figure 9: Comparison of the mean per-cluster class distribution entropy.

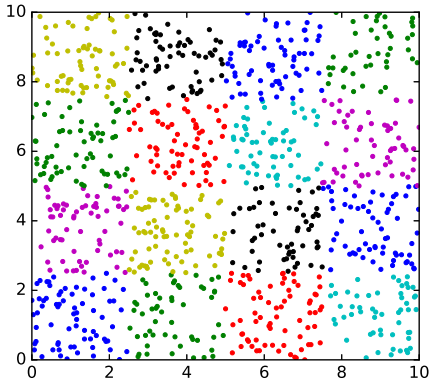


Figure 10: Scatter plot of the data after projecting onto the first two coordinates (note: some colors are reused)

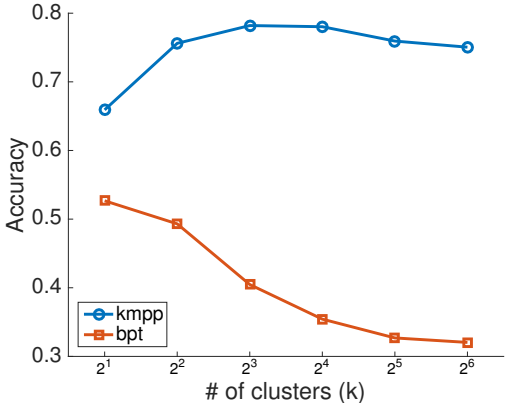


Figure 11: Accuracy of learning using balanced partition trees and k-means++

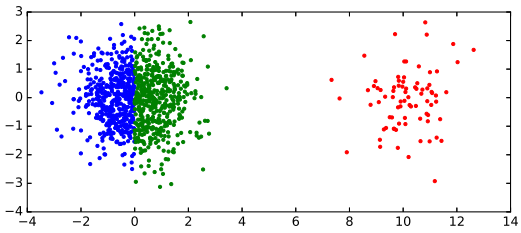


Figure 12: A dataset which has two well defined clusters that are not balanced in size.

points at C pay $10 \cdot 10 = 100$ to connect to the center at B. So the total k-median score is 100.

The LP rounding algorithm is guaranteed to output an 11-approximation, so it must output a 2-clustering with score ≤ 110 (it only works when $p \geq 2$, but we will modify the proof for $p \geq 1$ at the end). Note, the centers must stay at A and B, because if a center is not at (wlog) A, then the center must be distance at least 1 away from A, which means the score of this clustering is ≥ 111. Now we know the LP algorithm is guaranteed to output centers at A and B. Then the clusters must be A and $B \cup C$, because this assignment minimizes the flow in the graph the LP algorithm uses to assign points to clusters. Therefore, each cluster has ≤ 2 labels, which can easily be classified using a linear separator.

Now we consider the k-means++ algorithm. First we calculate the probability that a center is placed in group C. Note, there is zero probability that both centers are in the same group. So this probability is the complement of the probability the centers fall in A and B. By a simple calculation, the probability is $1 - \frac{112}{224} \cdot \frac{1 \cdot 111}{1 \cdot 111 + 11 \cdot 1} - \frac{111}{224} \cdot \frac{1 \cdot 112}{1 \cdot 112 + 10 \cdot 1} = .09016$. However, if a center is placed in group C, then the clusters will be $A \cup B$ and C, which means the k-means++ balancing heuristic will combine both clusters into a single cluster. Then, there are 3 groups of points with different labels on a line, so a linear separator must classify at least one point incorrectly. □

The proof of Lemma 47 can be modified for k-means. The probability that k-means outputs a bad clustering is inversely proportional to the approximation factor of the LP algorithm, but since the LP algorithm has constant-factor approximation ratios, the probability is constant. The proof can also be modified for $p = 2$ similar to the explanation in the experimental evaluation. Set $p = 2$ and $k = 4$, and the problem stays the same, since the optimal clustering puts two centers at A and two centers at B.