Distributed k-Means and k-Median Clustering on General Topologies

Maria Florina Balcan* Steven Ehrlich† Yingyu Liang‡

Abstract

This paper provides new algorithms for distributed clustering for two popular center-based objectives, k-median and k-means. These algorithms have provable guarantees and improve communication complexity over existing approaches. Following a classic approach in clustering by [18], we reduce the problem of finding a clustering with low cost to the problem of finding a coreset of small size. We provide a distributed method for constructing a global coreset which improves over the previous methods by reducing the communication complexity, and which works over general communication topologies. Experimental results on large scale data sets show that this approach outperforms other coreset-based distributed clustering algorithms.

1 Introduction

Most classic clustering algorithms are designed for the centralized setting, but in recent years data has become distributed over different locations, such as distributed databases [27, 9], images and videos over networks [26], surveillance [15] and sensor networks [8, 16]. In many of these applications the data is inherently distributed because, as in sensor networks, it is collected at different sites. As a consequence it has become crucial to develop clustering algorithms which are effective in the distributed setting.

Several algorithms for distributed clustering have been proposed and empirically tested. Some of these algorithms [14, 28, 10] are direct adaptations of centralized algorithms which rely on statistics that are easy to compute in a distributed manner. Other algorithms [19, 22] generate summaries of local data and transmit them to a central coordinator which then performs the clustering algorithm. No theoretical guarantees are provided for the clustering quality in these algorithms, and they do not try to minimize the communication cost. Additionally, most of these algorithms assume that the distributed nodes can communicate with all other sites or that there is a central coordinator that communicates with all other sites.

In this paper, we study the problem of distributed clustering where the data is distributed across nodes whose communication is restricted to the edges of an arbitrary graph. We provide algorithms with small communication cost and provable guarantees on the clustering quality. Our technique for reducing communication in general graphs is based on the construction of a small set of points which act as a proxy for the entire data set.

An *ϵ-coreset* is a weighted set of points whose cost on any set of centers is approximately the cost of the original data on those same centers up to accuracy ϵ. Thus an approximate solution for the coreset is also an approximate solution for the original data. Coresets have previously been studied in the centralized setting

*Georgia Institute of Technology, ninamf@cc.gatech.edu
†Georgia Institute of Technology, sehrlich@cc.gatech.edu
‡Georgia Institute of Technology, yliang39@gatech.edu

([18, 12]) but have also recently been used for distributed clustering as in [29] and as implied by [13]. In this work, we propose a distributed algorithm for k-means and k-median, by which each node constructs a local portion of a global coreset. Communicating the approximate cost of a global solution to each node is enough for the local construction, leading to low communication cost overall. The nodes then share the local portions of the coreset, which can be done efficiently in general graphs using a message passing approach.

More precisely, in Section 3, we propose a distributed coreset construction algorithm based on local approximate solutions. Each node computes an approximate solution for its local data, and then constructs the local portion of a coreset using only its local data and the total cost of each node's approximation. For ϵ constant, this builds a coreset of size $\tilde{O}(kd + nk)$ for k-median and k-means when the data lies in d dimensions and is distributed over n sites [1]. If there is a central coordinator among the n sites, then clustering can be performed on the coordinator by collecting the local portions of the coreset with a communication cost equal to the coreset size $\tilde{O}(kd + nk)$. For distributed clustering over general connected topologies, we propose an algorithm based on the distributed coreset construction and a message-passing approach, whose communication cost improves over previous coreset-based algorithms. We provide a detailed comparison below.

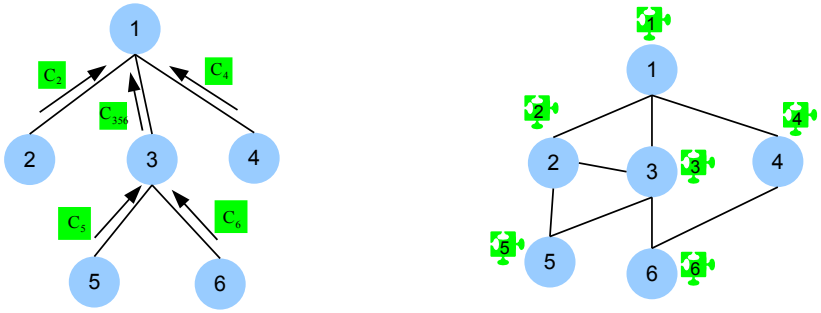
Experimental results on large scale data sets show that our algorithm performs well in practice. For a fixed amount of communication, our algorithm outperforms other coreset construction algorithms.

Comparison to Other Coreset Algorithms: Since coresets summarize local information they are a natural tool to use when trying to reduce communication complexity. If each node constructs an ϵ-coreset on its local data, then the union of these coresets is clearly an ϵ-coreset for the entire data set. Unfortunately the size of the coreset in this approach increases greatly with the number of nodes.

Another approach is the one presented in [29]. Its main idea is to approximate the union of local coresets with another coreset. They assume nodes communicate over a rooted tree, with each node passing its coreset to its parent. Because the approximation factor of the constructed coreset depends on the quality of its component coresets, the accuracy a coreset needs (and thus the overall communication complexity) scales with the height of this tree. Although it is possible to find a spanning tree in any communication network, when the graph has large diameter every tree has large height. In particular many natural networks such as grid networks have a large diameter ($\Omega(\sqrt{n})$ for grids) which greatly increases the size of coresets which must be communicated across the lower levels of the tree. We show that it is possible to construct a global coreset with low communication overhead. This is done by distributing the coreset construction procedure rather than combining local coresets. The communication needed to construct this coreset is negligible – just a single value from each data set representing the approximate cost of their local optimal clustering. Since the sampled global ϵ-coreset is the same size as any local ϵ-coreset, this leads to an improvement of the communication cost over the other approaches. See Figure 1 for an illustration. The constructed coreset is smaller by a factor of n in general graphs, and is independent of the communication topology. This method excels in sparse networks with large diameters, where the previous approach in [29] requires coresets that are quadratic in the size of the diameter for k-median and quartic for k-means; see Section 4 for details. [13] also merge coresets using coreset construction, but they do so in a model of parallel computation and ignore communication costs.

Balcan et al. [5] and Daume et al. [11] consider communication complexity questions arising when doing classification in distributed settings. In concurrent and independent work, Kannan and Vempala [20] study several optimization problems in distributed settings, including k-means clustering under an interesting

[1] For k-median and k-means in general metric spaces, the bound on the size of the coreset can be obtained by replacing d with the logarithm of the total number of points. The analysis for general metric spaces is largely the same as that for d dimensional Euclidean space, so we will focus on Euclidean space and point out the difference when needed.



(a) Zhang et al.[29] (b) Our Construction

Figure 1: **(a)** Each node computes a coreset on the weighted pointset for its own data and its subtrees' coresets. **(b)** Local constant approximation solutions are computed, and the costs of these solutions are used to coordinate the construction of a local portion on each node.

separability assumption.

Section 6 provides a review of additional related work.

2 Preliminaries

Let $d(p, q)$ denote the Euclidean distance between any two points $p, q \in \mathbf{R}^d$. The goal of k-means clustering is to find a set of k centers $\mathbf{x} = \{x_1, x_2, \ldots, x_k\}$ which minimize the k-means cost of data set $P \subseteq \mathbf{R}^d$. Here the k-means cost is defined as $\text{cost}(P, \mathbf{x}) = \sum_{p \in P} d(p, \mathbf{x})^2$ where $d(p, \mathbf{x}) = \min_{x \in \mathbf{x}} d(p, x)$. If P is a weighted data set with a weighting function w, then the k-means cost is defined as $\sum_{p \in P} w(p) d(p, \mathbf{x})^2$. Similarly, the k-median cost is defined as $\sum_{p \in P} d(p, \mathbf{x})$. Both k-means and k-median cost functions are known to be **NP**-hard to minimize (see for example [2]). For both objectives, there exist several readily available polynomial-time algorithms that achieve constant approximation solutions (see for example [21, 24]).

In the distributed clustering task, we consider a set of n nodes $V = \{v_i, 1 \leq i \leq n\}$ which communicate on an undirected connected graph $G = (V, E)$ with $m = |E|$ edges. More precisely, an edge $(v_i, v_j) \in E$ indicates that v_i and v_j can communicate with each other. Here we measure the communication cost in number of points transmitted, and assume for simplicity that there is no latency in the communication. On each node v_i, there is a local set of data points P_i, and the global data set is $P = \bigcup_{i=1}^{n} P_i$. The goal is to find a set of k centers $\mathbf{x}$ which optimize $\text{cost}(P, \mathbf{x})$ while keeping the computation efficient and the communication cost as low as possible. Our focus is to reduce the total communication cost while preserving theoretical guarantees for approximating clustering cost.

2.1 Coresets

For the distributed clustering task, a natural approach to avoid broadcasting raw data is to generate a local summary of the relevant information. If each site computes a summary for their own data set and then communicates this to a central coordinator, a solution can be computed from a much smaller amount of data, drastically reducing the communication.

In the centralized setting, the idea of summarization with respect to the clustering task is captured by the concept of coresets [18, 12]. A coreset is a set of points, together with a weight for each point, such that

the cost of this weighted set approximates the cost of the original data for any set of k centers. The formal definition of coresets is:

Definition 1 (coreset). *An ϵ-coreset for a set of points P with respect to a center-based cost function is a set of points S and a set of weights $w : S \to \mathbf{R}$ such that for any set of centers $\mathbf{x}$,*

$$(1-\epsilon)\text{cost}(P,\mathbf{x}) \leq \sum_{p\in S} w(p)\text{cost}(p,\mathbf{x}) \leq (1+\epsilon)\text{cost}(P,\mathbf{x}).$$

In the centralized setting, many coreset construction algorithms have been proposed for k-median, k-means and some other cost functions. For example, for points in $\mathbf{R}^d$, algorithms in [12] construct coresets of size $t = \tilde{O}(kd/\epsilon^4)$ for k-means and coresets of size $t = \tilde{O}(kd/\epsilon^2)$ for k-median. In the distributed setting, it is natural to ask whether there exists an algorithm that constructs a small coreset for the entire point set but still has low communication cost. Note that the union of coresets for multiple data sets is a coreset for the union of the data sets. The immediate construction of combining the local coresets from each node would produce a global coreset whose size was larger by a factor of n, greatly increasing the communication complexity. We present a distributed algorithm which constructs a global coreset the same size as the centralized construction and only needs a single value[2] communicated to each node. This serves as the basis for our distributed clustering algorithm.

3 Distributed Coreset Construction

In this section, we design a distributed coreset construction algorithm for k-means and k-median. Note that the underlying technique can be extended to other additive clustering objectives such as k-line median.

To gain some intuition on the distributed coreset construction algorithm, we briefly review the coreset construction algorithm in [12] in the centralized setting. The coreset is constructed by computing a constant approximation solution for the entire data set, and then sampling points proportional to their contributions to the cost of this solution. Intuitively, the points close to the nearest centers can be approximately represented by the nearest centers while points far away cannot be well represented. Thus, points should be sampled with probability proportional to their contributions to the cost.

Directly adapting the algorithm to the distributed setting would require computing a constant approximation solution for the entire data set. We show that a global coreset can be constructed in a distributed fashion by estimating the weight of the entire data set with the sum of local approximations. We first compute a local approximation solution for each local data set, and communicate the total costs of these local solutions. Then we sample points proportional to their contributions to the cost of their local solutions. At the end of the algorithm, the coreset consists of the sampled points and the centers in the local solutions. The coreset points are distributed over the nodes, so we call it distributed coreset. See Algorithm 1 for details.

Theorem 1. *For distributed k-means and k-median clustering on a graph, there exists an algorithm such that with probability at least $1-\delta$, the union of its output on all nodes is an ϵ-coreset for $P = \bigcup_{i=1}^{n} P_i$. The size of the coreset is $O(\frac{1}{\epsilon^4}(kd + \log\frac{1}{\delta}) + nk\log\frac{nk}{\delta})$ for k-means, and $O(\frac{1}{\epsilon^2}(kd+\log\frac{1}{\delta})+nk)$ for k-median. The total communication cost is $O(mn)$.*

[2]The value that is communicated is the sum of the costs of approximations to the local optimal clustering. This is guaranteed to be no more than a constant factor times larger than the optimal cost.

Algorithm 1 Communication aware distributed coreset construction

Input: Local datasets $\{P_i, 1 \leq i \leq n\}$, parameter t (number of points to be sampled).

Round 1: on each node $v_i \in V$

- Compute a constant approximation B_i for P_i.
  Communicate $\text{cost}(P_i, B_i)$ to all other nodes.

Round 2: on each node $v_i \in V$

- Set $t_i = \frac{t \text{ cost}(P_i,B_i)}{\sum_{j=1}^{n} \text{cost}(P_j,B_j)}$ and $m_p = \text{cost}(p, B_i), \forall p \in P_i$.
- Pick a non-uniform random sample S_i of t_i points from P_i,
  where for every $q \in S_i$ and $p \in P_i$, we have $q = p$ with probability $m_p / \sum_{z \in P_i} m_z$.
  Let $w_q = \frac{\sum_i \sum_{z \in P_i} m_z}{t m_q}$ for each $q \in S_i$.
- For $\forall b \in B_i$, let $P_b = \{p \in P_i : d(p, b) = d(p, B_i)\}$, $w_b = |P_b| - \sum_{q \in P_b \cap S} w_q$.

Output: Distributed coreset: points $S_i \cup B_i$ with weights $\{w_q : q \in S_i \cup B_i\}, 1 \leq i \leq n$.

As described below, the distributed coreset construction can be achieved by using Algorithm 1 with appropriate t, namely $O(\frac{1}{\epsilon^4}(kd + \log \frac{1}{\delta}) + nk \log \frac{nk}{\delta})$ for k-means and $O(\frac{1}{\epsilon^2}(kd + \log \frac{1}{\delta}))$ for k-median. The formal proofs are described in the following subsections.

3.1 Proof of Theorem 1: k-median

The analysis relies on the definition of the pseudo-dimension of a function space and a sampling lemma.

Definition 2 ([25, 12])**.** *Let F be a finite set of functions from a set P to $\mathbf{R}_{\geq 0}$. For $f \in F$, let $B(f, r) = \{p : f(p) \leq r\}$. The dimension of the function space $\dim(F, P)$ is the smallest integer d such that for any $G \subseteq P$, $\left|\{G \cap B(f, r) : f \in F, r \geq 0\}\right| \leq |G|^d$.*

Suppose we draw a sample S according to $\{m_p : p \in P\}$, namely for every $q \in S$ and every $p \in P$, we have $q = p$ with probability $\frac{m_p}{\sum_{z \in P} m_z}$. Set the weights of the points as $w_p = \frac{\sum_{z \in P} m_z}{m_p |S|}$ for $p \in P$. Then for any $f \in F$, the expectation of the weighted cost of S equals the cost of the original data P:

$$\begin{aligned} \mathbf{E}\left[\sum_{q \in S} w_q f(q)\right] &= \sum_{q \in S} \mathbf{E}[w_q f(q)] = \sum_{q \in S} \sum_{p \in P} \Pr[q = p] w_p f(p) \\ &= \sum_{q \in S} \sum_{p \in P} \frac{m_p}{\sum_{z \in P} m_z} \frac{\sum_{z \in P} m_z}{m_p |S|} f(p) = \sum_{q \in S} \sum_{p \in P} \frac{1}{|S|} f(p) = \sum_{p \in P} f(p). \end{aligned}$$

The following lemma shows that if the sample size is large enough, then we also have concentration for any $f \in F$. The lemma is implicit in [12] and we include the proof in the appendix for completeness.

Lemma 1. *Fix a set F of functions $f : P \to \mathbf{R}_{\geq 0}$. Let S be a sample drawn i.i.d. from P according to $\{m_p : p \in P\}$, namely, for every $q \in S$ and every $p \in P$, we have $q = p$ with probability $\frac{m_p}{\sum_{z \in P} m_z}$. Let $w_p = \frac{\sum_{z \in P} m_z}{m_p |S|}$ for $p \in P$. For a sufficiently large c, if $|S| \geq \frac{c}{\epsilon^2}\left(\dim(F, P) + \log \frac{1}{\delta}\right)$ then with probability at least $1 - \delta, \forall f \in F : \left|\sum_{p \in P} f(p) - \sum_{q \in S} w_q f(q)\right| \leq \epsilon \left(\sum_{p \in P} m_p\right) \left(\max_{p \in P} \frac{f(p)}{m_p}\right)$.*

To get a small bound on the difference between $\sum_{p\in P} f(p)$ and $\sum_{q\in S} w_q f(q)$, we need to choose m_p such that $\max_{p\in P} \frac{f(p)}{m_p}$ is bounded. More precisely, if we choose $m_p = \max_{f\in F} f(p)$, then the difference is bounded by $\epsilon \sum_{p\in P} m_p$.

We first consider the centralized setting and review how [12] applied the lemma to construct a coreset for k-median as in Definition 1. A natural approach is to apply this lemma directly to the cost, namely, to choose $f_{\mathbf{x}}(p) := \mathrm{cost}(p, \mathbf{x})$. The problem is that a suitable upper bound m_p is not available for $\mathrm{cost}(p, \mathbf{x})$. However, we can still apply the lemma to a different set of functions defined as follows. Let b_p denote the closest center to p in the approximation solution. Aiming to approximate the error $\sum_p[\mathrm{cost}(p, \mathbf{x}) - \mathrm{cost}(b_p, \mathbf{x})]$ rather than to approximate $\sum_p \mathrm{cost}(p, \mathbf{x})$ directly, we define $f_{\mathbf{x}}(p) := \mathrm{cost}(p, \mathbf{x}) - \mathrm{cost}(b_p, \mathbf{x}) + \mathrm{cost}(p, b_p)$, where $\mathrm{cost}(p, b_p)$ is added so that $f_{\mathbf{x}}(p) \geq 0$. Since $0 \leq f_{\mathbf{x}}(p) \leq 2\mathrm{cost}(p, b_p)$, we can apply the lemma to $f_{\mathbf{x}}(p)$ and $m_p = 2\mathrm{cost}(p, b_p)$. The lemma then bounds the difference $|\sum_{p\in P} f_{\mathbf{x}}(p) - \sum_{q\in S} w_q f_{\mathbf{x}}(q)|$ by $2\epsilon \sum_{p\in P} \mathrm{cost}(p, b_p)$, so we have an $O(\epsilon)$-approximation.

Note that $\sum_{p\in P} f_{\mathbf{x}}(p) - \sum_{q\in S} w_q f_{\mathbf{x}}(q)$ does not equal $\sum_{p\in P} \mathrm{cost}(p, \mathbf{x}) - \sum_{q\in S} w_q \mathrm{cost}(q, \mathbf{x})$. However, it equals the difference between $\sum_{p\in P} \mathrm{cost}(p, \mathbf{x})$ and a weighted cost of the sampled points and the centers in the approximation solution. To get a coreset as in Definition 1, we need to add the centers of the approximation solution with specific weights to the coreset. Then when the sample is sufficiently large, the union of the sampled points and the centers is an ϵ-coreset.

Our key contribution in this paper is to show that in the distributed setting, it suffices to choose b_p from the local approximation solution for the local dataset containing p, rather than from an approximation solution for the global dataset. Furthermore, the sampling and the weighting of the coreset points can be done in a local manner. In the following, we provide a formal verification of our discussion above. We have the following lemma for k-median with $F = \{f_{\mathbf{x}} : f_{\mathbf{x}}(p) = d(p, \mathbf{x}) - d(b_p, \mathbf{x}) + d(p, b_p), \mathbf{x} \in (\mathbf{R}^d)^k\}$.

Lemma 2. *For k-median, the output of Algorithm 1 is an ϵ-coreset with probability at least $1 - \delta$, if $t \geq \frac{c}{\epsilon^2}\left(\dim(F, P) + \log\frac{1}{\delta}\right)$ for a sufficiently large constant c.*

Proof. We want to show that for any set of centers $\mathbf{x}$ the true cost for using these centers is well approximated by the cost on the weighted coreset. Note that our coreset has two types of points: sampled points $p \in S = \cup_{i=1}^n S_i$ with weight $w_p := \frac{\sum_{z\in P} m_z}{m_p|S|}$ and local solution centers $b \in B = \cup_{i=1}^n B_i$ with weight $w_b := |P_b| - \sum_{p\in S\cap P_b} w_p$. We use b_p to represent the nearest center to p in the local approximation solution. We use P_b to represent the set of points having b as their closest center in the local approximation solution.

As mentioned above, we construct $f_{\mathbf{x}}$ to be the difference between the cost of p and the cost of b_p on $\mathbf{x}$ so that Lemma 1 can be applied to $f_{\mathbf{x}}$. Note that $0 \leq f_{\mathbf{x}}(p) \leq 2d(p, b_p)$ by triangle inequality, and S is sufficiently large and chosen according to weights $m_p = d(p, b_p)$, so the conditions of Lemma 1 are met. Then we have

$$D = \left|\sum_{p\in P} f_{\mathbf{x}}(p) - \sum_{q\in S} w_q f_{\mathbf{x}}(q)\right| \leq 2\epsilon \sum_{p\in P} m_p = 2\epsilon \sum_{p\in P} d(p, b_p) = 2\epsilon \sum_{i=1}^{n} d(P_i, B_i) \leq O(\epsilon) \sum_{p\in P} d(p, \mathbf{x})$$

where the last inequality follows from the fact that B_i is a constant approximation solution for P_i.

Next, we show that the coreset is constructed such that D is exactly the difference between the true cost and the weighted cost of the coreset, which then leads to the lemma.

Note that the centers are weighted such that

$$\sum_{b\in B} w_b d(b, \mathbf{x}) = \sum_{b\in B} |P_b| d(b, \mathbf{x}) - \sum_{b\in B} \sum_{q\in S\cap P_b} w_q d(b, \mathbf{x}) = \sum_{p\in P} d(b_p, \mathbf{x}) - \sum_{q\in S} w_q d(b_q, \mathbf{x}). \tag{1}$$

Also note that $\sum_{p\in P} m_p = \sum_{q\in S} w_q m_q$, so

$$D = \left|\sum_{p\in P}\left[d(p,\mathbf{x}) - d(b_p,\mathbf{x}) + m_p\right] - \sum_{q\in S} w_q\left[d(q,\mathbf{x}) - d(b_q,\mathbf{x}) + m_q\right]\right|$$
$$= \left|\sum_{p\in P} d(p,\mathbf{x}) - \sum_{q\in S} w_q d(q,\mathbf{x}) - \left[\sum_{p\in P} d(b_p,\mathbf{x}) - \sum_{q\in S} w_q d(b_q,\mathbf{x})\right]\right|. \quad (2)$$

By plugging (1) into (2), we have

$$D = \left|\sum_{p\in P} d(p,\mathbf{x}) - \sum_{q\in S} w_q d(q,\mathbf{x}) - \sum_{b\in B} w_b d(b,\mathbf{x})\right| = \left|\sum_{p\in P} d(p,\mathbf{x}) - \sum_{q\in S\cup B} w_q d(q,\mathbf{x})\right|$$

which implies the lemma. □

In [12] it is shown that [3] $\dim(F,P) = O(kd)$. Therefore, by Lemma 2, when $|S| \geq O\left(\frac{1}{\epsilon^2}(kd + \log\frac{1}{\delta})\right)$, the weighted cost of $S\cup B$ approximates the k-median cost of P for any set of centers, then $(S\cup B, w)$ is an ϵ-coreset for P. The total communication cost is bounded by $O(mn)$, since even in the most general case when every node only knows its neighbors, we can broadcast the local costs with $O(mn)$ communication (see Algorithm 3).

3.2 Proof of Theorem 1: k-means

We have for k-means a similar lemma that when $t = O(\frac{1}{\epsilon^4}(kd+\log\frac{1}{\delta})+nk\log\frac{nk}{\delta}))$, the algorithm constructs an ϵ-coreset with probability at least $1-\delta$. The key idea is the same as that for k-median: we use centers b_p from the local approximation solutions as an approximation to the original data points p, and show that the error between the total cost and the weighted sample cost is approximately the error between the cost of p and its sampled cost (compensated by the weighted centers), which is shown to be small by Lemma 1.

The key difference between k-means and k-median is that triangle inequality applies directly to the k-median cost. In particular, for the k-median problem note that $\mathrm{cost}(b_p,p) = d(b_p,p)$ is an upper bound for the error of b_p on any set of centers, i.e. $\forall\mathbf{x}\in(\mathbf{R}^d)^k$, $d(b_p,p) \geq |d(p,\mathbf{x}) - d(b_p,\mathbf{x})| = |\mathrm{cost}(p,\mathbf{x}) - \mathrm{cost}(b_p,\mathbf{x})|$ by triangle inequality. Then we can construct $f_{\mathbf{x}}(p) := \mathrm{cost}(p,\mathbf{x}) - \mathrm{cost}(b_p,\mathbf{x}) + d(b_p,p)$ such that $h_p(\mathbf{x})$ is bounded. In contrast, for k-means, the error $|\mathrm{cost}(p,\mathbf{x}) - \mathrm{cost}(b_p,\mathbf{x})| = |d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2|$ does not have such an upper bound. The main change to the analysis is that we divide the points into two categories: good points whose costs approximately satisfy the triangle inequality (up to a factor of $1/\epsilon$) and bad points. The good points for a fixed set of centers $\mathbf{x}$ are defined as

$$G(\mathbf{x}) = \{p\in P : |\mathrm{cost}(p,\mathbf{x}) - \mathrm{cost}(b_p,\mathbf{x})| \leq \Delta_p\}$$

where the upper bound is $\Delta_p = \frac{\mathrm{cost}(p,b_p)}{\epsilon}$. Good points we can bound as before. For bad points we can show that while the difference in cost may be larger than $\mathrm{cost}(p,b_p)/\epsilon$, it must still be small, namely $O(\epsilon\min\{\mathrm{cost}(p,\mathbf{x}),\mathrm{cost}(b_p,\mathbf{x})\})$.

[3] For both k-median and k-means in general metric spaces, $\dim(F,P) = O(k\log|P|)$, so the bound for general metric spaces (including Euclidean space we focus on) can be obtained by replacing d with $\log|P|$.

Formally, the functions $f_{\mathbf{x}}(p)$ are restricted to be defined only over good points:

$$f_{\mathbf{x}}(p) = \begin{cases} \text{cost}(p,\mathbf{x}) - \text{cost}(b_p,\mathbf{x}) + \Delta_p & \text{if } p \in G(\mathbf{x}), \\ 0 & \text{otherwise.} \end{cases}$$

Then $\sum_{p\in P} \text{cost}(p,\mathbf{x}) - \sum_{q\in S\cup B} w_q \text{cost}(q,\mathbf{x})$ is decomposed into three terms:

$$\sum_{p\in P} f_{\mathbf{x}}(p) - \sum_{q\in S} w_q f_{\mathbf{x}}(q) \tag{3}$$

$$+ \sum_{p\in P\setminus G(\mathbf{x})} [\text{cost}(p,\mathbf{x}) - \text{cost}(b_p,\mathbf{x}) + \Delta_p] \tag{4}$$

$$- \sum_{q\in S\setminus G(\mathbf{x})} w_q[\text{cost}(q,\mathbf{x}) - \text{cost}(b_q,\mathbf{x}) + \Delta_q] \tag{5}$$

Lemma 1 bounds (3) by $O(\epsilon)\text{cost}(P,\mathbf{x})$, but we need an accuracy of ϵ^2 to compensate for the $1/\epsilon$ factor in the upper bound, resulting in a $O(1/\epsilon^4)$ factor in the sample complexity.

We begin by bounding (4). Note that for each term in (4), $|\text{cost}(p,\mathbf{x}) - \text{cost}(b_p,\mathbf{x})| > \Delta_p$ since $p \notin G(\mathbf{x})$. Furthermore, $p \notin G(\mathbf{x})$ only when p and b_p are close to each other and far away from $\mathbf{x}$. In Lemma 3 we use this to show that $|\text{cost}(p,\mathbf{x}) - \text{cost}(b_p,\mathbf{x})| \le O(\epsilon)\min\{\text{cost}(p,\mathbf{x}), \text{cost}(b_p,\mathbf{x})\}$. The details are presented in the appendix.

Using Lemma 3, (4) can be bounded by $O(\epsilon)\sum_{p\in P\setminus G(\mathbf{x})} \text{cost}(p,\mathbf{x}) \le O(\epsilon)\text{cost}(P,\mathbf{x})$.

Similarly, by the definition of Δ_q and Lemma 3, (5) is bounded by

$$\begin{aligned} (5) &\le \sum_{q\in S\setminus G(\mathbf{x})} 2w_q|\text{cost}(q,\mathbf{x}) - \text{cost}(b_q,\mathbf{x})| \le O(\epsilon) \sum_{q\in S\setminus G(\mathbf{x})} w_q \,\text{cost}(b_q,\mathbf{x}) \\ &\le O(\epsilon)\sum_{b\in B}\left(\sum_{q\in P_b\cap S} w_q\right)\text{cost}(b,\mathbf{x}). \end{aligned}$$

Note that the expectation of $\sum_{q\in P_b\cap S} w_q$ is $|P_b|$. By a sampling argument (Lemma 4), if $t \ge O(nk\log\frac{nk}{\delta})$, then $\sum_{q\in P_b\cap S} w_q \le 2|P_b|$. Then (5) is bounded by $O(\epsilon)\sum_{b\in B}\text{cost}(b,\mathbf{x})|P_b| = O(\epsilon)\sum_{p\in P}\text{cost}(b_p,\mathbf{x})$ where $\sum_{p\in P}\text{cost}(b_p,\mathbf{x})$ is at most a constant factor more than the optimum cost.

Since each of (3),(4), and (5) is $O(\epsilon)\text{cost}(P,\mathbf{x})$, we know that their sum is the same magnitude. Combining the above bounds, we have $\left|\text{cost}(P,\mathbf{x}) - \sum_{q\in S\cup B} w_q\text{cost}(q,\mathbf{x})\right| \le O(\epsilon)\text{cost}(P,\mathbf{x})$. The proof is then completed by choosing a suitable ϵ, and bounding $\dim(F,P) = O(kd)$ as in [12].

4 Effect of Network Topology on Communication Cost

In the previous section, we presented a distributed coreset construction algorithm. The coreset constructed can then be used as a proxy for the original data, and we can run any distributed clustering algorithm on it. In this paper, we discuss the approach of simply collecting all local portions of the distributed coreset and run non-distributed clustering algorithm on it. If there is a central coordinator in the communication graph, then we can simply send the local portions of the coreset to the coordinator which can perform the clustering task. The total communication cost is just the size of the coreset.

In this section, we consider the distributed clustering tasks where the nodes are arranged in some arbitrary connected topology, and can only communicate with their neighbors. We propose a message passing approach for globally sharing information, and use it for collecting information for coreset construction and sharing the local portions of the coreset. We also consider the special case when the graph is a rooted tree.

Algorithm 2 Distributed clustering on a graph

Input: $\{P_i, 1 \leq i \leq n\}$: local datasets; $\{N_i, 1 \leq i \leq n\}$: the neighbors of v_i; $\mathcal{A}_\alpha$: an α-approximation algorithm for weighted clustering instances.

Round 1: on each node v_i
- Construct its local portion D_i of an $\epsilon/2$-coreset by Algorithm 1, using Message-Passing for communicating the local costs.

Round 2: on each node v_i
- Call Message-Passing(D_i, N_i).
- $\mathbf{x} = \mathcal{A}_\alpha(\bigcup_j D_j)$.

Output: $\mathbf{x}$

Algorithm 3 Message-Passing(I_i, N_i)

Input: I_i is the message, N_i are the neighbors.

- Let R_i denote the information received. Initialize $R_i = \{I_i\}$, and send I_i to all the neighbors.
- While $R_i \neq \{I_j, 1 \leq j \leq n\}$:
  If receive message $I_j \notin R_i$,
  $R_i = R_i \cup \{I_j\}$ and send I_j to all the neighbors.

4.1 General Graphs

We now present the main result for distributed clustering on graphs.

Theorem 2. *Given an α-approximation algorithm for weighted k-means (k-median respectively) as a subroutine, there exists an algorithm that with probability at least $1 - \delta$ outputs a $(1+\epsilon)\alpha$-approximation solution for distributed k-means (k-median respectively) clustering. The total communication cost is $O(m(\frac{1}{\epsilon^4}(kd + \log\frac{1}{\delta}) + nk\log\frac{nk}{\delta}))$ for k-means, and $O(m(\frac{1}{\epsilon^2}(kd + \log\frac{1}{\delta}) + nk))$ for k-median.*

Proof. The details are presented in Algorithm 2. By Theorem 1, the output of Algorithm 1 is a coreset. Observe that in Algorithm 3, for any j, I_j propagates on the graph in a breadth-first-search style, so at the end every node receives I_j. This holds for all $1 \leq j \leq n$, so all nodes has a copy of the coreset at the end, and thus the output is a $(1+\epsilon)\alpha$-approximation solution.

Also observe that in Algorithm 3, for any node v_i and $j \in [n]$, v_i sends out I_j once, so the communication of v_i is $|N_i| \times \sum_{j=1}^n |I_j|$. The communication cost of Algorithm 3 is $O(m\sum_{j=1}^n |I_j|)$. Then the total communication cost of Algorithm 2 follows from the size of the coreset constructed. □

In contrast, an approach where each node constructs an ϵ-coreset for k-means and sends it to the other nodes incurs communication cost of $\tilde{O}(\frac{mnkd}{\epsilon^4})$. Our algorithm significantly reduces this.

4.2 Rooted Trees

Our algorithm can also be applied on a rooted tree, and compares favorably to other approaches involving coresets [29]. We can restrict message passing to operating along this tree, leading to the following theorem.

Theorem 3. *Given an α-approximation algorithm for weighted k-means (k-median respectively) as a subroutine, there exists an algorithm that with probability at least $1-\delta$ outputs a $(1+\epsilon)\alpha$-approximation solution for distributed k-means (k-median respectively) clustering on a rooted tree of height h. The total communication cost is $O(h(\frac{1}{\epsilon^4}(kd+\log\frac{1}{\delta})+nk\log\frac{nk}{\delta}))$ for k-means, and $O(h(\frac{1}{\epsilon^2}(kd+\log\frac{1}{\delta})+nk))$ for k-median.*

Proof. We can construct the distributed coreset using Algorithm 1. In the construction, the costs of the local approximation solutions are sent from every node to the root, and the sum is sent to every node by the root. After the construction, the local portions of the coreset are sent from every node to the root. A local portion D_i leads to a communication cost of $O(|D_i|h)$, so the total communication cost is $O(h\sum_{i=1}^{n}|D_i|)$. Once the coreset is constructed at the root, the α-approximation algorithm can be applied centrally, and the results can be sent back to all nodes. □

Our approach improves the cost of $\tilde{O}(\frac{nh^4kd}{\epsilon^4})$ for k-means and the cost of $\tilde{O}(\frac{nh^2kd}{\epsilon^2})$ for k-median in [29] [4]. The algorithm in [29] builds on each node a coreset for the union of coresets from its children, and thus needs $O(\epsilon/h)$ accuracy to prevent the accumulation of errors. Since the coreset construction subroutine has quadratic dependence on $1/\epsilon$ for k-median (quartic for k-means), the algorithm then has quadratic dependence on h (quartic for k-means). Our algorithm does not build coreset on top of coresets, resulting in a better dependence on the height of the tree h.

In a general graph, any rooted tree will have its height h at least as large as half the diameter. For sensors in a grid network, this implies $h=\Omega(\sqrt{n})$. In this case, our algorithm gains a significant improvement over existing algorithms.

5 Experiments

In our experiments we seek to determine whether our algorithm is effective for the clustering tasks and how it compares to the other distributed coreset algorithms [5]. We present the k-means cost of the solution produced by our algorithm with varying communication cost, and compare to those of other algorithms when they use the same amount of communication.

Data sets: Following the setup of [29, 4], for the synthetic data we randomly choose $k=5$ centers from the standard Gaussian distribution in $\mathbf{R}^{10}$, and sample equal number of $20,000$ points from the Gaussian distribution around each center. Note that, as in [29, 4], we use the cost of the centers as a baseline for comparing the clustering quality. We choose the following real world data sets from [3]: Spam (4601 points in

[4] Their algorithm used coreset construction as a subroutine. The construction algorithm they used builds coreset of size $\tilde{O}(\frac{nkh}{\epsilon^d}\log|P|)$. Throughout this paper, when we compare to [29] we assume they use the coreset construction technique of [12] to reduce their coreset size and communication cost.

[5] Our theoretical analysis shows that our algorithm has better bounds on the communication cost. Since the bounds are from worst-case analysis, it is meaningful to verify that our algorithm also empirically outperforms other distributed coreset algorithms.

$\mathbf{R}^{58}$), Pendigits (10992 points in $\mathbf{R}^{16}$), Letter (20000 points in $\mathbf{R}^{16}$), and ColorHistogram of the Corel Image data set (68040 points in $\mathbf{R}^{32}$). We use $k = 10$ for these data sets. We further choose YearPredictionMSD (515345 points in $\mathbf{R}^{90}$) for larger scale experiments, and use $k = 50$ for this data set.

Experimental Methodology: To transform the centralized clustering data sets into distributed data sets we first generate a communication graph connecting local sites, and then partition the data into local data sets. To evaluate our algorithm, we consider several network topologies and partition methods.

The algorithms are evaluated on three types of communication graphs: random, grid, and preferential. The random graphs are Erdös-Renyi graphs $G(n, p)$ with $p = 0.3$, i.e. they are generated by including each potential edge independently with probability 0.3. The preferential graphs are generated according to the preferential attachment mechanism in the Barabási-Albert model [1]. For data sets Spam, Pendigits, and Letter, we use random/preferential graphs with 10 sites and 3×3 grid graphs. For synthetic data set and ColorHistogram, we use random/preferential graphs with 25 sites and 5×5 grid graphs. For large data set YearPredictionMSD, we use random/preferential graphs with 100 sites and 10×10 grid graphs.

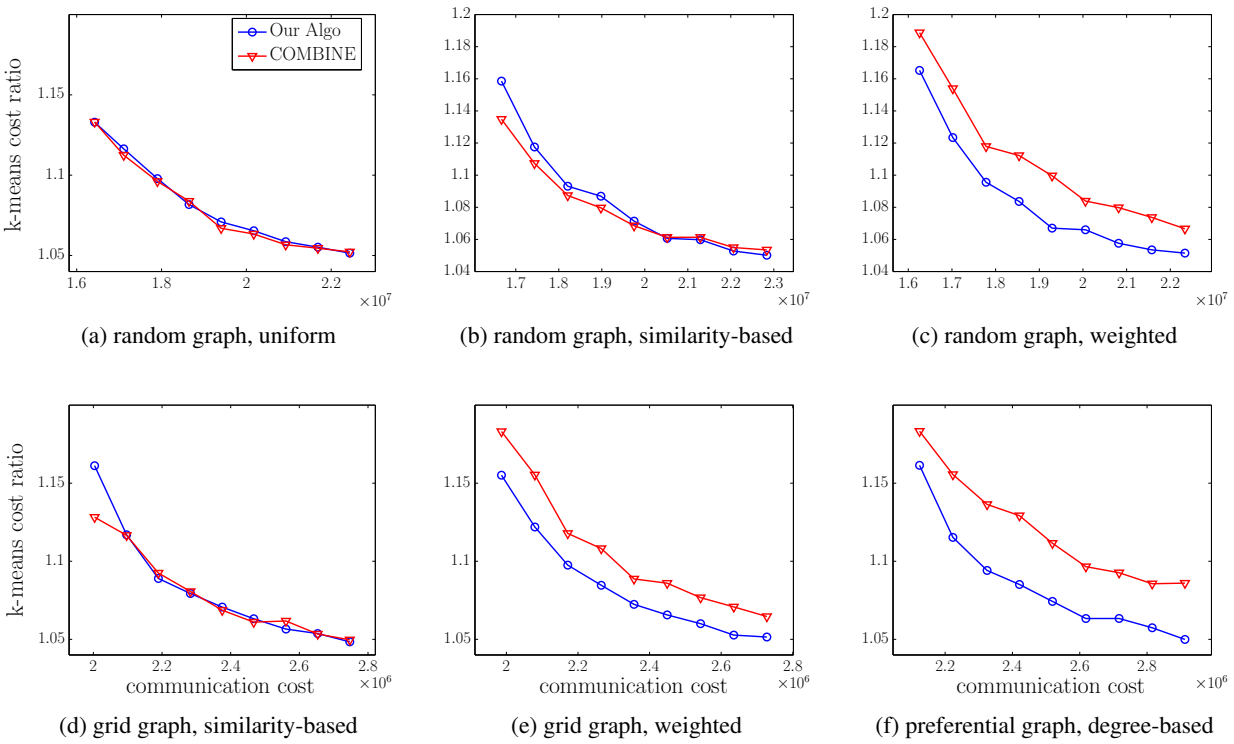
The data is then distributed over the local sites. When the communication network is a random graph, we consider three partition methods: uniform, similarity-based, and weighted. In the uniform partition, each data point in the global data set is assigned to the local sites with equal probability. In the similarity-based partition, each site has an associated data point randomly selected from the global data. Each data point in the global data is then assigned to the site with probability proportional to its similarity to the associated point of the site, where the similarities are computed by Gaussian kernel function. In the weighted partition, each local site is assigned a weight chosen by $|N(0, 1)|$ and then each data point is distributed to the local sites with probability proportional to the site's weight. When the network is a grid graph, we consider the similarity-based and weighted partitions. When the network is a preferential graph, we consider the degree-based partition, where each point is assigned with probability proportional to the site's degree.

To measure the quality of the coreset generated, we run Lloyd's algorithm on the coreset and the global data respectively to get two solutions, and compute the ratio between the costs of the two solutions over the global data. The average ratio over 30 runs is then reported. We compare our algorithm with COMBINE, the method of combining a coreset from each local data set, and with the algorithm of [29] (Zhang et al.). When running the algorithm of Zhang et al., we restrict the general communication network to a spanning tree by picking a root uniformly at random and performing a breadth first search.

Results: Here we focus on the results of the largest data set YearPredictionMSD, and in Appendix B we present the experimental results for all the data sets.

Figure 2 shows the results over different network topologies and partition methods. We observe that the algorithms perform well with much smaller coreset sizes than predicted by the theoretical bounds. For example, to get 1.1 cost ratio, the coreset size and thus the communication needed is only $0.1\% - 1\%$ of the theoretical bound.

In the uniform partition, our algorithm performs nearly the same as COMBINE. This is not surprising since our algorithm reduces to the COMBINE algorithm when each local site has the same cost and the two algorithms use the same amount of communication. In this case, since in our algorithm the sizes of the local samples are proportional to the costs of the local solutions, it samples the same number of points from each local data set. This is equivalent to the COMBINE algorithm with the same amount of communication. In the similarity-based partition, similar results are observed as it also leads to balanced local costs. However, when the local sites have significantly different costs (as in the weighted and degree-based partitions), our algorithm outperforms COMBINE. As observed in Figure 2, the costs of our solutions consistently improve



(a) random graph, uniform (b) random graph, similarity-based (c) random graph, weighted

(d) grid graph, similarity-based (e) grid graph, weighted (f) preferential graph, degree-based

Figure 2: k-means cost (normalized by baseline) v.s. communication cost over graphs. The titles indicate the network topology and partition method.

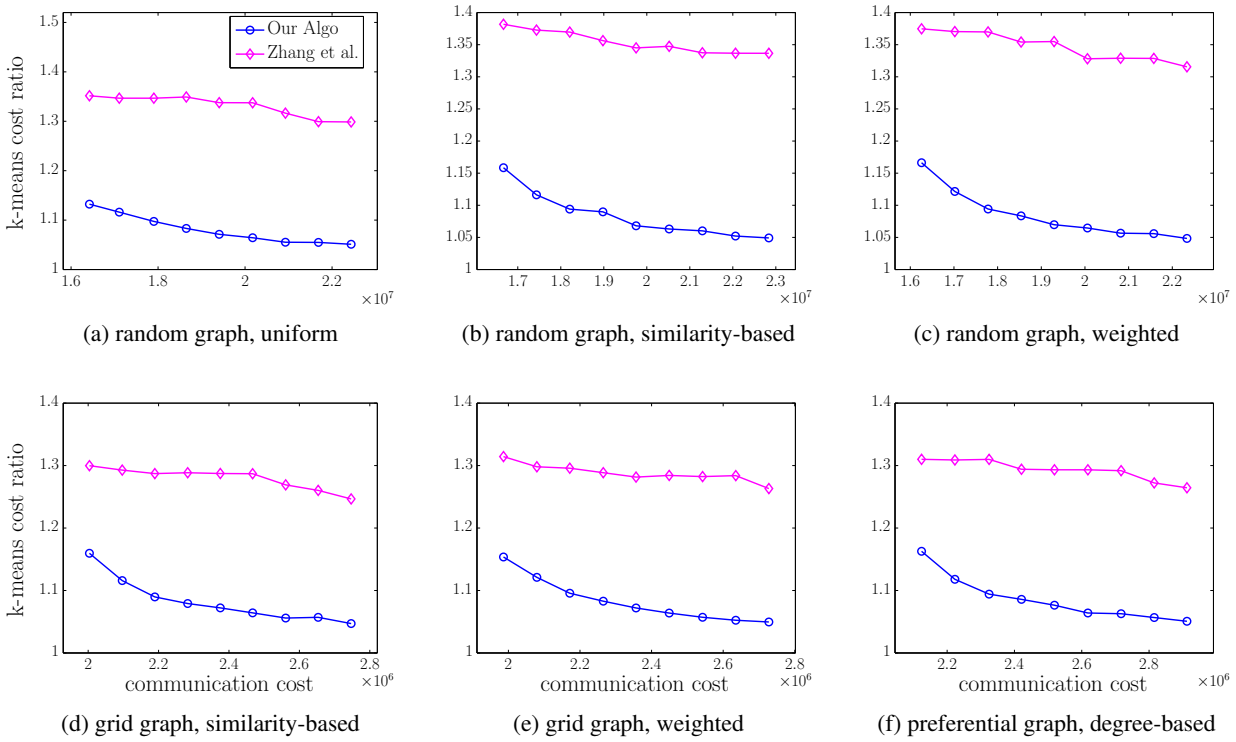
over those of COMBINE by $2\% - 5\%$. Our algorithm then saves $10\% - 20\%$ communication cost to achieve the same approximation ratio.

Figure 3 shows the results over the spanning trees of the graphs. Our algorithm performs much better than the algorithm of Zhang et al., achieving about 20% improvement in cost. This is due to the fact that their algorithm needs larger coresets to prevent the accumulation of errors when constructing coresets from component coresets, and thus needs higher communication cost to achieve the same approximation ratio.

Similar results are observed on the other datasets, which are presented in Appendix B.

6 Additional Related Work

Many empirical algorithms adapt the centralized algorithms to the distributed setting. They generally provide no bound for the clustering quality or the communication cost. For instance, a technique is proposed in [14] to adapt several iterative center-based data clustering algorithms including Lloyd's algorithm for k-means to the distributed setting, where sufficient statistics instead of the raw data are sent to a central coordinator. This approach involves transferring data back and forth in each iteration, and thus the communication cost depends on the number of iterations. Similarly, the communication costs of the distributed clustering algorithms proposed in [10] and [28] depend on the number of iterations. Some other algorithms gather local summaries and then perform global clustering on the summaries. The distributed density-based clustering algorithm in [19] clusters and computes summaries for the local data at each node, and sends the local summaries



(a) random graph, uniform (b) random graph, similarity-based (c) random graph, weighted

(d) grid graph, similarity-based (e) grid graph, weighted (f) preferential graph, degree-based

Figure 3: k-means cost (normalized by baseline) v.s. communication cost over the spanning trees of the graphs. The titles indicate the network topology and partition method.

to a central node where the global clustering is carried out. This algorithm only considers the flat two-tier topology. Some in-network aggregation schemes for computing statistics over distributed data are useful for such distributed clustering algorithms. For example, an algorithm is provided in [8] for approximate duplicate-sensitive aggregates across distributed data sets, such as SUM. An algorithm is proposed in [16] for power-preserving computation of order statistics such as quantile.

Several coreset construction algorithms have been proposed for k-median, k-means and k-line median clustering [18, 7, 17, 23, 12]. For example, the algorithm in [12] constructs a coreset of size $\tilde{O}(kd/\epsilon^2)$ whose cost approximates that of the original data up to accuracy ϵ with respect to k-median in $\mathbf{R}^d$. All of these algorithms consider coreset construction in the centralized setting, while our construction algorithm is for the distributed setting.

There has also been work attempting to parallelize clustering algorithms. [13] showed that coresets could be constructed in parallel and then merged together. Bahmani et al. [4] adapted `k-means++` to the parallel setting. Their algorithm, k-means$||$, essentially builds $O(1)$-coreset of size $O(k \log |P|)$. However, it cannot build ϵ-coreset for $\epsilon = o(1)$, and thus can only guarantee constant approximation solutions.

There is also related work providing approximation solutions for k-median based on random sampling [6]. Particularly, they showed that given a sample of size $\tilde{O}(\frac{k}{\epsilon^2})$ drawn i.i.d. from the data, there exists an algorithm that outputs a solution with an average cost bounded by twice the optimal average cost plus an error bound ϵ. If we convert it to a multiplicative approximation factor, the factor depends on the optimal average cost. When there are outlier points far away from all other points, the optimal average cost can be very small after

normalization, then the multiplicative approximation factor is large. The coreset approach provides better guarantees. Additionally, their approach is not applicable to k-means.

Balcan et al. [5] and Daume et al. [11] consider fundamental communication complexity questions arising when doing classification in distributed settings. In concurrent and independent work, Vempala et al. [20] study several optimization problems in distributed settings, including k-means clustering under an interesting separability assumption.

Acknowledgements This work was supported by ONR grant N00014-09-1-0751, AFOSR grant FA9550-09-1-0538, and by a Google Research Award. We thank Le Song for generously allowing us to use his computer cluster.

References

[1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 2002.

[2] P. Awasthi and M. Balcan. Center based clustering: A foundational perspective. Survey Chapter in Handbook of Cluster Analysis (Manuscript), 2013.

[3] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[4] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. In *Proceedings of the International Conference on Very Large Data Bases*, 2012.

[5] M.-F. Balcan, A. Blum, S. Fine, and Y. Mansour. Distributed learning, communication complexity and privacy. In *Proceedings of the Conference on Learning Thoery*, 2012.

[6] S. Ben-David. A framework for statistical clustering with a constant time approximation algorithms for k-median clustering. *Proceedings of Annual Conference on Learning Theory*, 2004.

[7] K. Chen. On k-median clustering in high dimensions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2006.

[8] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proceedings of the International Conference on Data Engineering*, 2004.

[9] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, et al. Spanner: Googles globally-distributed database. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, 2012.

[10] S. Datta, C. Giannella, H. Kargupta, et al. K-means clustering over peer-to-peer networks. In *Proceedings of the International Workshop on High Performance and Distributed Mining*, 2005.

[11] H. Daumé III, J. M. Phillips, A. Saha, and S. Venkatasubramanian. Efficient protocols for distributed classification and optimization. In *Algorithmic Learning Theory*, pages 154–168. Springer, 2012.

[12] D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2011.

[13] D. Feldman, A. Sugaya, and D. Rus. An effective coreset compression algorithm for large scale sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, 2012.

[14] G. Forman and B. Zhang. Distributed data clustering can be efficient and exact. *ACM SIGKDD Explorations Newsletter*, 2000.

[15] S. Greenhill and S. Venkatesh. Distributed query processing for mobile surveillance. In *Proceedings of the International Conference on Multimedia*, 2007.

[16] M. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2004.

[17] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 2007.

[18] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2004.

[19] E. Januzaj, H. Kriegel, and M. Pfeifle. Towards effective and efficient distributed clustering. In *Workshop on Clustering Large Data Sets in the IEEE International Conference on Data Mining*, 2003.

[20] R. Kannan and S. Vempala. Nimble algorithms for cloud computing. *arXiv preprint arXiv:1304.3162*, 2013.

[21] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the Annual Symposium on Computational Geometry*, 2002.

[22] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 2001.

[23] M. Langberg and L. Schulman. Universal ε-approximators for integrals. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2010.

[24] S. Li and O. Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2013.

[25] Y. Li, P. M. Long, and A. Srinivasan. Improved bounds on the sample complexity of learning. In *Proceedings of the eleventh annual ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[26] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti. Characterizing web-based video sharing workloads. *ACM Transactions on the Web*, 2011.

[27] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.

[28] D. Tasoulis and M. Vrahatis. Unsupervised distributed clustering. In *Proceedings of the International Conference on Parallel and Distributed Computing and Networks*, 2004.

[29] Q. Zhang, J. Liu, and W. Wang. Approximate clustering on distributed data streams. In *Proceedings of the IEEE International Conference on Data Engineering*, 2008.

A Proofs for Section 3

The proof of Lemma 1 follows from the analysis in [12], although not explicitly stated there. We begin with the following theorem for uniform sampling on a function space. The theorem is from [12] but rephrased for convenience.

Theorem 4 (Theorem 6.9 in [12]). *Let F be a set of functions from P to $\mathbf{R}_{\geq 0}$, and let $\epsilon \in (0,1)$. Let S be a sample of*

$$|S| = \frac{c}{\epsilon^2}(\dim(F,P) + \log\frac{1}{\delta})$$

i.i.d items from P, where c is a sufficiently large constant. Then, with probability at least $1-\delta$, for any $f \in F$ and any $r \geq 0$,

$$\left| \frac{\sum_{p\in P, f(p)\leq r} f(p)}{|P|} - \frac{\sum_{q\in S, f(q)\leq r} f(q)}{|S|} \right| \leq \epsilon r.$$

Proof of Lemma 1. Without loss of generality, assume $m_p \in \mathbf{N}^+$. Define G as follows: for each $p \in P$, include m_p copies $\{p_i\}_{i=1}^{m_p}$ of p in G and define $f(p_i) = f(p)/m_p$. Then S is equivalent to a sample draw i.i.d. and uniformly at random from G. We now apply Theorem 4 on G and $r = \max_{f\in F, p'\in G} f(p')$. By Theorem 4, we know that for any $f \in F$,

$$\left| \frac{\sum_{p'\in G} f(p')}{|G|} - \frac{\sum_{q'\in S} f(q')}{|S|} \right| \leq \epsilon \max_{p'\in G} f(p'). \tag{6}$$

The lemma then follows from multiplying both sides of (6) by $|G| = \sum_{p\in P} m_p$. Also note that the dimension $\dim(F,G)$ is the same as that of $\dim(F,P)$ as pointed out by [12]. □

Lemma 3. *If $d(p,b_p)^2/\epsilon \leq |d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2|$, then*

$$|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq 8\epsilon \min\{d(p,\mathbf{x})^2, d(b_p,\mathbf{x})^2\}.$$

Proof. We first have by triangle inequality

$$|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq d(p,b_p)[d(p,\mathbf{x}) + d(b_p,\mathbf{x})].$$

Then by $d(p,b_p)^2/\epsilon \leq |d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2|$,

$$d(p,b_p) \leq \epsilon[d(p,\mathbf{x}) + d(b_p,\mathbf{x})].$$

Therefore, we have

$$\begin{aligned}
|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| &\leq d(p,b_p)[d(p,\mathbf{x}) + d(b_p,\mathbf{x})] \leq \epsilon[d(p,\mathbf{x}) + d(b_p,\mathbf{x})]^2 \\
&\leq 2\epsilon[d(p,\mathbf{x})^2 + d(b_p,\mathbf{x})^2] \leq 2\epsilon[d(p,\mathbf{x})^2 + (d(p,\mathbf{x}) + d(p,b_p))^2] \\
&\leq 2\epsilon[d(p,\mathbf{x})^2 + 2d(p,\mathbf{x})^2 + 2d(p,b_p)^2] \leq 6\epsilon d(p,\mathbf{x})^2 + 4\epsilon d(p,b_p)^2 \\
&\leq 6\epsilon d(p,\mathbf{x})^2 + 4\epsilon^2 |d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2|
\end{aligned}$$

for sufficiently small ϵ. Then

$$|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq \frac{6\epsilon}{1-4\epsilon^2} d(p,\mathbf{x})^2 \leq 8\epsilon d(p,\mathbf{x})^2.$$

Similarly, $|d(p,\mathbf{x})^2 - d(b_p,\mathbf{x})^2| \leq 8\epsilon d(b_p,\mathbf{x})^2$. The lemma follows from the last two inequalities. □

Lemma 4 (Corollary 15.4 in [12]). *Let $0 < \delta < 1/2$, and $t \geq c|B|\log\frac{|B|}{\delta}$ for a sufficiently large c. Then with probability at least $1-\delta$, $\forall b \in B_i, \sum_{q\in P_b\cap S} w_q \leq 2|P_b|$.*

B Complete Experimental Results

Here we present the results of all the data sets over different network topologies and data partition methods.

Figure 4 shows the results of all the data sets on random graphs. The first column of Figure 4 shows that our algorithm and COMBINE perform nearly the same in the uniform data partition. This is not surprising since our algorithm reduces to the COMBINE algorithm when each local site has the same cost and the two algorithms use the same amount of communication. In this case, since in our algorithm the sizes of the local samples are proportional to the costs of the local solutions, it samples the same number of points from each local data set. This is equivalent to the COMBINE algorithm with the same amount of communication. In the similarity-based partition, similar results are observed as this partition method also leads to balanced local costs. However, in the weighted partition where local sites have significantly different contributions to the total cost, our algorithm outperforms COMBINE. It improves the k-means cost by $2\% - 5\%$, and thus saves $10\% - 30\%$ communication cost to achieve the same approximation ratio.

Figure 5 shows the results of all the data sets on grid and preferential graphs. Similar to the results on random graphs, our algorithm performs nearly the same as COMBINE in the similarity-based partition and outperforms COMBINE in the weighted partition and degree-based partition. Furthermore, Figure 4 and 5 also show that the performance of our algorithm merely changes over different network topologies and partition methods.

Figure 6 shows the results of all the data sets on the spanning trees of the random graphs and Figure 7 shows those on the spanning trees of the grid and preferential graphs. Compared to the algorithm of Zhang et al., our algorithm consistently shows much better performance on all the data sets in different settings. It improves the k-means cost by $10\% - 30\%$, and thus can achieve even better approximation ratio with only 10% communication cost. This is because the algorithm of Zhang et al. constructs coresets from component coresets and needs larger coresets to prevent the accumulation of errors. Figure 6 also shows that although their costs decrease with the increase of the communication, the decrease is slower on larger graphs (e.g., as in the experiments for YearPredictionMSD). This is due to the fact that the spanning tree of a larger graph has larger height, leading to more accumulation of errors. In this case, more communication is needed to prevent the accumulation.

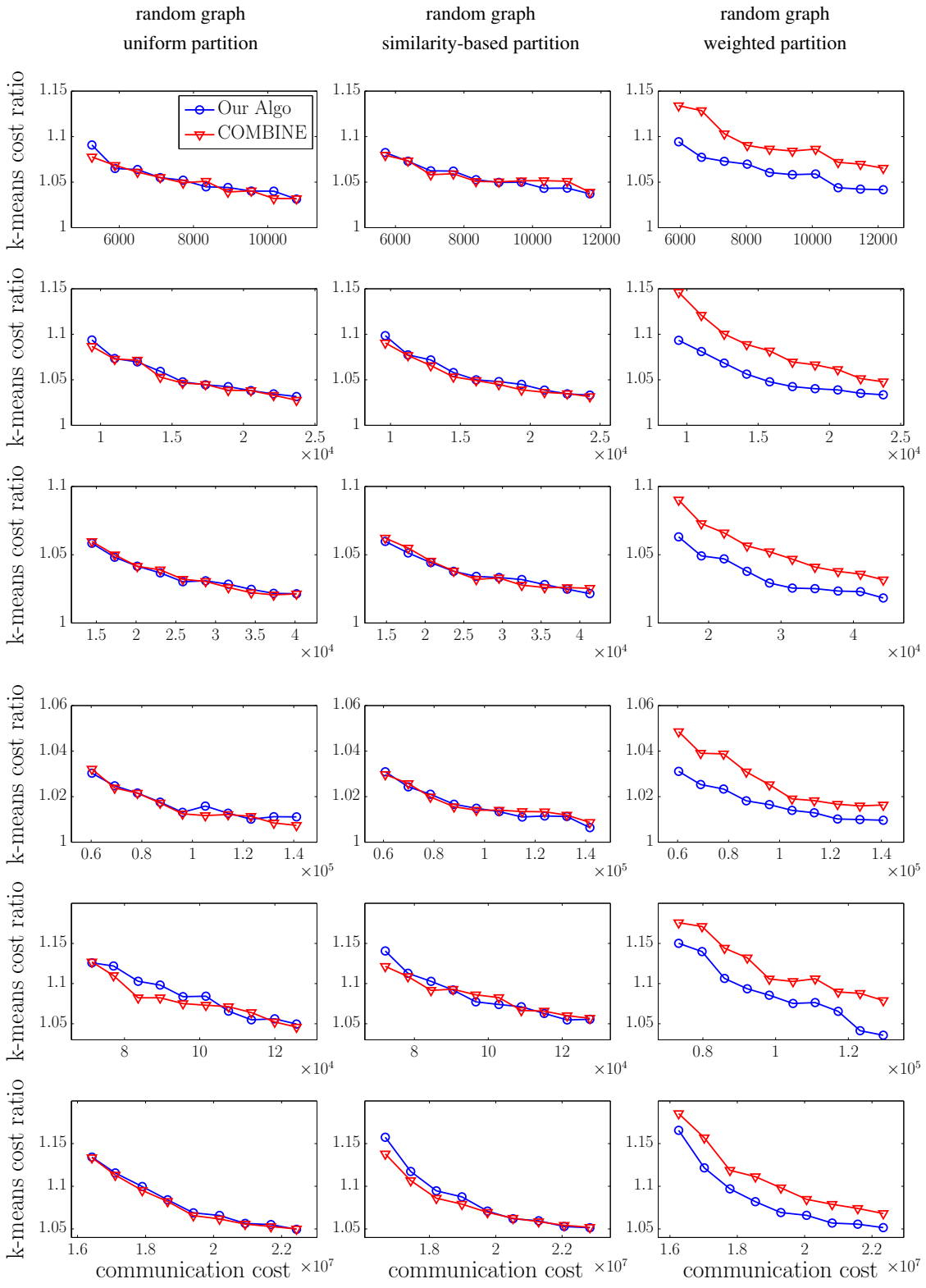


Figure 4: k-means cost on random graphs. Columns: random graph with uniform partition, random graph with similarity-based partition, and random graph with weighted partition. Rows: Spam, Pendigits, Letter, synthetic, ColorHistogram, and YearPredictionMSD.

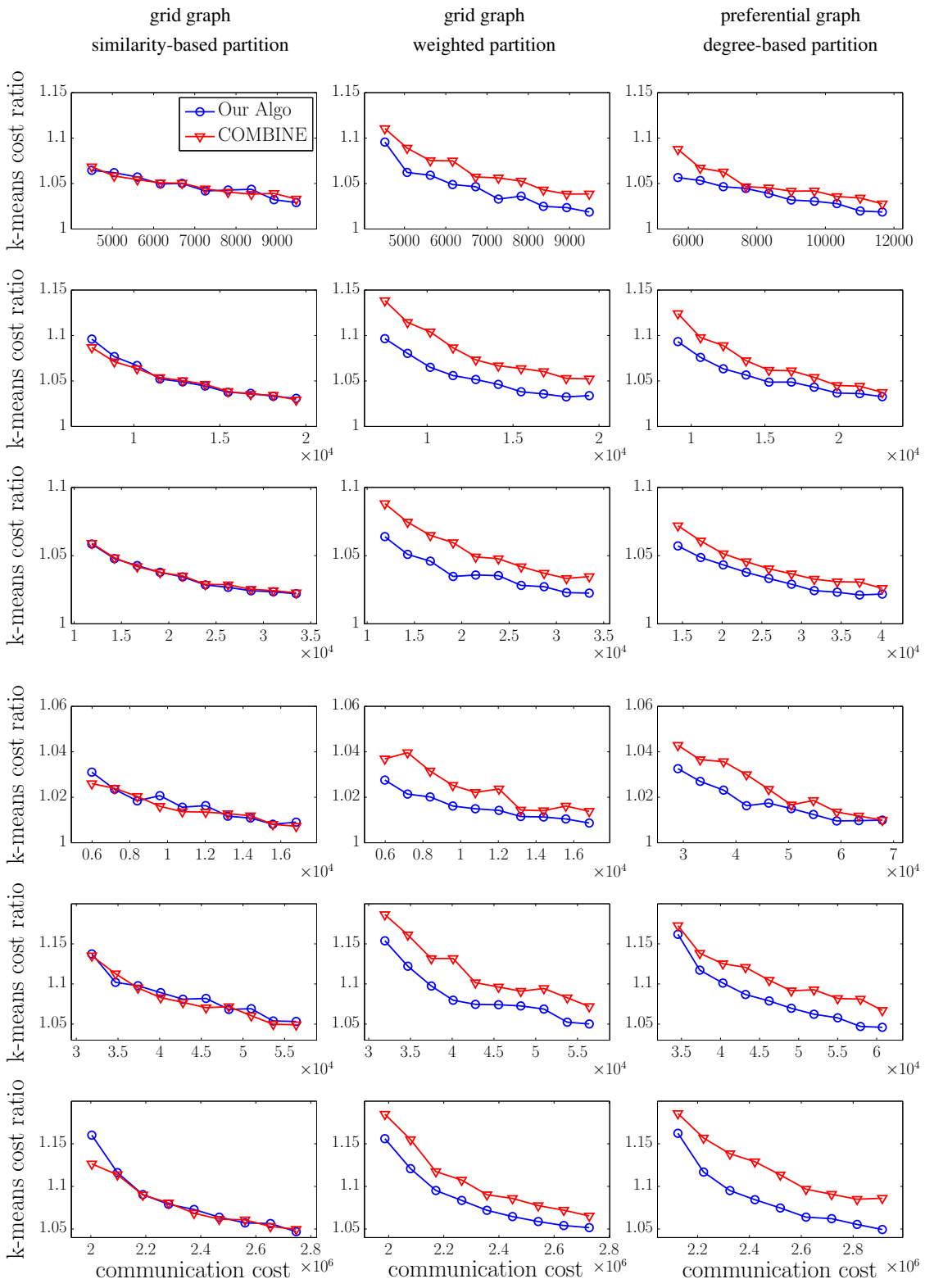


Figure 5: k-means cost on grid and preferential graphs. Columns: grid graph with similarity-based partition, grid graph with weighted partition, and preferential graph with degree-based partition. Rows: Spam, Pendigits, Letter, synthetic, ColorHistogram, and YearPredictionMSD.

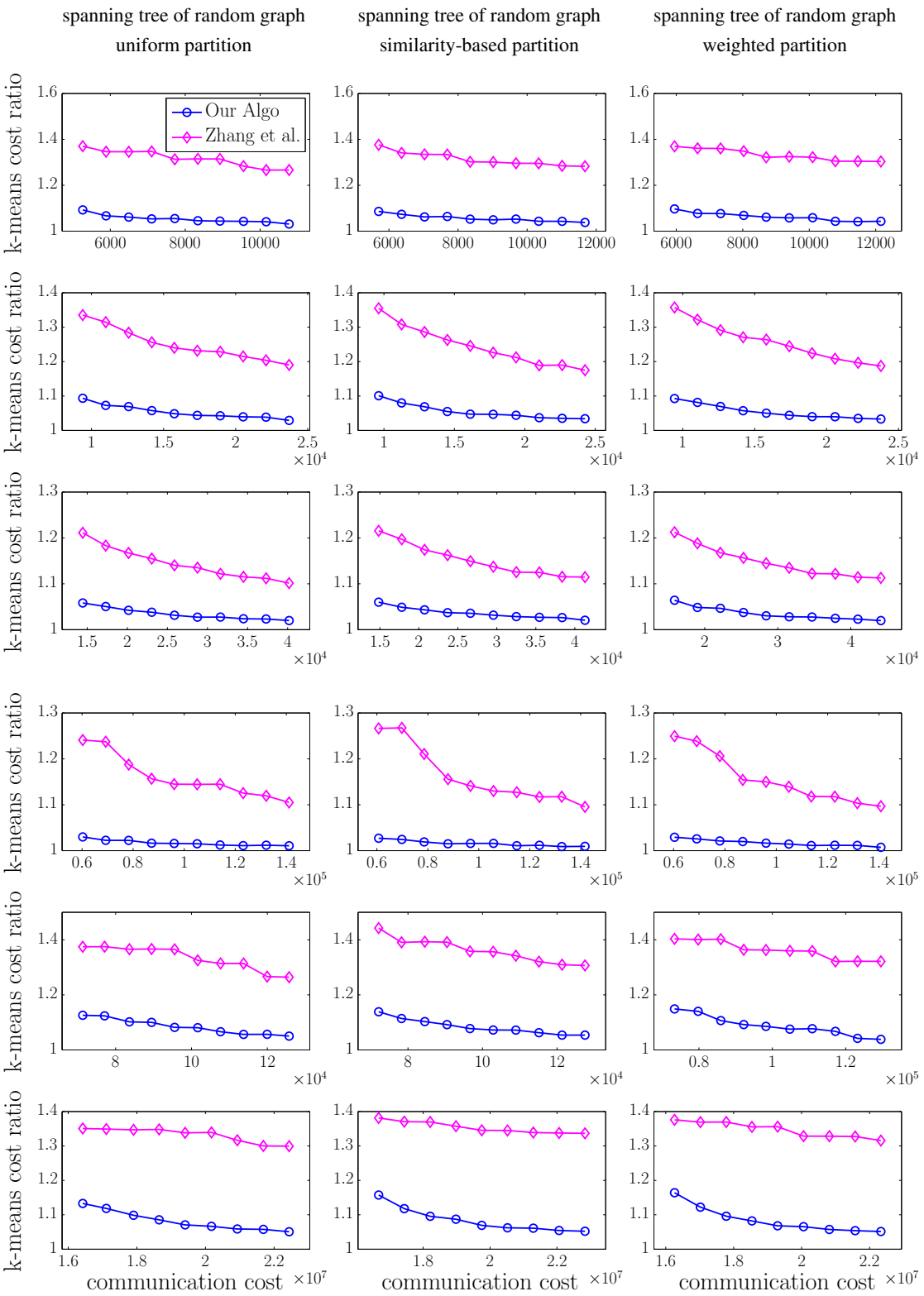


Figure 6: k-means cost on the spanning trees of the random graphs. Columns: random graph with uniform partition, random graph with similarity-based partition, and random graph with weighted partition. Rows: Spam, Pendigits, Letter, synthetic, ColorHistogram, and YearPredictionMSD.

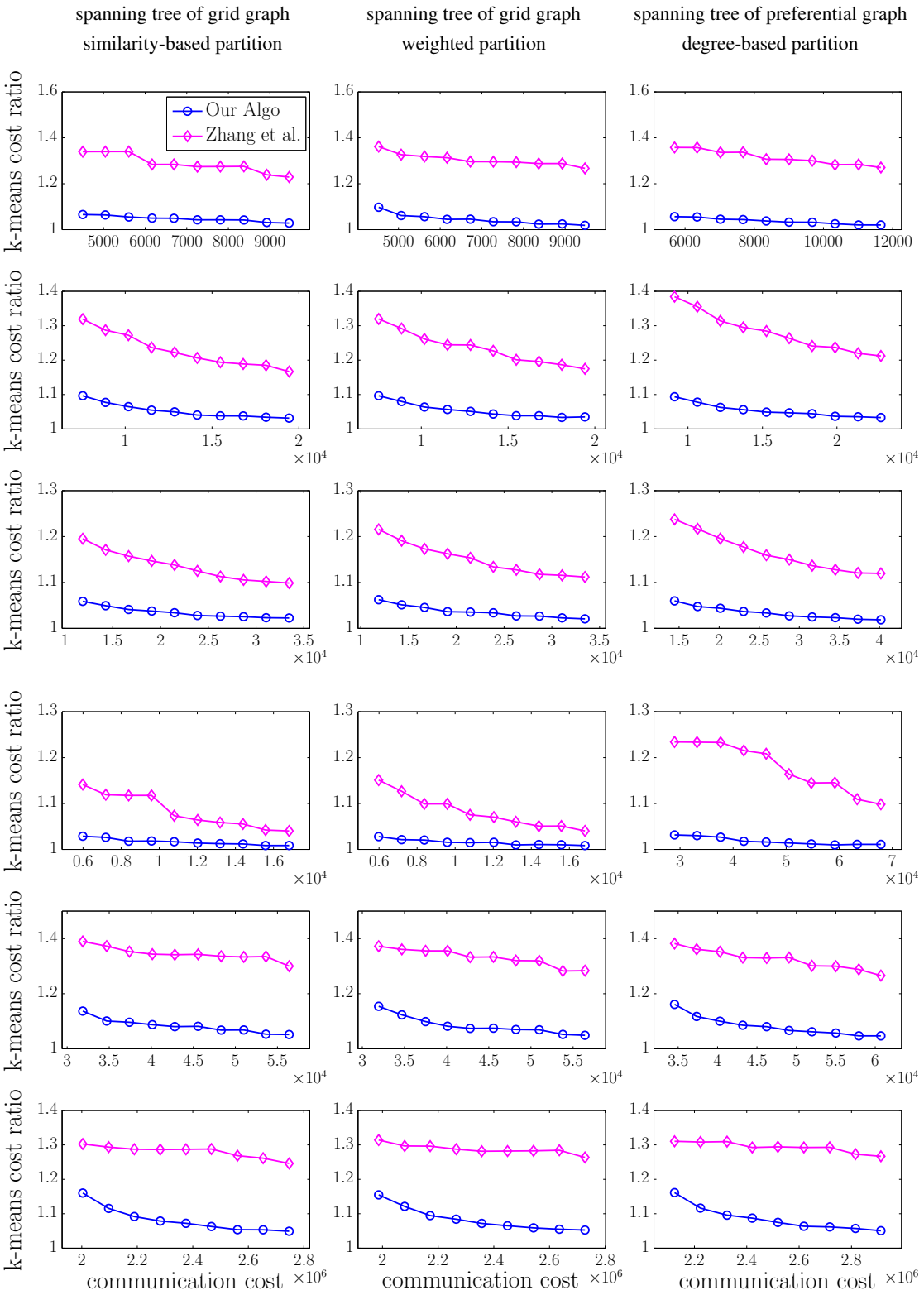


Figure 7: k-means cost on the spanning trees of the grid and preferential graphs. Columns: grid graph with similarity-based partition, grid graph with weighted partition, and preferential graph with degree-based partition. Rows: Spam, Pendigits, Letter, synthetic, ColorHistogram, and YearPredictionMSD.