Communication-Optimal Distributed Clustering

Abstract

Clustering large datasets is a fundamental problem with a number of applications in machine learning. Data is often collected on different sites and clustering needs to be performed in a distributed manner with low communication. We would like the quality of the clustering in the distributed setting to match that in the centralized setting for which all the data resides on a single server. In this work, we study both graph and geometric clustering problems in two distributed models: (1) a point-to-point model, and (2) a model with a broadcast channel. We give protocols in both models which we show are nearly optimal by proving almost matching communication lower bounds. Our work highlights the surprising power of a broadcast channel for clustering problems; roughly speaking, to cluster $n$ points or $n$ vertices in a graph distributed across $s$ servers, for a worst-case partitioning the communication complexity in a point-to-point model is $n \cdot s$, while in the broadcast model it is $n + s$. We implement our algorithms and demonstrate this phenomenon on real life datasets, showing that our algorithms are also very efficient in practice.

1 Introduction

Clustering is a fundamental task in machine learning with widespread applications in data mining, computer vision, and social network analysis. Example applications of clustering include grouping similar webpages by search engines, finding users with common interests in a social network, and identifying different objects in a picture or video. For these applications, one can model the objects that need to be clustered as points in Euclidean space $\mathbb{R}^d$, where the similarities of two objects are represented by the Euclidean distance between the two points. Then the task of clustering is to choose $k$ points as centers, so that the total distance between all input points to their corresponding closest center is minimized. Depending on different distance objective functions, three typical problems have been studied: $k$-means, $k$-median, and $k$-center.

The other popular approach for clustering is to model the input data as vertices of a graph, and the similarity between two objects is represented by the weight of the edge connecting the corresponding vertices. For this scenario, one is asked to partition the vertices into clusters so that the “highly connected” vertices belong to the same cluster. A widely-used approach for graph clustering is spectral clustering, which embeds the vertices of a graph into the points in $\mathbb{R}^k$ through the bottom $k$ eigenvectors of the graph’s Laplacian matrix, and applies $k$-means on the embedded points.

Both the spectral clustering and the geometric clustering algorithms mentioned above have been widely used in practice, and have been the subject of extensive theoretical and experimental studies over the decades. However, these algorithms are designed for the centralized setting, and are not applicable in the setting of large-scale datasets that are maintained remotely by different sites. In particular, collecting the information from all the remote sites and performing a centralized clustering algorithm is infeasible due to high communication costs, and new distributed clustering algorithms with low communication cost need to be developed.

There are several natural communication models, and we focus on two of them: (1) a point-to-point model, and (2) a model with a broadcast channel. In the former, sometimes referred to as the message-passing model, there is a communication channel between each pair of users. This may be impractical,
and the so-called coordinator model can often be used in place; in the coordinator model there is a
centralized server called the coordinator, and all communication goes through the coordinator. This
affects the total communication by a factor of two, since the coordinator can forward a message from
one server to another and therefore simulate a point-to-point protocol. There is also an additional
additive $O(\log s)$ bits per message, where $s$ is the number of servers, since a server must specify to
the coordinator where to forward its message. In the model with a broadcast channel, sometimes
referred to as the blackboard model, the coordinator has the power to send a single message which is
received by all $s$ servers at once. This can be viewed as a model for single-hop wireless networks.

In both models we study the total number of bits communicated among all servers. Although the
blackboard model is at least as powerful as the message-passing model, it is often unclear how to
exploit its power to obtain better bounds for specific problems. Also, for a number of problems the
communication complexity is the same in both models, such as computing the sum of $s$ length-$n$ bit
vectors modulo two, where each server holds one bit vector [20], or estimating large moments [22].
Still, for other problems like set disjointness it can save a factor of $s$ in the communication [5].

Our Contributions. We present algorithms for graph clustering: for any $n$-vertex graph whose
edges are arbitrarily partitioned across $s$ sites, our algorithms have communication cost $\tilde{O}(ns)$ in the
message passing model, and have communication cost $\tilde{O}(n+s)$ in the blackboard model, where
the $\tilde{O}$ notation suppresses polylogarithmic factors. The algorithm in the message passing model
has each server send a spectral sparsifier of its local data to the coordinator, who then merges
them in order to obtain a spectral sparsifier of the union of the datasets, which is sufficient for
solving the graph clustering problem. Our algorithm in the blackboard model is technically more
involved, as we show a particular recursive sampling procedure for building a spectral sparsifier
can be efficiently implemented using a broadcast channel. It is unclear if other natural ways of
building spectral sparsifiers can be implemented with low communication in the blackboard model.
Our algorithms demonstrate the surprising power of the blackboard model for clustering problems.
Since our algorithms compute sparsifiers, they also have applications to solving symmetric diagonally
dominant linear systems in a distributed model. Any such system can be converted into a system
involving a Laplacian (see, e.g., [1]), from which a spectral sparsifier serves as a good preconditioner.

Next we show that $\Omega(n^2s)$ bits of communication is necessary in the message passing model to even
recover a constant fraction of a cluster, and $\Omega(n+s)$ bits of communication is necessary in the
blackboard model. This shows the optimality of our algorithms up to poly-logarithmic factors.

We then study clustering problems in constant-dimensional Euclidean space. We show for any $c > 1$,
computing a $c$-approximation for $k$-median, $k$-means, or $k$-center correctly with constant probability
in the message passing model requires $\Omega(sk)$ bits of communication. We then strengthen this lower
bound, and show even for bicriteria clustering algorithms, which may output a constant factor more
clusters and a constant factor approximation, our $\Omega(sk)$ bit lower bound still holds. Our proofs are
based on communication and information complexity. Our results imply that existing algorithms [3]
for $k$-median and $k$-means with $\tilde{O}(sk)$ bits of communication, as well as the folklore parallel guessing
algorithm for $k$-center with $\tilde{O}(sk)$ bits of communication, are optimal up to poly-logarithmic factors.
For the blackboard model, we present an algorithm for $k$-median and $k$-means that achieves an
$O(1)$-approximation using $\tilde{O}(s+k)$ bits of communication. This again separates the models.

We give empirical results which show that using spectral sparsifiers preserves the quality of spectral
clustering surprisingly well in real-world datasets. For example, when we partition a graph with over
70 million edges (the Sculpture dataset) into 30 sites, only 6% of the input edges are communicated in
the blackboard model and 8% are communicated in the message passing model, while the values
of the normalized cut (the objective function of spectral clustering) given in those two models are
at most 2% larger than the one given by the centralized algorithm, and the visualized results are
almost identical. This is strong evidence that spectral sparsifiers can be a powerful tool in practical,
distributed computation. When the number of sites is large, the blackboard model incurs significantly
less communication than the message passing model, e.g., in the Two moons dataset when there are
90 sites, the message passing model communicates 9 times as many edges as communicated in the
blackboard model, illustrating the strong separation between these models that our theory predicts.

Related Work. There is a rich literature on spectral and geometric clustering algorithms from various
aspects (see, e.g., [2] [18] [19] [21]). Balcan et al. [3] [4] and Feldman et al. [10] study distributed
$\tilde{O}(k)$ also studies $k$-median), and present provable guarantees on the clustering quality.
Cohen et al. [7] study dimensionality reduction techniques for the input data matrices that can be used for distributed $k$-means. The main takeaway from previous work is that there is no previous work which develops protocols for spectral clustering in the common message passing and blackboard models, and lower bounds are lacking as well. For geometric clustering, while upper bounds exist (e.g., [3, 4, 10]), no provable lower bounds in either model existed, and our main contribution is to show that previous algorithms are optimal. We also develop a new protocol in the blackboard model.

2 Preliminaries

Let $G = (V, E, w)$ be an undirected graph with $n$ vertices, $m$ edges, and weight function $V \times V \to \mathbb{R}_{\geq 0}$, where every edge’s weight is bounded by a polynomial of $n$. The set of neighbors of a vertex $u$ is represented by $N(u)$, and its degree is $d_u = \sum_{v \sim u} w(u, v)$. The maximum degree of $G$ is defined to be $\Delta(G) = \max_v \{d_v\}$. For any set $S \subseteq V$, let $\mu(S) = \sum_{u \in S} d_u$. For any sets $S, T \subseteq V$, we define $w(S, T) = \sum_{u \in S, v \in T} w(u, v)$ to be the total weight of edges crossing $S$ and $T$. For two sets $X$ and $Y$, the symmetric difference of $X$ and $Y$ is defined as $X \triangle Y = (X \setminus Y) \cup (Y \setminus X)$.

For any matrix $A \in \mathbb{R}^{n \times n}$, let $\lambda_1(A) \leq \cdots \leq \lambda_n(A)$ be the eigenvalues of $A$, and let $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ be the maximum and minimum eigenvalues of $A$. For any two matrices $A, B \in \mathbb{R}^{n \times n}$, we write $A \preceq B$ to represent $B - A$ is positive semi-definite (PSD). Notice that this condition implies that $x^T A x \leq x^T B x$ for any $x \in \mathbb{R}^n$. Sometimes we also use a weaker notation $(1 - \epsilon)A \preceq B \preceq (1 + \epsilon)A$ to indicate that $(1 - \epsilon)x^T A x \leq x^T B x \leq (1 + \epsilon)x^T A x$ for all $x$ in the row span of $A$.

**Graph Laplacian.** The Laplacian matrix of $G$ is an $n \times n$ matrix $L_G$ defined by $L_G = D_G - A_G$, where $A_G$ is the adjacency matrix of $G$ defined by $A_G(u, v) = w(u, v)$, and $D_G$ is the $n \times n$ diagonal matrix with $D_{G,u,u} = \sum_{v \sim u} w(u, v)$ for any $u \in V(G)$. Alternatively, we can write $L_G$ with respect to a signed edge-vertex incidence matrix: we assign every edge $e = \{u, v\}$ an arbitrary orientation, and let $B_G(e, u) = 1$ if $v$ is $e$’s head, $B_G(e, v) = -1$ if $v$ is $e$’s tail, and $B_G(e, v) = 0$ otherwise. We further define a diagonal matrix $W_G \in \mathbb{R}^{n \times n}$, where $W_G(e, u) = w_e$. Then, we can write $L_G$ as $L_G = B_G^T W_G B_G$. The normalized Laplacian matrix of $G$ is defined by $L_G = D^{-1/2}_G L_G D^{-1/2}_G = I - D^{-1/2}_G A_G D^{-1/2}_G$. We sometimes drop the subscript $G$ when the underlying graph is clear from the context.

**Spectral sparsification.** For any graph $G = (V, E, w)$ with $n$ vertices, we call a subgraph $H$ of $G$ with proper reweighting of the edges is a $(1 + \epsilon)$-spectral sparsifier if

$$
(1 - \epsilon)L_G \preceq L_H \preceq (1 + \epsilon)L_G,
$$

By definition, it is easy to show that, if we decompose the edge set of a graph $G = (V, E)$ into $E_1, \ldots, E_\ell$ for a constant $\ell$ and $H_i$ is a spectral sparsifier of $G_i = (V, E_i)$ for any $1 \leq i \leq \ell$, then the graph formed by the union of edge sets from $H_i$ is a spectral sparsifier of $G$. It is known that, for any undirected graph $G$ of $n$ vertices, there is a $(1 + \epsilon)$-spectral sparsifier of $G$ with $O(n/\epsilon^2)$ edges, and it can be constructed in near-linear time [14]. We will show that a spectral sparsifier preserves the cluster structure of a graph.

**Models of computation.** We will study distributed clustering in two models for distributed data: the message passing model and the blackboard model. The message passing model represents those distributed computation systems with point-to-point communication, and the blackboard model represents those where messages can be broadcasted to all parties.

More precisely, in the message passing model there are $s$ sites $P_1, \ldots, P_s$, and one coordinator. These sites can talk to the coordinator through a two-way private channel. In fact, this is referred to as the coordinator model in Section [1] where it is shown to be equivalent to the point-to-point model up to small factors. The input is initially distributed at the $s$ sites. The computation is in terms of rounds: at the beginning of each round, the coordinator sends a message to some of the $s$ sites, and then each of those sites that have been contacted by the coordinator sends a message back to the coordinator. At the end, the coordinator outputs the answer. In the alternative blackboard model, the coordinator is simply a blackboard where these $s$ sites $P_1, \ldots, P_s$ can share information; in other words, if one site sends a message to the coordinator/blackboard then all the other $s - 1$ sites can see this information without further communication. The order for the sites to speak is decided by the contents of the blackboard.
For both models we measure the communication cost as the total number of bits sent through the channels. The two models are now standard in multiparty communication complexity (see, e.g., [5, 20, 22]). They are similar to the congested clique model [15] studied in the distributed computing community; the main difference is that in our models we do not post any bandwidth limitations at each channel but instead consider the total number of bits communicated.

3 Distributed graph clustering

In this section we study distributed graph clustering. We assume that the input graph $G = (V, E)$ has $n$ vertices, and consists of $k$ clusters. For any set $S \subseteq V$ with $\mu(S) \leq \mu(V)/2$, we define the conductance of set $S$ by $\phi_G(S) \triangleq w(S, V \setminus S)/\mu(S)$, and the $k$-way expansion constant of graph $G$ is defined by $\rho(k) \triangleq \min_{\text{partition } A_1, \ldots, A_k} \max_{1 \leq i \leq k} \phi_G(A_i)$. Informally, we call $S$ a cluster if vertices in $S$ are highly connected to each other, and there are fewer edges between $S$ and $V \setminus S$, i.e., set $S$ has low conductance $\phi_G(S)$. Moreover, we say $G$ has $k$ clusters if we can partition $V$ into $S_1, \ldots, S_k$ such that every $S_i$ has low conductance $\phi_G(S_i)$, i.e., $G$ has a small value of $\rho(k)$. While it is NP-hard to compute $\rho(k)$ and a partition $S_1, \ldots, S_k$ achieving $\rho(k)$, it is known that $\rho(k)$ is closely related to $\lambda_k(L_G)$ through the following higher-order Cheeger inequality [13]:

$$\frac{\lambda_k(L_G)}{2} \leq \rho(k) \leq O(k^2) \sqrt{\lambda_k(L_G)}.$$  

Based on (2), a large gap between $\lambda_{k+1}(L_G)$ and $\rho(k)$ implies (i) existence of a $k$-way partition $\{S_i\}_{i=1}^k$ with bounded $\phi_G(S_i) \leq \rho(k)$, and (ii) any $(k+1)$-way partition of $G$ contains a subset with significantly high conductance $\rho(k+1) \geq \lambda_{k+1}(L_G)/2$. That is, a large gap between $\lambda_{k+1}(L_G)$ and $\rho(k)$ implies that $G$ has exactly $k$ clusters. In the following, we assume that $G$ satisfies $\Upsilon \triangleq \lambda_{k+1}(L_G)/\rho(k) = \Omega(k^3)$, and this condition guarantees that $G$ has exactly $k$ clusters. We use $S_1, \ldots, S_k$ to express a partition that achieves $\rho(k)$. We remark that the same assumption has been used in the literature for studying graph clustering problems [19].

Both algorithms presented in the section are based on the following spectral clustering algorithm: (i) compute the bottom $k$ eigenvectors $f_1, \ldots, f_k$ of $L_G$; (ii) embed every vertex $v$ to a point in $\mathbb{R}^k$ through the embedding $F(v) = (f_1(v), \ldots, f_k(v))$, with an appropriate normalization, and run $k$-means on the embedded points $\{F(v)\}_{v \in V}$.

3.1 The message passing model

We assume that the input graph $G = (V, E)$ has $n$ vertices and $m$ edges, and these $m$ edges are arbitrarily allocated among $s$ sites $P_1, \ldots, P_s$. We further use $E_i$ to denote the edge set maintained by site $P_i$. Our proposed algorithm consists of two steps: (i) every site $P_i$ computes a linear-sized $(1 + \Theta(1))$-spectral sparsifier $H_i = (V, E_i)$ of $G_i = (V, E_i)$, and sends $H_i$ to the coordinator; (ii) the coordinator runs a spectral clustering algorithm on the union of received graphs $H = (V, \bigcup_{i=1}^s E_i)$. The following theorem summarizes the performance of this algorithm, and the approximation guarantee of our algorithm is as good as the provable guarantee of spectral clustering known in the non-distributed setting.

**Theorem 3.1.** Let $G = (V, E)$ be an $n$-vertex graph that satisfies $\Upsilon = \Omega(k^3)$, and the edges of $G$ are arbitrarily allocated among $s$ sites. Then, there is a distributed algorithm that returns a partition $A_1, \ldots, A_k$ satisfying $\text{vol}(A_i \Delta S_i) = O(k^3 \cdot \Upsilon^{-1} \cdot \text{vol}(S_i))$ for any $1 \leq i \leq k$. The total communication cost of this algorithm is $O(n s)$ words.

**Proof.** By the definition of the Laplacian matrix, we have that $L_G = \sum_{i=1}^s L_{G_i}$. Since every $H_i$ is a $(1 + \Theta(1))$-spectral sparsifier of $G_i$, we have that $(1 - \Theta(1))L_{H_i} \preceq L_{G_i} \preceq (1 + \Theta(1))L_{H_i}$. This implies that $(1 - \Theta(1))L_H \preceq L_G \preceq (1 + \Theta(1))L_H$. By the definition of $H_i$ and graph Laplacians. Now we show that the our assumption of $\Upsilon$ preserves in $H$. By Lemma B.1 in Appendix B.1 we have for any $1 \leq i \leq k$ that $\phi_H(S_i) \in (\frac{1}{2}, \frac{1}{2}) \phi_G(S_i)$, which implies that $S_i$ has low conductance in $H$, and $\rho_H(k) \in (\frac{1}{2}, \frac{1}{2}) \rho_G(k)$. To show that $\lambda_k(L_H)$ is a constant approximation of $\lambda_k(L_G)$, notice that $(1 - \Theta(1)) \cdot x^T L_G x \preceq x^T L_H x \preceq (1 + \Theta(1)) \cdot x^T L_G x$ holds for any $x \in \mathbb{R}^n$. Hence it holds for any $x \in \mathbb{R}^n$ that $(1 - \epsilon) \cdot x^T D_G^{-1/2} L_G D_G^{-1/2} x \leq x^T D_G^{-1/2} L_H D_G^{-1/2} x \leq (1 + \epsilon) \cdot x^T D_G^{-1/2} L_G D_G^{-1/2} x$.
Since \( D_G^{-1/2} L_G D_G^{-1/2} = L_G \) and \( \frac{1}{2} D_G^{-1} \preceq D_H^{-1} \preceq 2 D_G^{-1} \), we have that \( \lambda_i(L_H) = \Theta(\lambda_i(L_G)) \), and the assumption of \( \Upsilon \) in \( H \) is preserved from \( G \) up to a constant factor. By Lemma B.2 in Appendix B.1 the output of a spectral clustering algorithm on \( H \) satisfies the claimed properties. The total communication cost of \( O(ns) \) bits follows from the fact that every \( H_i \) has \( O(n) \) edges.

Our proposed algorithm is very easy to implement, and the next theorem shows that the communication cost of our algorithm is optimal up to a logarithmic factor. The proof is given in Appendix B.2.

**Theorem 3.2.** Let \( G \) be an undirected graph with \( n \) vertices, and the edges of \( G \) are distributed among \( s \) sites. Then, any algorithm that correctly outputs a constant fraction of a cluster in \( G \) requires \( \Omega(ns) \) bits of communication cost. This holds even if each cluster has constant expansion.

As a remark, it is easy to see that this lower bound also holds for constructing spectral sparsifiers: for any \( n \times n \) PSD matrix \( A \) whose entries are arbitrarily distributed among \( s \) sites, any distributed algorithm that constructs a \((1 + \Theta(1))\)-spectral sparsifier of \( A \) requires \( \Omega(ns) \) bits of communication.

This follows since such a spectral sparsifier can be used to solve the spectral clustering problem. Spectral sparsification has played an important role in designing fast algorithms from different areas, e.g. machine learning, and numerical linear algebra. Hence our lower bound result for constructing spectral sparsifiers may have applications in studying other distributed learning algorithms.

### 3.2 The blackboard model

Next we present a graph clustering algorithm with \( \widetilde{O}(n + s) \) bits of communication cost in the blackboard model. We already saw from the proof of Theorem 3.1 that a spectral sparsifier preserves the blackboard model. We already saw from the proof of Theorem 3.1 that a spectral sparsifier preserves the cluster structure of a graph, so it suffices to present a distributed algorithm for constructing a spectral sparsifier in the blackboard model.

Our distributed algorithm for constructing a spectral sparsifier is based on constructing a chain of coarse sparsifiers [17], which is described as follows: for any input PSD matrix \( K \) with \( \lambda_{\text{max}}(K) \leq \lambda_u \) and all the non-zero eigenvalues of \( K \) at least \( \lambda_{\ell} \), we construct a chain of matrices

\[
[K(0), K(1), \ldots, K(d), K]
\]

of length \( d = \lceil \log_2(\lambda_u/\lambda_{\ell}) \rceil \), where \( K(i) = K + \gamma(i)I \), and \( \gamma(i) = \lambda_u/2^i \). Notice that in the chain every \( K(i - 1) \) is obtained by adding weights to the diagonal entries of \( K(i) \), and \( K(i - 1) \) approximates \( K(i) \) as long as the weights added to the diagonal entries are small. Such a chain can be constructed recursively so that \( K(0) \) has heavy diagonal entries, and can be approximated by a diagonal matrix. Moreover, it is easy to prove that \( d = O(\log n) \) if \( K \) is the Laplacian matrix of a graph \( G \), provided \( G \)'s edge weights are polynomially bounded.

**Lemma 3.3 ([17]).** The chain \([3]\) satisfies the following relations: (1) \( K \preceq_r K(d) \preceq_r 2K \); (2) \( K(\ell) \preceq (K(\ell - 1) - 2K(\ell)) \) for all \( \ell \in \{1, \ldots, d\} \); (3) \( K(0) \preceq 2\gamma(0)I \preceq 2K(0) \).

Based on Lemma 3.3 to construct a spectral sparsifier of \( K \) we build a chain of matrices

\[
[\widetilde{K}(0), \widetilde{K}(1), \ldots, \widetilde{K}(d)]
\]

such that every \( \widetilde{K}(\ell + 1) \) can be constructed from \( \widetilde{K}(\ell) \), and every \( \widetilde{K}(\ell) \) is a spectral sparsifier of \( K(\ell) \). It is well known that, for \( K(\ell) \preceq B^T B \), sampling rows \( b_i \) of \( B \) according to their leverage scores, denoted by \( \tau_i \mathrel{\triangleq} b_i^T K^+ b_i \), will give a matrix approximating \( K \). Formally, we assume that \( \tau \) is the vector of leverage score overestimate for \( B \)'s rows such that \( \tau_i \geq \tau \) for all \( i \in [m] \), \( 0 < \varepsilon < 1 \), and \( c \) is a fixed constant. We sample every row \( b_i \) with probability \( p_i = \min\{1, c e^{-2\tau} \log n\} \), and define a diagonal matrix \( W \) with \( W_{i,i} = \frac{1}{p_i} \), with probability \( p_i \), and \( W_{i,i} = 0 \) otherwise. Then, it holds with high probability that \( (1 - \varepsilon)K(\ell) \preceq \widetilde{K}(\ell) \preceq B^T W B \preceq (1 + \varepsilon)K(\ell) \). Moreover, \( W \) has \( O(\|\tau\|_1 e^{-2} \log n) \) non-zeros with high probability. We prove in Appendix B.3 that the above sampling procedure can be implemented in the blackboard model, and this gives the following result.

**Theorem 3.4.** Let \( G \) be an undirected graph of \( n \) vertices, where edges of \( G \) are allocated in \( s \) sites. Then, a spectral sparsifier of \( G \) can be constructed with total communication cost \( \widetilde{O}(n + s) \) in the blackboard model. That is, the chain \([4]\) can be constructed with communication cost \( \widetilde{O}(n + s) \) in the blackboard model.
Combining Theorem 3.4 and the proof of Theorem 3.1, we obtain a distributed algorithm in the blackboard model with total communication cost $\tilde{O}(n + s)$, and the performance of our algorithm is the same as the statement of Theorem 3.1. Notice that $\Omega(n + s)$ bits of communication are needed for graph clustering in the blackboard model, since the output of a clustering algorithm contains $\Omega(n)$ bits of information and each server needs to communicate at least one bit. Hence the communication cost of our proposed algorithm is optimal up to a poly-logarithmic factor.

4 Distributed geometric clustering

We now consider geometric clustering, including $k$-median, $k$-means and $k$-center. Let $P$ be a set of points of size $n$ in a metric space with distance function $d(\cdot,\cdot)$, and let $k \leq n$ be an integer. In $k$-center we want to find a set $C'(|C| = k)$ such that $\max_{p \in P} d(p, C')$ is minimized, where $d(p, C) = \min_{c \in C} d(p, c)$. In $k$-median and $k$-means we replace the objective function $\max_{p \in P} d(p, C)$ with $\sum_{p \in P} d(p, C)$ and $\sum_{p \in P} (d(p, C))^2$ respectively.

4.1 The message passing model

As mentioned, for constant dimensional Euclidean space and a constant $c > 1$, there are algorithms that $c$-approximate $k$-median and $k$-means using $\tilde{O}(sk)$ bits of communication [4]. For $k$-center, the folklore parallel guessing algorithms (see, e.g., [9]) achieve a 2.01 approximation using $\tilde{O}(sk)$ bits of communication.

Our proof is based on the following Multiparty Set-Disjointness problem (DISJ$_{s,n}$): for any $s$ sites $P_1, \ldots, P_s$, where each $P_i$ has a set $S_i \subseteq [n]$, let $X_i = (X_{i1}, \ldots, X_{in})$ be the characteristic vector of $S_i$, and let $X = (X_1, \ldots, X_s)$ be the input matrix with $X_i$ being the $i$-th row. Let $X_j = (X_{j1}, \ldots, X_{jn})$ be the $j$-th column of the input matrix $X$. We define a function $\text{ALLONE}_s$ on an $s$-bit vector $Y = (Y_1, \ldots, Y_s)$ as $\text{ALLONE}_s(Y) = \bigwedge_{i \in [s]} Y_i$, and $\text{DISJ}_{s,n}(X) = \bigvee_{j \in [n]} \text{ALLONE}_s(X_j)$. Then the DISJ$_{s,n}$ problem asks for the value of DISJ$_{s,n}(X)$.

We now show that these bounds are tight up to logarithmic factors.

**Theorem 4.1.** For any $c > 1$, computing $c$-approximation for $k$-median, $k$-means or $k$-center correctly with probability $0.99$ in the message passing model needs $\Omega(sk)$ bits of communication.

**Proof.** We prove a more general results: the $\Omega(sk)$ lower bound holds for any eligible function which evaluates 0 if there are at most $k$ points, and evaluates greater than 0 if there are at least $k + 1$ points. Note that $k$-median, $k$-means and $k$-center are all eligible functions. We prove this by a simple reduction from DISJ$_{s,\ell}$ where $\ell = (k + 1)/2$ (w.l.o.g., assuming $k$ is odd). The reduction is as follows. Given an $s$-player set-disjointness instance of size $\ell$ (i.e., DISJ$_{s,\ell}$), let $X_i = (X_{i1}, \ldots, X_{i\ell})$ be the $i$-th row of the input matrix $X$. Let $p^1, \ldots, p^\ell$ and $q^1, \ldots, q^\ell$ be $2\ell$ distinct point locations on a line. Each site $i$ does the following: for each coordinate $j$, if $X_{ij} = 0$ then it put a point $u^j_i$ at location $q^j$; otherwise if $X_{ij} = 1$ it put a point at location $p^j$. It is easy to see that DISJ$_{s,\ell} = 1$ iff the number of distinct points in $\bigcup_{i \in [s], j \in [\ell]} u^j_i$ is $2(\ell - 1) + 1 = k$; and DISJ$_{s,\ell} = 1$ iff the number of distinct points in $\bigcup_{i \in [s], j \in [\ell]} u^j_i$ is $2(\ell - 1) + 2 = k + 1$. The lower bound follows from the definition of eligible function and Theorem B.3 in Appendix B.2.

A number of works on clustering consider bicriteria solutions (e.g., [12, 6]). An algorithm is an $(c_1, c_2)$-approximation ($c_1, c_2 > 1$) if the optimal solution costs $W$ when using $k$ centers, then the output of the algorithm costs at most $c_1 W$ when using at most $c_2 k$ centers. We can show that for $k$-median and $k$-means, the $\Omega(sk)$ lower bounds even for algorithms with bicriteria approximations. The proof of the following theorem can be found in Appendix C.1. In this proof we use a finer notion of communication complexity called information complexity; see Appendix A for details.

**Theorem 4.2.** For any $c \in [1, 1.01]$, computing $(7.1 - 6c, c)$-bicriteria-approximation for $k$-median or $k$-means correctly with probability $0.99$ in the message passing model needs $\tilde{O}(sk)$ bits of communication.
4.2 The blackboard model

In Appendix C.2 we show that there is an algorithm that achieves an $O(1)$-approximation using $O(s + k)$ bits of communication for $k$-median and $k$-means. For $k$-center, it is straightforward to implement the parallel guessing algorithm in the blackboard model using $O(s + k)$ bits of communication.

Theorem 4.3. There are algorithms that compute $O(1)$-approximations for $k$-median, $k$-means and $k$-center correctly with probability 0.9 in the blackboard model using $O(sk)$ bits of communication.

5 Experiments

In this section we present experimental results for spectral graph clustering in the message passing and blackboard models. We will compare the following three algorithms. (1) Baseline: each site sends all the data to the coordinator directly; (2) MsgPassing: our algorithm in the message passing model (Section 3.1); (3) Blackboard: our algorithm in the the blackboard model (Section 3.2).

Besides giving the visualized results of these algorithms on various datasets, we also measure the qualities of the results via the normalized cut, defined as

$$\text{ncut}(A_1, \ldots, A_s) = \frac{1}{2} \sum_{i \in [s]} \frac{W(A_i, V \setminus A_i)}{\text{vol}(A_i)},$$

which is the objective function that the original spectral clustering algorithm wants to minimize.

We implemented the algorithms using multiple languages, including Matlab, Python and C++. Our experiments were conducted on an IBM NeXtScale nx360 M4 server, which is equipped with 2 Intel Xeon E5-2652 v2 8-core processors, 32GB RAM and 250GB local storage.

Datasets. We test the algorithms in the following real and synthetic datasets, whose visualizations are presented in Figure 4 in Appendix D.1.

- Twomoons: this dataset contains $n = 14,000$ coordinates in $\mathbb{R}^2$. We consider each point as a vertex. For any two vertices $u, v$, we add an edge with weight $w_{uv} = \exp\left(-\frac{1}{2}\frac{\|u - v\|^2}{\sigma^2}\right)$ with $\sigma = 0.1$ when one vertex is among the 7000-nearest points of the other. This construction results in a graph with about 110,000,000 edges.

- Gauss: this dataset contains $n = 10,000$ points in $\mathbb{R}^2$. There are 4 clusters in this dataset, each generated using a Gaussian distribution. We construct a complete graph as the similarity graph. For any two vertices $u, v$, we define the weight $w_{uv} = \exp\left(-\frac{1}{2}\frac{\|u - v\|^2}{\sigma^2}\right)$ with $\sigma = 1$. The resulting graph has about 100,000,000 edges.

- Sculpture: a photo of The Greek Slave. We use an $80 \times 150$ version of this photo where each pixel is viewed as a vertex. To construct a similarity graph, we map each pixel to a point in $\mathbb{R}^6$, i.e., $(x, y, r, g, b)$, where the latter three coordinates are the RGB values. For any two vertices $u, v$, we then put an edge between $u, v$ with weight $w_{uv} = \exp\left(-\frac{1}{2}\frac{\|u - v\|^2}{\sigma^2}\right)$ with $\sigma = 0.5$ if one of $u, v$ is among the 5000-nearest points of the other. This results in a graph with about 70,000,000 edges.

In the distributed model edges are randomly partitioned across $s$ sites.

Results on Clustering Quality. We visualize the clustered results for different datasets in Figure 1 and Figure 3 in Appendix D.1. It can be seen that Baseline, MsgPassing and Blackboard give results of very similar qualities. We only present the visualization for $s = 15$. Similar results were observed when we varied the values of $s$.

We also compare the normalized cut (ncut) values of the clustering results of different algorithms. The results are presented in Figure 2. In all datasets, the ncut values of different algorithms are very close. The ncut value of MsgPassing slightly decreases when we increase the value of $s$, while the ncut value of Blackboard is independent of $s$.

Results on Communication Costs. We compare the communication costs of different algorithms in Figure 5. We observe that while achieving similar clustering qualities as Baseline, both MsgPassing and Blackboard are significantly more communication-efficient (by one or two orders of magnitudes in our experiments). We also notice that the value of $s$ does not affect the communication cost of Blackboard, while the communication cost of MsgPassing grows almost linearly with

\footnote{Available in e.g.
http://artgallery.yale.edu/collections/objects/14794}
Twomoons, \( k = 2 \);

Figure 1: visualization of results on Twomoons; in the message passing model each site samples \( 5n \) edges; in the blackboard model all sites jointly sample \( 10n \) edges and the chain has length 18.

![Figure 1: visualization of results on Twomoons; in the message passing model each site samples 5n edges; in the blackboard model all sites jointly sample 10n edges and the chain has length 18.](image)

Figure 2: Comparisons on normalized cuts. In the message passing model, each site samples \( 5n \) edges; in each round of the algorithm in the blackboard model, all sites jointly sample \( 10n \) edges (in Twomoons and Gauss) or \( 20n \) edges (in Sculpture) edges and the chain has length 18.

\( s \); when \( s \) is large, \textit{MsgPassing} uses significantly more communication than \textit{Blackboard}. These confirm our theory. In Figure 6 and Figure 7 in Appendix D.3 we present how the performance of \textit{MsgPassing} and \textit{Blackboard} are affected by their parameters.

![Figure 2: Comparisons on normalized cuts. In the message passing model, each site samples 5n edges; in each round of the algorithm in the blackboard model, all sites jointly sample 10n edges (in Twomoons and Gauss) or 20n edges (in Sculpture) edges and the chain has length 18.](image)

Figure 3: Comparisons on communication costs. In the message passing model, each site samples \( 5n \) edges; in each round of the algorithm in the blackboard model, all sites jointly sample \( 10n \) (in Twomoons and Gauss) or \( 20n \) (in Sculpture) edges and the chain has length 18.

![Figure 3: Comparisons on communication costs. In the message passing model, each site samples 5n edges; in each round of the algorithm in the blackboard model, all sites jointly sample 10n (in Twomoons and Gauss) or 20n (in Sculpture) edges and the chain has length 18.](image)
References


A Omitted details from Section 2

In this section we list some omitted definitions that will be used in our analysis.

**Communication complexity.** For any problem \( \mathcal{A} \) and any protocol \( \Pi \) solving \( \mathcal{A} \), the communication complexity of a protocol \( \Pi \) is the maximum communication cost of \( \Pi \) over all possible inputs \( X \).

When the protocol is randomised, we define the error of \( \Pi \) by

\[
\max_x P(\text{the coordinator outputs an incorrect answer on } X),
\]

where the max is over all inputs \( X \) and the probability is over all random strings of the coordinator and \( s \) sites. The \( \delta \)-error randomised communication complexity \( R_\delta(\mathcal{A}) \) of a problem \( \mathcal{A} \) in the message passing model is the minimum communication complexity of any randomised protocol \( \Pi \) that solves \( \mathcal{A} \) with error at most \( \delta \).

Let \( \mu \) be an input distribution on \( X \). We call a deterministic protocol \( (\delta, \mu) \)-error if it gives the correct answer for \( \mathcal{A} \) on at least a \( 1 - \delta \) fraction of all input pairs, weighted by the distribution \( \mu \). We denote \( D_{\delta, \mu}(\mathcal{A}) \) as the cost of the minimum-communication \( (\delta, \mu) \)-error protocol. A standard lemma in communication complexity called Yao’s minimax lemma shows that \( R_\delta(\mathcal{A}) \geq \max_\mu D_{\delta, \mu}(\mathcal{A}) \).

**Information complexity.** We abuse notation by using \( \Pi \) for both the protocol and its transcript (its concatenation of messages). In the message passing model, let \( \Pi_i \) \( (i \in [s]) \) be the transcript (set of messages exchanged) between the \( i \)-th site and the coordinator. Then \( \Pi \) can be seen as a concatenation \( \Pi_1 \circ \Pi_2 \circ \ldots \circ \Pi_s \) ordered by the timestamps of the messages. We define the information complexity of a problem \( \mathcal{A} \) in the message passing model by

\[
\text{IC}_{\mu, \delta}(\mathcal{A}) = \min_{(\delta, \mu) \text{-error } \Pi} \sum_{i \in [s]} I(X_1, \ldots, X_s; \Pi_i).
\]

It has been shown in [11] that \( R_\delta(\mathcal{A}) \geq \text{IC}_{\delta, \mu}(\mathcal{A}) \) for any input distribution \( \mu \).

B Omitted details from Section 3

This section lists all omitted details from Section 3

B.1 Omitted details used in proving Theorem 3.1

Now we list all omitted details used in proving Theorem 3.1. We first show that a spectral sparsifier of \( G \) preserves the cluster structure.

**Lemma B.1.** Let \( H \) be a \( (1 + \varepsilon) \)-spectral sparsifier of \( G \) for some \( \varepsilon \leq 1/3 \). Then, it holds for any set \( S \subseteq V \) that \( \phi_H(S) \in \left( \frac{1}{2}, 2 \right) \phi_G(S) \).

**Proof.** Let \( x_u \in \mathbb{R}^n \) be the indicator vector of vertex \( u \), i.e., \( x_u(v) = 1 \) if \( u = v \), and \( x_u(v) = 0 \) otherwise. We have that

\[
(1 - \varepsilon) \cdot x_u^T L_G x_u \leq x_u^T L_H x_u \leq (1 + \varepsilon) \cdot x_u^T L_G x_u,
\]

which implies that \( (1 - \varepsilon) \cdot \text{vol}_G(S) \leq \text{vol}_H(S) \leq (1 + \varepsilon) \cdot \text{vol}_G(S) \) for any subset \( S \).

Similarly, for any set \( S \subseteq V \) we define the indicator vector of \( S \) by \( x_S \in \mathbb{R}^n \), where \( x_S(u) = 1 \) if \( u \in S \), and \( x_S(u) = 0 \) otherwise. Hence, \( x_S^T L_G x_S = w_G(S, V \setminus S) \), and \( x_S^T L_H x_S = w_H(S, V \setminus S) \). Combining these with [11], we have that

\[
(1 - \varepsilon) \cdot w_G(S, V \setminus S) \leq w_H(S, V \setminus S) \leq (1 + \varepsilon) \cdot w_G(S, V \setminus S).
\]

Hence, for any subset \( S \) we have that

\[
\phi_H(S) = \frac{w_H(S, V \setminus S)}{\text{vol}_H(S)} \leq \frac{(1 + \varepsilon) w_G(S, V \setminus S)}{(1 - \varepsilon) \text{vol}_G(S)} \leq 2 \cdot \phi_G(S),
\]

where the last inequality holds by assuming \( \varepsilon \leq 1/3 \). Similarly, we have that

\[
\phi_H(S) = \frac{w_H(S, V \setminus S)}{\text{vol}_H(S)} \geq \frac{(1 - \varepsilon) w_G(S, V \setminus S)}{(1 + \varepsilon) \text{vol}_G(S)} \geq \frac{1}{2} \cdot \phi_G(S).
\]

Hence, \( \phi_H(S_i) \) and \( \phi_G(S_i) \) differ by at most a factor of 2. \( \square \)
Our analysis on the performance of spectral clustering relies on the following theorem.

**Lemma B.2** ([19]). Let $G$ be a graph satisfying the condition $\Upsilon = \Omega(k^3)$, and $k \in \mathbb{N}$. Then, a spectral clustering algorithm outputs sets $A_1, \ldots, A_k$ such that $\text{vol}(A_i \triangle S_i) = O\left(k^3 \cdot \Upsilon^{-1} \cdot \text{vol}(S_i)\right)$ holds for any $1 \leq i \leq k$, where $S_i$ is the optimal cluster corresponding to $A_i$.

### B.2 Omitted details used in proving Theorem 3.2

Now we show a lower bound on the communication complexity of graph clustering in the message passing model. Our proof is based on a reduction from graph clustering to the Multiparty Set-Disjointness problem ($\text{DISJ}_{s,n}$): for any $s$ sites $P_1, \ldots, P_s$, where each $P_i$ has a set $S_i \subseteq [n]$, let $X_i = (X_i^1, \ldots, X_i^n)$ be the characteristic vector of $S_i$, and let $X = (X_1, \ldots, X_s)$ be the input matrix with $X_i$ being the $i$-th row. Let $X_j = (X_j^1, \ldots, X_j^n)$ be the $j$-th column of the input matrix $X$. We define a function $\text{ALLONE}_s$ on an $s$-bit vector $Y = (Y_1, \ldots, Y_s)$ as $\text{ALLONE}_s(Y) = \bigwedge_{i \in [s]} Y_i$, and $\text{DISJ}_{s,n}(X) = \bigvee_{j \in [n]} \text{ALLONE}_s(X_j)$. Then the $\text{DISJ}_{s,n}$ problem asks the value of $\text{DISJ}_{s,n}(X)$.

We introduce two hard input distributions for $\text{ALLONE}_s$ and $\text{DISJ}_{s,n}$ respectively.

1. **Hard input distribution $\nu$ on $Y \in \{0, 1\}^s$ for $\text{ALLONE}_s$:** with probability 1/2, we choose each $Y_i$ ($i \in [s]$) to be 0 or 1 with equal probability; with probability 1/4 we choose $Y$ to be an all-1 vector; and with the rest probability 1/4 we choose $Y$ to be a random vector with $n - 1$ coordinates being 1’s and a random coordinate being 0.

2. **Hard input distribution $\mu_n$ on $X \in \{0, 1\}^{s \times n}$ for $\text{DISJ}_{s,n}$:** For each $j \in [n]$, we choose $X^{j\cdot} \sim \nu$.

**Theorem B.3** ([5]). It holds that $\text{IC}_{0.49, \nu}(\text{ALLONE}_s) = \Omega(s)$, and $\text{IC}_{0.49, \nu}(\text{DISJ}_{s,n}) = \Omega(sn)$.

**Lemma B.4.** In the message passing model, any randomized algorithm that computes $\text{DISJ}_{s,n}$ correctly with probability 0.9 needs $\Omega(sn)$ bits of communication.

**Proof.** The lemma follows from Theorem B.3 and Yao’s minimax lemma.

Now we are ready to prove Theorem 3.2.

**Proof of Theorem 3.2.** Our proof is based on the reduction from graph clustering to the Multiparty Set-Disjointness problem ($\text{DISJ}_{s,n}$). For any item $j$ and site $P_i$, we set $X_i^j = 0$ if item $j$ appears in site $P_i$, and $X_i^j = 1$ otherwise. Then $\text{DISJ}_{s,n}(X) = 1$ if there is some item not appearing in any site. Now we construct a graph $G$ based on the hard instance $X$ of $\text{DISJ}_{s,n}$ as follows: initially, graph $G$ consists of $n$ isolated vertices $\ell_1, \ldots, \ell_n$, and $r$ isolated vertices $r_1, \ldots, r_s$. Then, we add an edge between $\ell_j$ and $r_i$ if item $j$ appears in site $P_i$. With this construction, it is easy to see that $\text{DISJ}_{s,n}(X) = 0$ if every vertex $\ell_j$ is connected to some $r_i$, and $\text{DISJ}_{s,n}(X) = 1$ if there are some isolated vertices $\ell_j$.

We will show that, when $\text{DISJ}_{s,n}(X) = 0$, our constructed graph $G$ is a bipartite expander, i.e., $G$ has only 1 cluster. To prove this, notice that, from the hard input distribution $\mu$ on $Y \in \{0, 1\}^s$ described above, with probability 1/2 we choose each $Y_i$ ($i \in [s]$) to be 0 or 1 with equal probability. This implies that, for any $\ell_i$ and $r_j$, there is an edge between $\ell_i$ and $r_j$ independently with probability at least $1/4$. By standard results on constructing expanders, this implies $G$ is a bipartite expander with constant expansion, and in particular is connected.

On the other side, when $\text{DISJ}_{s,n}(X) = 1$, every isolated vertex $\ell_j$ itself forms a cluster with conductance 0 and constant expansion, and the giant component of $G$ forms a cluster with conductance 0 and constant expansion (since, as argued in the previous paragraph, it is a bipartite expander). Let $k$ be the number of connected components in graph $G$. Then, $\rho(k) = 0$, and our assumption on $\Upsilon = \lambda_{k+1}(L_G)/\rho(k) = \Omega(k^3)$ holds trivially. Hence any clustering algorithm that is able to find a constant fraction of each cluster in graph $G$ satisfying $\Upsilon = \Omega(k^3)$ can be used to solve $\text{DISJ}_{s,n}$, the lower bound of communication complexity of graph clustering follows from the lower bound for $\text{DISJ}_{s,n}$.
B.3 Proof of Theorem 3.4

Proof of Theorem 3.4 Let $K = B^T B$ be the Laplacian matrix of the underlying graph $G$, where $B \in \mathbb{R}^{n \times n}$ is the edge vertex incident matrix of $G$. We will prove that every $\tilde{K}(d)$ can be constructed based on $K(i)$ with communication cost $\tilde{O}(n + s)$. This implies that $\tilde{K}(d)$, a $(1 + \varepsilon)$-spectral sparsifier of $K$, can be constructed with communication cost $\tilde{O}(n + s)$, as the length of the chain $d = O(\log n)$.

First of all, notice that $\gamma_u \leq 2n$, and the value of $n$ can be obtained with communication cost $\tilde{O}(n + s)$ (different sites sequentially write the new IDs of the vertices on the blackboard). For simplicity, in the following we assume that $\gamma_u$ is the upper bound of $\lambda_{max}$ that we actually obtained in the blackboard model.

Base case of $\ell = 0$: By definition, $K(0) = K + \lambda_u \cdot I$, and

$$\frac{1}{2} \cdot K(0) \preceq \gamma(0) \cdot I \preceq K(0),$$

due to Statement 3 of Lemma 3.3. Letting $\oplus$ denote appending the rows of one matrix to another, we define $B_{\gamma(0)} = B \oplus \sqrt{\gamma(0)} \cdot I$, and write $K(0) = K + \gamma(0) \cdot I = B_{\gamma(0)}^T B_{\gamma(0)}$. Since $\tau_i = b_i^T (K(0))^T b_i$ for each row of $B_{\gamma(0)}$, we have

$$\tau_i \preceq b_i^T (\gamma(0) \cdot I) b_i \leq 2 \cdot \tau_i.$$ 

Let $\bar{\tau}_i = b_i^T (\gamma(0) \cdot I)^+ b_i$ be the leverage score of $b_i$ approximated using $\gamma(0) \cdot I$, and let $\bar{\tau}$ be the vector of approximate leverage score, with the leverage scores of the $n$ rows corresponding to $\sqrt{\gamma(0)} \cdot I$ rounded up to 1. Then, with high probability sampling $O(\varepsilon^{-2} n \log n)$ rows of $B$ will give a matrix $\tilde{K}(0)$ such that

$$(1 - \varepsilon) K(0) \preceq \tilde{K}(0) \preceq (1 + \varepsilon) K(0),$$

Notice that, as every row of $B$ corresponds to an edge of $G$, the approximate leverage scores $\bar{\tau}_i$ for different edges can be computed locally by different sites maintaining the edges, and the sites only need to send the information of the sampled edges to the coordinator, hence the communication cost is $\tilde{O}(n + s)$.

Induction step: We assume that

$$(1 - \varepsilon) K(\ell) \preceq_{r_1} \tilde{K}(\ell) \preceq_{r_2} (1 + \varepsilon) K(\ell),$$

and the coordinator maintains the matrix $\tilde{K}(\ell)$. This implies that

$$\frac{1 - \varepsilon}{1 + \varepsilon} K(\ell) \preceq_{r_1} \frac{1}{1 + \varepsilon} \tilde{K}(\ell) \preceq_{r_1} K(\ell).$$

Combining this with Statement 2 of Lemma 3.3, we have that

$$\frac{1 - \varepsilon}{2(1 + \varepsilon)} K(\ell + 1) \preceq_{r_1} \frac{1}{2(1 + \varepsilon)} \tilde{K}(\ell) \preceq_{r_1} K(\ell + 1).$$

We apply the same sampling procedure as the base case, and obtain a matrix $\tilde{K}(\ell + 1)$ such that

$$(1 - \varepsilon) K(\ell + 1) \preceq_{r_1} \tilde{K}(\ell + 1) \preceq_{r_2} (1 + \varepsilon) K(\ell + 1).$$

Notice that, given $\tilde{K}(\ell)$ in the blackboard, the probabilities used for sampling individual edges can be computed locally by different sites, and in each round only the sampled edges will be sent to the coordinator in order for the coordinator to obtain $\tilde{K}(\ell + 1)$. Hence, the total communication cost in each iteration is $\tilde{O}(n + s)$. 

\hfill \Box

C Omitted details from Section 4

C.1 Proof of Theorem 4.2

Proof. We can show the following technical lemma.
Lemma C.1. In the message-passing model, \( \Omega(s\ell) \) bits of communication is needed for computing at least a 0.8 fraction of \( j \in [\ell] \) \( \text{ALLONE}_s(X^j) \) correctly with probability 0.99 under the input distribution \( X \sim \mu_\ell \).

Proof. By a Markov inequality, there must exist \( \Omega(s) \) coordinates \( j \) such that the algorithm computes \( \text{ALLONE}_s(X^j) \) \( (X^j \sim \nu) \) with error probability at most 0.24. Call each of such coordinates \( j \) good. Let \( \Pi \) be the protocol transcript. We have

\[
I(X; \Pi) = \sum_{j \in [\ell]} I(X^j; \Pi | X^{-j}) \geq \sum_{j \in [\ell]} I(X^j; \Pi) \quad (X^j \text{ and } X^{-j} \text{ are independent}) \\
= \sum_{j \in \text{good } j} I(X^j; \Pi) \geq \Omega(s) \cdot \text{IC}_{0.24, \nu}(\text{ALLONE}_s) \geq \Omega(s\ell). \quad \text{(Theorem B.3)}
\]

The reduction. We consider \( 8\ell \) point locations on a line with \( x \)-coordinates being \( 1, 2, \ldots, 8\ell \). We put a point with infinite weight at every even point location. We name the \( 4\ell \) odd point locations from left to right as \( p^1, q^1, p^2, q^2, \ldots, p^\ell, q^\ell, z^1, z^2, \ldots, z^{2\ell} \).

For each site \( i \in [s] \) and each column \( j \in [\ell] \), if \( X^i_j = 0 \) then we put a point with weight 1 at location \( q^j \); otherwise if \( X^i_j = 1 \) then we put a point with weight 1 at location \( p^j \). We also put a point with weight 1/2 at each of the “dummy” locations \( z^1, \ldots, z^{2\ell} \). Let the weight of a location be the sum of the weights of points falling into that location.

Given such an input \( X \), for both \( k \)-median and \( k \)-means, the optimal solution (OPT) which is allowed to use \( k = 6\ell \) centers will include all locations \( p^j \) and \( q^j \) whose weights are at least 1 (note that there are at most \( 2\ell \) such locations), the \( 4\ell \) even point locations, and as many as dummy locations that it can still include. The cost of the optimal solution will be precisely the cost of linking the points in the rest of the dummy locations to their nearest centers (at the even locations), which can be written as

\[
\text{OPT} = 1/2 \cdot (k/3 - (k/3 - F_0)) = 1/2 \cdot F_0 \leq \ell,
\]

where \( F_0 \) is the number of locations in \( \{p^1, q^1, \ldots, p^\ell, q^\ell\} \) that have weights at least 1.

Now suppose our solution (SOL) outputs \( ck \) centers for a constant \( c \in [1, 1.01] \). Each time we include a location \( q^j \) as a center when there is no \( 0 \)-coordinate in the input column \( X^j \), we have a loss of 1/2 since we miss out on including a dummy location (i.e., we can take one more dummy location instead of taking \( q^j \) as a center). Similarly, each time we do not include a location \( q^j \) as a center when there is a \( 0 \)-coordinate in \( X^j \), we have a loss of 1/2 since a point at \( q^j \) has weight at least 1 but a point at a dummy location has weight at 1/2. Therefore, even if we are allowed to output \( ck \) medians, we will still need to figure out whether there is any point at location \( q^j \) for at least an \( \alpha = 0.9 \) fraction of the coordinates \( j \in [\ell] \).

If not, then

\[
\text{SOL} - \text{OPT} \geq 1/2 \cdot (1 - \alpha) \ell - 1 \cdot (c - 1) k = \frac{(1 - \alpha) - 12(c - 1)}{2} \ell \\
\geq (6.1 - 6c) \text{OPT},
\]

where the first term in the RHS of (5) counts the loss of incorrectly computing the (at least) \( (1 - \alpha) \ell \) coordinates \( j \in [\ell] \), and the second term counts the maximum gain of the extra \( (c - 1)k \) centers SOL can use (compared with OPT).

By Lemma C.1, we have that for any \( c \in [1, 1.01] \), computing \( (7.1 - 6c, c) \)-bicriteria-approximation for \( k \)-median or \( k \)-means in the message passing model correctly with probability 0.9 under distribution \( X \sim \mu \) needs \( \Omega(sk) \) bits of communication. Theorem 4.2 follows by Yao’s minimax principle. \( \square \)
C.2 Algorithm for geometric clustering in the blackboard model

Our algorithm for $k$-median/means is an easy adaptation of the successive sampling algorithm proposed by Mettu and Plaxton [16] in the (centralized) RAM model. We first summarize their algorithm and then describe how to port it to the blackboard model.

Let $X_1, \ldots, X_s$ be the point sets at sites $P_1, \ldots, P_k$ respectively. The successive sampling algorithm proceeds in rounds. At each round $j$ it does the following:

1. $s$ sites jointly sample $O(k)$ point centers, denoted by $Y_j$;
2. $s$ sites grow balls from each of the point centers in $Y_j$ synchronously until a time step when a $0.9$ fraction of points in $\bigcup_{i \in [s]} X_i$ are covered;
3. each site $P_i$ updates $X_j$ by removing those points that are covered by any of the balls centered at points in $Y_j$;
4. $s$ sites remove all the points covered by balls centered at points in $Y_j$, and proceed to the next round $j + 1$.

It is easy to see that the computation will finish in $r = O(\log n)$ rounds since at each round we remove a constant fraction of points. At the end we compute an $O(1)$-approximation of $k$-median or $k$-means on the $O(k \log n)$ points $\bigcup_{j \in [r]} Y_j$. In [16] it has been shown that this algorithm gives an $O(1)$-approximation to $k$-median or $k$-means with high probability.

We now describe how to implement this centralized algorithm in the blackboard model. We first consider each round. Step 1 can be done by the distributed sampling algorithm in [8] using $\tilde{O}(k + s)$ bits of communication; note that at the end of this step the sampled points in $Y_j$ are written on the blackboard. Step 2 can be done by a binary search for the minimum ball radius $t_j$ such that $\bigcup_{p \in Y_j} \text{Ball}(p, t_j)$ covers at least a $0.9$ fraction of points in $\bigcup_{i \in [s]} X_i$, where $\text{Ball}(p, t_j)$ denotes the ball centered at $p$ with radius $t_j$; this binary search can be done using $\tilde{O}(1)$ bits of communication. Step 3 and 4 can be done locally without any communication. After $r$ rounds, the final clustering step can be done by any of the $s$ sites since all points in $\bigcup_{j \in [r]} Y_j$ have already been written on the blackboard.

D More Experimental Results

D.1 Datasets

Figure 4 visualizes the datasets for our experiments.

D.2 Quality of the Clustering

Figure 5 visualizes the qualities of the results returned by the three algorithms on datasets Gauss and Sculpture.
Figure 5: Visualization of results on Gauss and Sculpture; in the message passing model each site samples $5n$ edges; in the blackboard model all sites jointly sample $10n$ (in Gauss) or $20n$ (in Sculpture) edges and the chain has length 18.

D.3 Parameters in MsgPassing and Blackboard

Figure 6 shows in MsgPassing, how the value of ncut is affected by the number of sites and the number of edges sampled in each site. Here, each site samples $cn$ edges. When $c = 3$ and $s = 1$, the ncut value diverges in all datasets. This is because with such a small $c$, the algorithm does not generate a valid sparsifier. In general, increasing $c$ or $s$ will slightly decrease the ncut value. But once they are above some thresholds, the ncut values of MsgPassing and Baseline become very close.

Figure 7 shows in Blackboard, how the ncut value is affected by the number of iterations and the number of edges sampled. When the number of iterations is set to be 5, ncut values diverge in all datasets. This is because we cannot expect to generate a valid sparsifier by using such few iterations. It can be seen from 7(b) that for a fixed $c$, performing more iterations will help to reduce ncut values. From the same figure, one can also conclude that for fixed iterations, increasing $c$ also help to reduce the ncut values.
Figure 6: The pictures above show the ncut values with respect to the values of $c$ and $s$ for the MsgPassing algorithm. Here each site samples $cn$ edges.
Figure 7: The pictures above show how the ncut values are affected by the number of iterations and the value of $c$ for the Blackboard algorithm. Here all sites jointly sample $cn$ edges.