Communication-Optimal Distributed Dynamic Graph Clustering

Chun Jiang Zhu,[1] Tan Zhu,[1] Kam-Yiu Lam,[2] Song Han,[1] Jinbo Bi[1]
[1]Department of Computer Science and Engineering, University of Connecticut, Storrs, CT, USA
{chunjiang.zhu, tan.zhu, song.han, jinbo.bi}@uconn.edu
[2]Department of Computer Science, City University of Hong Kong, Hong Kong, PRC
cskylam@cityu.edu.hk

Abstract

We consider the problem of clustering graph nodes over large-scale dynamic graphs, such as citation networks, images and web networks, when graph updates such as node/edge insertions/deletions are observed distributively. We propose communication-efficient algorithms for two well-established communication models namely the message passing and the blackboard models. Given a graph with n nodes that is observed at s remote sites over time $[1, t]$, the two proposed algorithms have communication costs $\tilde{O}(ns)$ and $\tilde{O}(n + s)$ ($\tilde{O}$ hides a polylogarithmic factor), almost matching their lower bounds, $\Omega(ns)$ and $\Omega(n + s)$, respectively, in the message passing and the blackboard models. More importantly, we prove that at each time point in $[1, t]$ our algorithms generate clustering quality nearly as good as that of centralizing all updates up to that time and then applying a standard centralized clustering algorithm. We conducted extensive experiments on both synthetic and real-life datasets which confirmed the communication efficiency of our approach over baseline algorithms while achieving comparable clustering results.

1 Introduction

Graph clustering is one of the most fundamental tasks in artificial intelligence and machine learning (Giatsidis et al. 2014; Tian et al. 2014; Anagnostopoulos et al. 2016). Given a graph consisting of a node set and an edge set, graph clustering asks to partition graph nodes into clusters such that nodes within the same cluster are "densely-connected" by graph edges, while nodes in different clusters are "loosely-connected". Graph clustering on modern large-scale graphs imposes high computational and storage requirements, which are too expensive, if not impossible, to obtain from a single machine. In contrast, distributed computing clusters and server storages are a popular and cheap way to meet the requirements. Distributed graph clustering has received considerable research interests (Hui et al. 2007; Yang and Xu 2015; Chen et al. 2016; Sun and Zanetti 2017). However, the dynamic nature of modern graphs makes the clustering problem even more challenging. We discuss several motivational examples and their characteristics as follows.

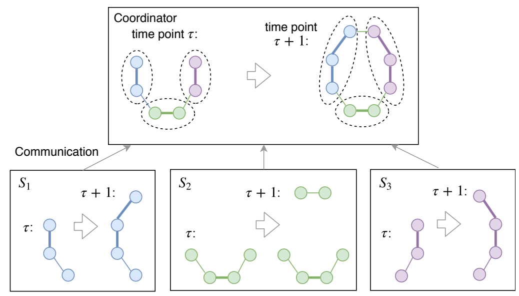


Figure 1: Illustration of distributed dynamic graph clustering. Thick edges have an edge weight 3 while thin edges have an edge weight 1. Clustering results are evolving over time.

Citation Networks. Graph clustering on citation networks aims to generate groups of papers/manuscripts/patents with many similar citations. This implies that the authors within each cluster share similar research interests. The clustering results can be useful for recommending research collaboration, e.g. in ResearchGate. Large-scale citation networks, e.g. the US patent citation network (1963-1999)[1], contain millions of patents and tens of millions of citations, and they are dynamic with frequent insertions. New papers are published everyday with new citations to be added to the network graph. Citation networks usually have negligible deletions because very few works get revoked.

Large Images. Image segmentation is a fundamental task in computer vision (Arbelaez et al. 2011). Graph-based image segmentation has been studied extensively (Shi and Malik 2000; Maier, Luxburg, and Hein 2009; Kim et al. 2011). In these methods, each pixel is mapped into a node in a high-dimensional space (considering coordinates and intensity) that then connects to its K-nearest nodes. In many applications such as in astronomy and microscopy, high-resolution images are captured with an extremely large size, up to gigapixels. Segmentation of these images usually requires pipelining, such as with deblurring as a preprocessing, so new pixels could be added for image segmentation over time. Similar to citation networks, no pixels and their edges would be deleted once they are inserted into the images.

[1]https://snap.stanford.edu/data/cit-Patents.html

Web Graphs. In a web graph with web pages as nodes and hyperlinks between pages as edges, web pages within the same community are usually densely-connected. Clustering results on a web graph can be helpful for eliminating duplicates and recommending related pages. There have been over 46 billion web pages on the WWW until July, 2018 (Worldwidewebsize 2018), and its size grows fast as new web pages have been constantly crawled over time. The deletions of web pages are much less frequent and more difficult to discover than insertions. In some cases, deleted web pages are still kept in Web graphs for analytic purposes.

All these examples require effective ways to clustering over large-scale dynamic graphs, when node/edge insertions/deletions are observed distributively and over time. For notation convenience, we assume that we know an estimated total number of nodes in the graphs, and then node insertions and deletions are treated as insertions/deletions of its edges. Since deletions seldom happen, we first only consider node/edge insertions, and then discuss how to include a small number of deletions in detail. Formally, there are s distributed remote sites $S_1, \cdots, S_s$ and a coordinator. At each time point $\tau \in [1, t]$, each of these sites observes a graph update stream $\hat{E}_i^\tau$, defining the local graph $G_i^\tau(V, E_i^\tau = \cup_{j=1}^{\tau} \hat{E}_i^j)$ observed up to the time point τ, and these sites corporate with the coordinator to generate graph clustering over the global graph $G^\tau(V, E^\tau = \cup_{i=1}^{s} E_i^\tau)$. For simplicity, edge weights cannot be updated but an edge can be observed at different sites. We illustrate the problem by an example in Fig. 1.

For distributed systems, communication costs are one of the major performance measures we aim to optimize. In this paper, we consider two well-established communication models in multi-party communication literature (Phillips, Verbin, and Zhang 2016), namely the message passing and the blackboard models. In the former model, there is a communication channel between each of the s remote sites and a distinguished coordinator. Each site can send a message to another site by first sending to the coordinator, who then forwards the message to the destination. In the latter model, there is a broadcast channel to which a message sent is visible to all sites. Note that both models abstract away issues of message delay, synchronization and loss and assume that each message is delivered immediately. These assumptions can be removed by using standard techniques of timestamping, acknowledgements and re-sending, respectively. We measure communication costs in terms of the total number of bits communicated.

Unfortunately, existing graph clustering algorithms cannot work reasonably well for the problem we considered. In order to show the challenge, we discuss two natural methods central (*CNTRL*) and static (*ST*). For every time point in $[1, t]$, *CNTRL* centralizes all graph updates that are distributively arriving and then applies any centralized graph clustering algorithm. However, the total communication cost $\tilde{O}(m)$ for *CNTRL* is very high, especially when the number m of edges is very large. On the other hand, for every time point in $[1, t]$, *ST* applies any distributed static graph clustering algorithm on the current graph and thus adapt it to distributed dynamic setting. According to (Chen et al. 2016), the lower bounds on communication cost for distributed graph clustering in the message passing and the blackboard models are $\Omega(ns)$ and $\Omega(n + s)$, respectively, where n is the number of nodes in the graph and s is the number of sites. Summing over t time points, the total communication cost for *ST* are $\Omega(nst)$ and $\Omega(nt + st)$ resp., which could be very high especially when t is very large. Therefore, designing new algorithms for distributed dynamic graph clustering is significant and challenging because of the scarce of any valid algorithms.

Contribution. The contribution of our work are summarized as follows.

- For the message passing model, we analyze the problem of *ST* and propose an algorithm framework namely Distributed Dynamic Clustering Algorithm with Monotonicity Property (*D^2-CAMP*), which can significantly reduce the total communication cost to $\tilde{O}(ns)$, for an n-node graph distributively observed at s sites in a time interval $[1, t]$. Any spectral sparsification algorithms (we will formally introduce in Sec. 2) satisfying the monotonicity property can be used in *D^2-CAMP* to achieve the communication cost.
- We propose an algorithm namely Distributed Dynamic Clustering Algorithm for the BLackboard model (*D^2-CABL*) with communication cost $\tilde{O}(n + s)$ by adapting the spectral sparsification algorithm (Cohen, Musco, and Pachocki 2016). *D^2-CABL* is also a new static distributed graph clustering algorithm with nearly-optimal communication cost, the same as the iterative sampling approach (Li, Miller, and Peng 2013) based state of the art (Chen et al. 2016). However, it is much simpler and also works for the more complicated distributed dynamic setting.
- More importantly, we show that the communication costs of *D^2-CAMP* and *D^2-CABL* match their lower bounds $\Omega(ns)$ and $\Omega(n + s)$ up to polylogarithmic factors, respectively. And then we prove that at every time point, *D^2-CAMP* and *D^2-CABL* can generate clustering results of quality nearly as good as *CNTRL*.
- Finally, we have conducted extensive experiments on both synthetic and real-world networks to compare *D^2-CAMP* and *D^2-CABL* with *CNTRL* and *ST*, which shows that our algorithms can achieve communication cost significantly smaller than these baselines, while generating nearly the same clustering results.

Related Work. Geometric clustering has been studied by (Cormode, Muthukrishnan, and Wei 2007) in the distributed dynamic setting. They presented an algorithm for k-center clustering with theoretical bounds on the clustering quality and the communication cost. However, it is not for the graph clustering. There have been extensive research on graph clustering in the distributed setting (Hui et al. 2007; Yang and Xu 2015; Chen et al. 2016; Sun and Zanetti 2017) where the graph is static (does not change over time) but distributed. (Yang and Xu 2015) proposed a divide and conquer method for distributed graph clustering. (Chen et al.

2016) used spectral sparsifiers in graph clustering for two distributed communication models to reduce communication cost. (Sun and Zanetti 2017) presented a node degree based sampling scheme for distributed graph clustering, and their method does not need to compute approximate effective resistance. However, as discussed earlier, all these methods suffer from very high communication costs, depending on the time duration, and thus cannot be used in the studied dynamic distributed clustering. Independently, (Jian, Lian, and Chen 2018) studied distributed community detection on dynamic social networks. However, their algorithm is not optimized for communication cost, focusing on finding overlapping clusters and only accepts unweighted graphs. In contrast, our algorithms are optimized for communication cost. They can generate non-overlapping clusters and process both weighted and unweighted graphs.

2 The Proposed Algorithms

We first introduce spectral sparsification that we will use in subsequent algorithm design. Recall that the message passing communication model represents distributed systems with point-to-point communication, while the blackboard model represents distributed systems with a broadcast channel, which can be used to broadcast a message to all sites. We then propose two algorithms for different practical scenarios in Sec. 2.1 and 2.2, respectively.

Graph Sparsification. In this paper, we consider weighted undirected graphs $G(V, E, W)$ and will use n and m to denote the numbers of nodes and edges in G respectively. Graph sparsification is the procedure of constructing sparse subgraphs of the original graphs such that certain important property of the original graphs are well approximated. For instance, a subgraph $H(V, E' \subseteq E)$ is called a *spanner* of G if for every $u, v \in V$, the shortest distance between u and v is at most $\alpha \geq 1$ times of their distance in G (Peleg and Schaffer 1989). Let A_G be the adjacency matrix of G. That is, $(A_G)_{u,v} = W(u, v)$ if $(u, v) \in E$ and zero otherwise. Let D_G be the degree matrix of G defined as $(D_G)_{u,v} = \sum_{v \in V} W(u, v)$, and zero otherwise. Then the unnormalized *Laplacian matrix* and *normalized Laplacian matrix* of G are defined as $L_G = D_G - A_G$ and $\mathcal{L}_G = D_G^{-1/2} L_G D_G^{-1/2}$, resp.. (Spielman and Teng 2011) introduced *spectral sparsification*: a $(1 + \epsilon)$-*spectral sparsifier* for G is a subgraph H of G, such that for every $x \in R^n$, the inequality $(1 - \epsilon)x^T L_G x \leq x^T L_H x \leq (1 + \epsilon)x^T L_G x$ holds. There is a rich literature on improving the trade-off between the size of spectral sparsifiers and the construction time, e.g. (Spielman and Srivastava 2011; Lee and Sun 2017). Recently, (Lee and Sun 2017) proposed the state-of-the-art algorithm to construct a $(1 + \epsilon)$-spectral sparsifier of optimal size $O(n/\epsilon^2)$ (up to a constant factor) in nearly linear time $\tilde{O}(m)$.

2.1 The Message Passing Model

Because spectral sparsifiers have much fewer edges than the original graphs but can preserve cut-based clustering and spectrum information of the original graphs (Spielman and Srivastava 2011), we propose an algorithm framework as follows. At each time point τ, each site S_i first constructs a spectral sparsifier H_i^τ for the local graph $G_i^\tau(V, E_i^\tau)$, and then transmits the much smaller H_i^τ, instead of G_i^τ itself, to the coordinator. Upon receiving the spectral sparsifier H_i^τ from every site at the time τ, the coordinator first takes their union $H^\tau = \cup_{i=1}^s H_i^\tau$ and then applies a standard centralized graph clustering algorithm, e.g., the spectral clustering algorithm (Ng, Jordan, and Weiss 2001), on H^τ to get the clustering C^τ. This process is repeated at the next time point $\tau + 1$ to get the clustering $C^{\tau+1}$ until t.

However, simply re-constructing spectral sparsifiers from scratch at every time point does not provide any bound on the size of the updates to the previous spectral sparsifiers $H_i^{\tau-1}$ for obtaining H_i^τ at every time point τ, and thus needs to communicate the entire spectral sparsifiers H_i^τ of size $O(n)$ at every time point τ. Summing over all s sites and all t time points, the total communication cost is $\tilde{O}(nst)$.

It is natural to consider algorithms for dynamically maintaining spectral sparsifiers in dynamic computational models (Abraham et al. 2016; Kelner and Levin 2013; Kapralov et al. 2014). Unfortunately, applying them also does not provide such a bound, incurring the same communication cost! To see this, the key of (algorithms in) dynamic computational models is a data structure for dynamically maintaining the result of a computation while the underlying input data is updated periodically. For instance, dynamic algorithms (Abraham et al. 2016), after each update to the input data, are allowed to process the update to compute the new result within a fast time; online algorithms (Kelner and Levin 2013) allow to process the input data that are revealed step by step; and streaming algorithms (Kapralov et al. 2014) impose a space constraint while processing the input data that are revealed step by step. The main principle of all these computational models is on efficiently processing the dynamically changing input data, instead of bounding the size of the updates to the previous output result over time.

We define a new type of spectral sparsification algorithms, which can provide such a bound, and is defined as follows.

Definition 1. *For an n-node graph $G(V, E=\{e_1, \cdots, e_m\})$, let $G(V, E_i=\{e_1, \cdots, e_i\})$ be the graph consisting of the first i edges. A spectral sparsification algorithm is called a Spectral Sparsification Algorithm with Monotonicity Property (*S^2AMP*), if the spectral sparsifers $H_1, \cdots, H_m$, constructed for $G_1, \cdots, G_m$, respectively, satisfy that (1) $H_1 \subseteq \cdots \subseteq H_m$; and (2) H_m has size $\tilde{O}(n)$.*

We show that, by using any *S^2AMP* in the algorithm framework mentioned above, we can reduce the total communication cost from $\tilde{O}(nst)$ to $\tilde{O}(ns)$, removing a factor of t. We refer to the resultant algorithm framework as Distributed Dynamic Clustering Algorithm with Monotonicity Property (*D^2-CAMP*). The intuition for the significant reduction in the total communication cost is that, the monotonicity property guarantees that, for every time point $\tau \in [1, t]$, the constructed spectral sparsifiers H_i^τ is a superset of $H_i^{\tau-1}$ at the previous time point $\tau - 1$. Then, we only need to transmit edges in H_i^τ and at the same time not in $H_i^{\tau-1}$ to the coordinator for maintaining H_i^τ. Every communicated bit transmitted at the time point t is used at all subsequent time points $\{\tau+1,\cdots,t\}$, and thus no communication is "wasted". Furthermore, we show that by only switching an arbitrary spectral sparsification algorithm to *S*^{2}*AMP*, the total communication cost $\tilde{O}(ns)$ achieved has been optimal, up to a polylogarithmic factor. That is, we cannot design another algorithm with communication cost smaller than *D*^{2}*-CAMP* by a polylogarithmic factor.

We summarize the results in Theorem 3. For every node set $S \subseteq V$ in G, let its *volume* and *conductance* be $vol_G(S) = \sum_{u\in S, v\in V} W(u,v)$ and $\phi_G(S) = (\sum_{u\in S, v\in V-S} W(u,v))/vol_G(S)$, respectively. Intuitively, a small value of conductance $\phi(S)$ implies that nodes in S are likely to form a cluster. A collection of subsets $A_1,\cdots,A_k$ of nodes is called a *(k-way) partition* of G if (1) $A_i \cap A_j = \emptyset$ for $1 \le i \ne j \le k$; and (2) $\cup_{i=1}^k A_i = V$. The *k-way expansion constant* is defined as $\rho(k) = \min_{partition A_1,\cdots,A_k} \max_{i\in[1,k]} \phi(A_i)$. The eigenvalues of $\mathcal{L}_G$ are denoted as $\lambda_1(\mathcal{L}_G) \le \cdots \le \lambda_n(\mathcal{L}_G)$. The high-order Cheeger inequality shows that $\lambda_k/2 \le \rho(k) \le O(k^2)\sqrt{\lambda_k}$ (Lee, Gharan, and Trevisan 2014). A lower bound on $\Upsilon_G(k) = \lambda_{k+1}/\rho(k)$ implies that, G has exactly k well-defined clusters (Peng, Sun, and Zanetti 2015). It is because a large gap between λ_{k+1} and $\rho(k)$ guarantees the existence of a k-way partition $A_1,\cdots,A_k$ with bounded $\phi(A_i) \le \rho(k)$, and that any $(k+1)$-way partition $A_1,\cdots,A_{k+1}$ contains a subset A_i with significantly higher conductance $\rho(k+1) \ge \lambda_{k+1}/2$ compared with $\rho(k)$. For any two sets X and Y, the symmetric difference of X and Y is defined as $X\Delta Y = (X-Y)\cup(Y-X)$. To prove Theorem 3, we will use the following lemma and theorems.

Lemma 1. *(Chen et al. 2016) Let H be a $(1+\epsilon)$-spectral sparsifier of $G(V,E)$ for some $\epsilon \le 1/3$. For all node sets $S \subseteq V$, the inequality $0.5\cdot\phi_G(S) \le \phi_H(S) \le 2\cdot\phi_G(S)$ holds.*

Theorem 1. *(Chen et al. 2016) Let G be an n-node graph and the edges of G are distributed amongst s sites. Any algorithm that correctly outputs a constant fraction of each cluster in G requires $\Omega(ns)$ bits of communications.*

Theorem 2. *(Peng, Sun, and Zanetti 2015) Given a graph G with $\Upsilon_G(k) = \Omega(k^3)$ and an optimal partition $S_1,\cdots,S_k$ achieving $\rho(k)$ for some positive integer k, the spectral clustering algorithm can output partition $A_1,\cdots,A_k$ such that, for every $i \in [1,k]$, the inequality $vol(A_i\Delta S_i) = O(k^3\Upsilon^{-1}vol(S_i))$ holds.*

Theorem 3 (The Message Passing model)**.** *For every time point $\tau \in [1,t]$, suppose that G^τ satisfies that $\Upsilon(k) = \Omega(k^3)$ and there is an optimal partition $P_1,\cdots,P_k$ which achieves $\rho(k)$ for some positive integer k, D^2-CAMP can output partition $A_1,\cdots,A_k$ at the coordinator such that for every $i \in [1,k]$, $vol(A_i\Delta P_i) = O(k^3\Upsilon^{-1}vol(P_i))$ holds. Summing over all t time points, the total communication cost is $\tilde{O}(ns)$. It is optimal up to a polylogarithmic factor.*

Proof. We start by proving that for every time point $\tau \in [1,t]$, the structure H^τ constructed at the coordinator is a $(1+\epsilon)$-spectral sparsifier of the graph G^τ received up to the time point t. By the monotonicity property of a *S*^{2}*AMP*, for every $i \in [1,s]$, H_i^τ is a $(1+\epsilon)$-spectral sparsifier of the graph $G_i^\tau(V,E_i^\tau)$. The decomposability of spectral sparsifiers states that the union of spectral sparisifiers of some graphs is a spectral sparsifier for the union of the graphs (Sun and Zanetti 2017). Then by this property, the union of $H^\tau = \cup_{i=1}^s H_i^\tau$ obtained at the coordinator is a $(1+\epsilon)$-spectral sparsifier of the graph $G^\tau = \cup_{i=1}^s G_i^\tau$.

Now we prove that for every time point $\tau \in [1,t]$, if G^τ satisfies that $\Upsilon_{G^\tau}(k) = \Omega(k^3)$, H^τ also satisfies that $\Upsilon_{H^\tau}(k) = \Omega(k^3)$. By the definition of Υ, it suffices to prove that $\rho_{H^\tau}(k) = \Theta(\rho_{H^\tau}(k))$ and $\lambda_{k+1}(\mathcal{L}_{H^\tau}) = \Theta(\lambda_{k+1}(\mathcal{L}_{G^\tau}))$. The former follows from that for every $i \in [1,k]$, the inequality

$$0.5\cdot\phi_{G^\tau}(S_i) \le \phi_{H^\tau}(S_i) \le 2\cdot\phi_{G^\tau}(S_i)$$

holds, according to Lemma 1. According to the definition of $(1+\epsilon)$-spectral sparsifier and simple math, it holds for every vector $x \in R^n$ that

$$\begin{aligned}(1-\epsilon)x^T D_{G^\tau}^{-1/2} L_{G^\tau} D_{G^\tau}^{-1/2} x &\le x^T D_{G^\tau}^{-1/2} L_{H^\tau} D_{G^\tau}^{-1/2} x \\ &\le (1+\epsilon)x^T D_{G^\tau}^{-1/2} L_{G^\tau} D_{G^\tau}^{-1/2} x.\end{aligned}$$

By the definition of normalized graph Laplacian $\mathcal{L}_G$, and the fact that for every vector $y \in R^n$,

$$0.5\cdot y^T D_{G^\tau}^{-1} y \le y^T D_{H^\tau}^{-1} y \le 2y^T D_{G^\tau}^{-1} y,$$

we have that for every $i \in [1,n]$,

$$\lambda_i(\mathcal{L}_{H^\tau}) = \Theta(\lambda_i(\mathcal{L}_{G^\tau})),$$

which implies that $\lambda_{k+1}(\mathcal{L}_{H^\tau}) = \Theta(\lambda_{k+1}(\mathcal{L}_{G^\tau}))$. Then we can apply the spectral clustering algorithm on H^τ to get the desirable properties, according to Theorem 2.

For the upper bound on the communication cost, by the monotonicity property of a *S*^{2}*AMP*, each site only needs to transmit $\tilde{O}(n)$ number of edges over all t time points. Summing over all s sites, the total communication cost is $\tilde{O}(ns)$.

For the lower bound, we show the following statement. For every time point $\tau \in [1,t]$, suppose G^τ satisfies that $\Upsilon(k) = \Omega(k^3)$ and there is an optimal partition $P_1,\cdots,P_k$ which achieves $\rho(k)$ for positive integer k, in the message passing model there is an algorithm which can output $A_1,\cdots,A_k$ at the coordinator, such that for every $i \in [1,k]$, $vol(A_i\Delta P_i) = \Theta(vol(P_i))$ holds. Then the algorithm requires $\Omega(ns)$ total communication cost over t time points.

Consider any time point τ. We assume by contradiction that there exists an algorithm which can output $A_1,\cdots,A_k$ in G^τ at the coordinator, such that for every $i \in [1,k]$, $vol(A_i\Delta P_i) = \Theta(vol(P_i))$ holds, using $o(ns)$ bits of communications. Then the algorithm can be used to solve a corresponding graph clustering problem in the distributed but static setting using $o(ns)$ bits of communications. This contradicts Theorem 1, and then completes the proof. □

Combining Theorems 2 and 3, *D*^{2}*-CAMP* could generate clustering of quality asymptotically the same as *CNTRL*. We stress that the monotonicity property in general can be helpful for improving the communication efficiency over distributed dynamic graphs. In Sec. 3, we will discuss a new application which also benefits from the property.

As mentioned earlier, any *S^2AMP* algorithm can be plugged in *D^2-CAMP*, e.g., the online sampling technique (Cohen, Musco, and Pachocki 2016). But the resultant algorithm becomes a randomized algorithm which succeeds w.h.p. because the constructed subgraphs are spectral sparsifiers w.h.p. Another *S^2AMP* algorithm is the online-BSS algorithm (Baston, Spielman, and Srivastava 2012; Cohen, Musco, and Pachocki 2016), which has a slightly smaller communication cost (by a logarithmic factor) but requires larger memory and is more complicated.

2.2 The Blackboard Model

How to efficiently exploit the broadcast channel in the blackboard model to reduce the communication complexity in distributed graph clustering is non-trivial. For example, (Chen et al. 2016) proposed to construct $O(\log n)$ spectral sparisifers as a chain in the blackboard based on the iterative sampling technique (Li, Miller, and Peng 2013). Each spectral sparsifier in the chain is a spectral sparsifer of its following sparsifier. However, the technique fails to extend to the dynamic setting, as each graph update could incur a large number of updates in the maintained spectral sparsifiers, especially for those in the latter part of the chain.

We propose a simple algorithm called Distributed Dynamic Clustering Algorithm for the BLackboard model (*D^2-CABL*), based on adapting Cohen et al.'s algorithm (Cohen, Musco, and Pachocki 2016). The basic idea is that every site corporates with each other to construct a spectral sparsifier H^τ for $G^\tau(V, E^\tau)$ at each time point τ in the blackboard.

Algorithm 1: *D^2-CABL* at Time Point τ

Input: The incidence matrix $B^{\tau-1}$, new edges $\hat{E}^\tau$ coming at τ, $\delta > 0$, $\epsilon \in (0, 1/3)$
Output: The incidence matrix B^τ

$\lambda \leftarrow \delta/\epsilon$; $c \leftarrow 8 \log n/\epsilon^2$;
$B' \leftarrow B^{\tau-1}$;
for $e \in \hat{E}^\tau$ **do**
  $l = (1+\epsilon) b(e)^T (B'^T B' + (\delta/\epsilon)I)^{-1} b(e)$;
  $p \leftarrow \min\{cl, 1\}$;
  $B' \leftarrow [B'; b(e)/\sqrt{p}]$ with probability p;
end
return $B^\tau \leftarrow B'$;

The edge-node incidence matrix $B_{m\times n}$ of G is defined as $B(e, v) = 1$ if v is e's head, $B(e, v) = -1$ if v is e's tail, and zero otherwise. At the beginning, the parameters δ and ϵ of the algorithm are set by a distinguished site and then sent to every site, and the blackboard has an empty spectral sparsifier H^0, or equivalently an empty incidence matrix B^0 of dimension $0 \times n$. Consider the time point τ. Suppose that at the previous time point $\tau - 1$, the incidence matrix $B^{\tau-1}$ for $H^{\tau-1}$ was in the blackboard. For each newly observed edge $e \in \hat{E}^\tau$ at the time point τ, the site S_i observing e computes the online ridge leverage score $l = (1+\epsilon) b(e)^T (B'^T B' + (\delta/\epsilon)I)^{-1} b(e)$ by accessing the incidence matrix B' currently in the blackboard, where $b(e)$ is an n-dimensional vector with all zeroes except that the entries corresponding to e's head and tail are 1 and -1, resp..

Let the sampling probability $p = \min\{(8 \log n/\epsilon^2)l, 1\}$. With probability p, e is sampled, or discarded otherwise. If e is sampled, the site S_i transmits the rescaled vector $b(e)/\sqrt{p}$ corresponding to e to the blackboard to append it at the end of B'. After all the newly observed edges $\hat{E}^\tau$ at the time point τ at all the sites are processed, B^τ for H^τ will be in the blackboard. Then the coordinator applies any standard graph clustering algorithm, e.g. (Ng, Jordan, and Weiss 2001), on H^τ to get the clustering C^τ. The process is repeated for every subsequent time point until t. The algorithm is summarized in Alg. 1.

Our results for the blackboard model are summarized in Theorem 4. To prove Theorem 4, first it follows from (Cohen, Musco, and Pachocki 2016) that the constructed subgraph in the blackboard for every time point τ is a spectral sparsifier for the graph G^τ w.h.p.. Then the rest of the proof is the same as the proof of Theorem 3. In the algorithm, processing an edge requires only B', which is in the blackboard and visible to every site. Therefore, each site can process its edges locally and only transmit the sampled edges to the blackboard. The total communication cost is $\tilde{O}(n + s)$, because the size of the constructed spectral sparsifier is $\tilde{O}(n)$ and each site has to transmit at least one bit of information. It is easy to see this communication cost is optimal up to polylogarithmic factors, because even only for one time point, the clustering result itself has $\Omega(n)$ bits of information and each site has to transmit at least one bit of information.

Theorem 4 (The Blackboard model)**.** *For every time point $\tau \in [1, t]$, suppose that G^τ satisfies that $\Upsilon(k) = \Omega(k^3)$ and there is an optimal partition $P_1, \cdots, P_k$ which achieves $\rho(k)$ for some positive integer k, w.h.p.* D^2-CABL *can output partition $A_1, \cdots, A_k$ at the coordinator such that for every $i \in [1, k]$, $vol(A_i \Delta P_i) = O(k^3 \Upsilon^{-1} vol(P_i))$ holds. Summing over t time points, the total communication cost is $\tilde{O}(n + s)$. It is optimal up to a polylogarithmic factor.*

D^2-CABL can also work in the distributed static setting by considering that there is only one time point, at which all graph information comes together. As mentioned earlier, it is a brand new algorithm with nearly-optimal communication complexity, the same as the state-of-the-art algorithm (Chen et al. 2016). But our algorithm is much simpler without having to maintain a chain of spectral sparsifiers. Another advantage is the simplicity that one algorithm works for both distributed settings. The computational complexity for computing the online ridge leverage score for each edge in Alg. 1 is $O(n^2 m)$. To save computational cost, we can batch process in every site new edges $\hat{E}_i^\tau$ observed at each time point τ in a batch of $O(n)$. By using the Johnson-Linderstrauss random projection trick (Spielman and Srivastava 2011), we can approximate online ridge leverage scores for a batch of $O(n)$ edges in $\tilde{O}(n \log m) = \tilde{O}(n)$ time, and then sample all edges together according to the computed scores.

3 Discussions

Another Application of the Monotonicity Property. Consider the same computational and communication models.

When the queries posed at the coordinator are changed to approximate shortest path distance queries between two given nodes, we use graph spanners (Peleg and Schaffer 1989; Althofer et al. 1993) to sparsify the original graphs while well approximating all-pair shortest path distances in the original graphs.

We now describe the algorithm. In the message passing model, at each time point t each site S_i first constructs a graph spanner Q_i^τ of the local graph $G_i^\tau(V, E_i^\tau)$ using a *D^2-CAMP* for constructing graph spanners (Elkin 2011), and then transmits Q_i^τ to the coordinator. Upon receiving Q_i^τ from every site, the coordinator first takes their union $Q^\tau = \cup_{i=1}^{s} Q_i^\tau$ and then applies a point-to-point shortest path algorithm (e.g., Dijkstra's algorithm (Dijkstra 1959)) on Q^τ to get the shortest distance between the two nodes at the time point τ. This process is repeated for every $\tau \in [1, t]$. The theoretical guarantees of the algorithm are summarized in Theorem 5, and its proof is in Sec. 3 of Appendix, included in the full version of this paper [2].

Theorem 5. *Given two nodes $u, v \in V$ and an integer $k > 1$, for every time point $\tau \in [1, t]$, the proposed algorithm can answer approximate shortest distance between u and v in G^τ no larger than $2k - 1$ times of their actual shortest distance at the coordinator in the message passing model. Summing over t time points, the total communication cost is $\tilde{O}(n^{1+1/k}s)$.*

Dynamic Graph Streams. When the graph update stream observed at each site is a fully dynamic stream containing a small number of node/edge deletions, we present a simple trick which enables that our algorithms still have good performance. We observe that the spectral sparsifiers can probably keep unchanged, when there is only a small number of deletions. This is reasonable because spectral sparsifiers are sparse subgraphs which could contain much smaller edges than the original graphs. When the number of deletions is small, the deletions may not affect the spectral sparsifiers at all. Even when the deletions lead to small changes in the spectral sparsifiers, there is a high probability that the clustering is not changed significantly. Therefore, in order to save communication and computation, we can ignore and do not process or transmit these deletions while still approximately preserving the clustering. We experimentally confirm the effects of this thick in the experiment section.

4 Experiments

In this section, we present the experimental results that we conducted on both synthetic and real-life datasets, where we compared the proposed algorithms *D^2-CAMP* and *D^2-CABL* with baseline algorithms *CNTRL* and *ST*. For *ST*, we used the distributed static graph clustering algorithms (Chen et al. 2016) in the message passing and the blackboard models, and refer the resultant algorithms as *STMP* and *STBL*, respectively. For measuring the quality of the clustering results, we used the normalized cut value (NCut) of the clustering (Sun and Zanetti 2017). A smaller value of NCut implies a better clustering while a larger value of NCut implies a worse clustering. For simplicity, we used the total number of edges communicated as the communication cost, which approximates the total number of bits by a logarithmic factor. We implemented all five algorithms in Matlab programs, and conducted the experiments on a machine equipped with Intel i7 7700 2.8GHz CPU, 8G RAM and 1T disk storage.

Time	s	Gaussians *D^2-CAMP*	Gaussians *D^2-CABL*	Sculpture *D^2-CAMP*	Sculpture *D^2-CABL*
50	15	4485	3132	15292	7130
	30	4607	3133	15235	6054
	45	4660	3126	15560	6076
	60	4669	3095	15764	6705
90	15	7342	4988	27036	12153
	30	7533	4982	27020	10287
	45	7586	4979	27700	10336
	60	7630	4960	28001	11421
100	15	7748	5238	28408	12846
	30	7988	5230	28338	10874
	45	7998	5235	29038	10897
	60	8062	5218	29343	12062

Table 1: Communication cost with varied values of s

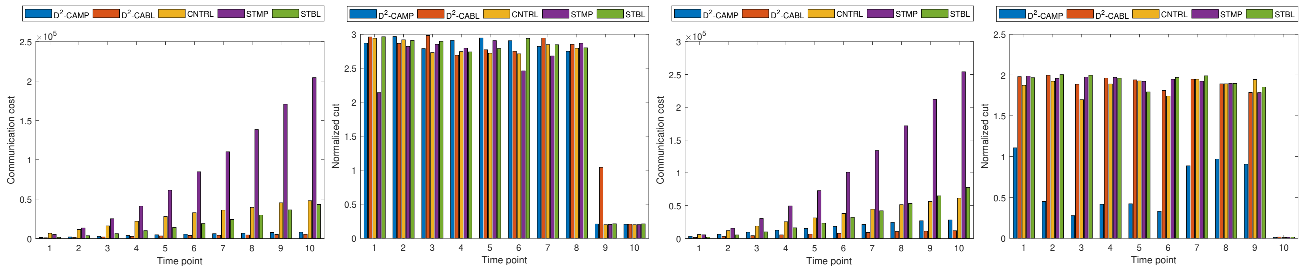
The details of the datasets we used in the experiments are described as follows. The *Gaussians* dataset consists of 800 nodes and 47,897 edges. Each point from each of four clusters is sampled from an isotropic Gaussians of variance 0.01. We consider each point to be a node in constructing the similarity graph. For every two nodes u and v such that one is among the 100-nearest points of the other, we add an edge of weight $W(u, v) = exp\{-||u - v||_2^2/2\sigma^2\}$ with $\sigma = 1$. The number k of clusters is 4. For the *Sculpture* dataset, we used a 22×30 version of a photo of The Greek Slave[3], and it contains 1980 nodes and 61,452 edges. We consider each pixel to be a node by mapping each pixel to a point in R^5, i.e. (x, y, r, g, b), where the last three coordinates are the RGB values. For every two nodes u and v such that u (v) is among the 80-nearest points of v (u), we add an edge of weight $W(u, v) = exp\{-||u - v||_2^2/2\sigma^2\}$ with $\sigma = 20$. The number k of clusters is 3.

In the problem studied, the site and the time point each edge comes is arbitrary. Therefore, we make that the edges of nodes with smaller x coordinates have smaller arrival times than the edges of nodes with larger x coordinates. Intuitively, this results in that the edges of nodes on the left side come before the edges of nodes on the right side. This helps us to easily monitor the changing of the clustering results. Independently, the site every edge comes is randomly picked from the interval $[1, s]$.

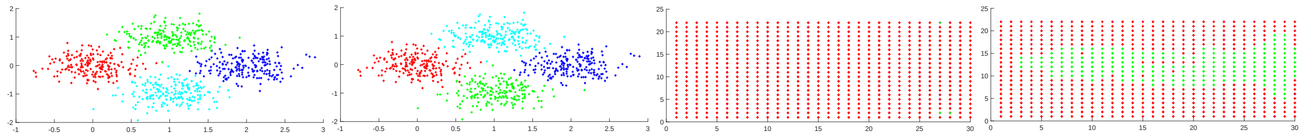
Experimental Results. As the baseline setting, we selected the total number of time points $t = 10$ and the total number of sites $s = 30$. The communication cost and NCut of different algorithms on both datasets are shown in Fig. 2. On both datasets, the communication cost of *D^2-CAMP* and *D^2-CABL* are much smaller than *CNTRL*, *STMP* and *STBL*. Specifically, on Gaussians dataset, the communication cost of *D^2-CAMP* can be only 4% of that of *STMP* and on aver-

[2]https://chunjiangzhu.github.io/cdgc3-full-version.pdf

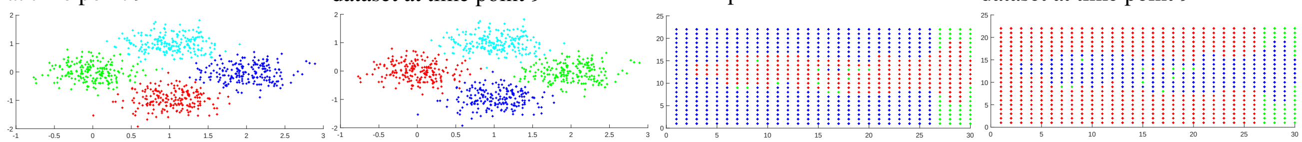
[3]http://artgallery.yale.edu/collections/objects/14794



(a) Comm. on Gaussians dataset (b) NCut on Gaussians dataset (c) Comm. on Sculpture dataset (d) NCut on Sculpture dataset



(e) *CNTRL* on Gaussians dataset at time point 9 (f) D^2-*CAMP* on Gaussians dataset at time point 9 (g) *CNTRL* on Sculpture dataset at time point 9 (h) D^2-*CAMP* on Sculpture dataset at time point 9



(i) *CNTRL* on Gaussians dataset at time point 10 (j) D^2-*CAMP* on Gaussians dataset at time point 10 (k) *CNTRL* on Sculpture dataset at time point 10 (l) D^2-*CAMP* on Sculpture dataset at time point 10

Figure 2: Communication cost, NCut and clustering results in the baseline setting

Time	t	Gaussians D^2-*CAMP*	Gaussians D^2-*CABL*	Sculpture D^2-*CAMP*	Sculpture D^2-*CABL*
50%	10	4562	3127	15078	5998
	30	4645	3126	15278	6063
	100	4607	3133	15235	6054
	300	4620	3113	15269	6064
90%	10	7467	4979	26699	10202
	30	7581	4983	27012	10278
	100	7533	4982	27020	10287
	300	7618	4958	27042	10299
100%	10	7917	5225	28046	10779
	30	8045	5234	28278	10847
	100	7988	5230	28338	10874
	300	8031	5211	28345	10869

Table 2: Communication cost with varied values of t

age 16% of that of *CNTRL*. The communication cost of D^2-*CABL* is on average 11% of *CNTRL* and can be only 12% of that of *STBL*. *STMP* has communication cost even much larger than *CNTRL*. D^2-*CABL* has a smaller communication cost than D^2-*CAMP*. On Sculpture dataset, the communication cost of D^2-*CAMP* can be only 11% of that of *STMP* and is on average 49% of that of *CNTRL*. The communication cost of D^2-*CABL* can be only 15% of that of *STBL* and is on average 21% of that of *CNTRL*. Similar to *STMP*, *STBL* also has communication cost larger than *CNTRL*. D^2-*CABL* has a much smaller communication cost than D^2-*CAMP* and the difference here is larger than in Gaussians dataset.

For both datasets, all algorithms have comparable NCut at every time point, except that on Gaussians dataset, at the time point 9, D^2-*CABL* has a slightly larger NCut. This could be due to that D^2-*CABL* is a randomized algorithm with high success probability. In Fig. 2(e-l), the clustering results of *CNTRL* and D^2-*CAMP* on both datasets at time points 9 and 10 are visually very similar. (The same cluster colors in different figures do not have relation.) But for Sculpture dataset at the time point 9, the clustering result of D^2-*CAMP* visually looks even more reasonable.

We then varied the value of s from 15 to 60 with a step of 15 or the value of t from 10 to 300 with a factor of 3 while keeping the other parameters unchanged as in the baseline setting. Due to limit of space, we only show the resultant communication cost of D^2-*CAMP* and D^2-*CABL* on both datasets in Tables 1 and 2. But the complete results are referred to Appendix. When we varied the value of s, the communication cost of D^2-*CAMP* increases roughly linearly with the increase of the value of s from 15 to 60, while that of D^2-*CABL* do not obviously increase with the value of s. These observations are consistent with their theoretical communication cost $\tilde{O}(ns)$ and $\tilde{O}(n+s)$, respectively. When we varied the value of t, both the communication cost of D^2-*CAMP* and D^2-*CABL* roughly keep the same, also supporting our theory above.

Finally, we tested the performance of D^2-*CAMP* and D^2-*CABL* for dynamic graph streams. We randomly chose 5% of edges to delete at a random time point after their arrival. This increases the communicate cost of *CNTRL* by 5% as *CNTRL* sends every deletion to the coordinator/blackboard. However, the communication cost of D^2-*CAMP* and D^2-*CABL* are not changed. More importantly, even ignoring the deletions, the resultant clusterings of D^2-*CAMP* and D^2-*CABL* at every time point have NCut comparable to that of *CNTRL*. Due to limit of space, we refer to Fig. 1 in Appendix.

5 Conclusion and Future Work

In this paper, we study the problem of how to efficiently perform graph clustering over modern graph data that are often *dynamic* and collected at *distributed* sites. We design communication-optimal algorithms D^2-*CAMP* and D^2-*CABL* for two different communication models and prove their optimality rigorously. Finally, we conducted extensive simulations to confirm that D^2-*CAMP* and D^2-*CABL* significantly outperform baseline algorithms in practice. As the future work, we will study whether and how we can achieve similar results for geometric clustering, and how to achieve better computational bounds for the studied problems. We will also study other related problems in the distributed dynamic setting such as low-rank approximation (Bringmann, Kolev, and Woodruff 2017), source-wise and standard round-trip spanner constructions (Zhu and Lam 2017; 2018) and cut sparsifier constructions (Abraham et al. 2016).

Acknowledgments

This work was partially supported by NSF grants DBI-1356655, CCF-1514357, IIS-1718738, as well as NIH grants R01DA037349 and K02DA043063 to Jinbo Bi.

References

Abraham, I.; Durfee, D.; Koutis, I.; Krinninger, S.; and Peng, R. 2016. On fully dynamic graph sparsifiers. In *Proceedings of FOCS Conference*, 335–344.

Althofer, I.; Das, G.; Dobkin, D.; Joseph, D.; and Soares, J. 1993. On sparse spanners of weighted graphs. *Discrete Computational Geometry* 9:81–100.

Anagnostopoulos, A.; Lacki, J.; Lattanzi, S.; Leonardi, S.; and Mahdian, M. 2016. Community detection on evolving graphs. In *Proceedings of NIPS Conference*, 3530–3538.

Arbelaez, P.; Maire, M.; Fowlkes, C.; and Malik, J. 2011. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33(5):898–916.

Baston, J.; Spielman, D.; and Srivastava, N. 2012. Twice-ramanujan sparsifiers. *SIAM Journal on Computing* 41(6):1704–1721.

Bringmann, K.; Kolev, P.; and Woodruff, D. 2017. Approximation algorithms for l_0-low rank approximation. In *Proceedings of NIPS Conference*, 6648–6659.

Chen, J.; Sun, H.; Woodruff, D.; and Zhang, Q. 2016. Communication-optimal distributed clustering. In *Proceedings of NIPS Conference*, 3720–3728.

Cohen, M.; Musco, C.; and Pachocki, J. 2016. Online row sampling. In *Proceedings of APPROX-RANDOM Conference*, 7:1–7:18.

Cormode, G.; Muthukrishnan, S.; and Wei, Z. 2007. Conquering the divide: continuous clustering of distributed data streams. In *Proceedings of ICDE Conference*, 1036–1045.

Dijkstra, E. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1):269–271.

Elkin, M. 2011. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Transactions on Algorithms* 7(2):20.

Giatsidis, C.; Malliaros, F.; Thilikos, D.; and Vazirgiannis, M. 2014. CORECLUSTER: A degeneracy based graph clustering framework. In *Proceedings of AAAI Conference*, 44–50.

Hui, P.; Yoneki, E.; Chan, S.; and Crowcroft, J. 2007. Distributed communicty detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE International Workshop on Mobility in Evolving Internet Architecture*.

Jian, X.; Lian, X.; and Chen, L. 2018. On efficientl detecting overlapping communities over distributed dynamic graphs. In *Proceedings of ICDE Conference*.

Kapralov, M.; Lee, Y.; Musco, C.; Musco, C.; and Aaron, S. 2014. Single pass spectral sparsification in dynamic streams. In *Proceedings of FOCS Conference*, 561–570.

Kelner, J., and Levin, A. 2013. Spectral sparsification in the semi-streaming setting. *Theory of Computing Systems* 53(2):243–262.

Kim, S.; Nowozin, S.; Kohli, P.; and Yoo, C. 2011. Higher-order correlation clustering for image segmentation. In *Proceedings of NIPS Conference*, 1530–1538.

Lee, Y., and Sun, H. 2017. An SDP-based algorithm for linear-sized spectral sparsification. In *Proceedings of STOC Conference*, 678–687.

Lee, J.; Gharan, S.; and Trevisan, L. 2014. Multiway spectral partitioning and higher-order Cheeger inequalities. *Journal of the ACM* 61(6):37.

Li, M.; Miller, G.; and Peng, R. 2013. Iterative row sampling. In *Proceedings of FOCS Conference*, 127–136.

Maier, M.; Luxburg, U.; and Hein, M. 2009. Influence of graph construction on graph-based clustering measures. In *Proceedings of NIPS Conference*, 1025–1032.

Ng, A.; Jordan, M.; and Weiss, Y. 2001. On spectral clustering: analysis and an algorithm. In *Proceedings of NIPS Conference*, 849–856.

Peleg, D., and Schaffer, A. 1989. Graph spanners. *Journal of Graph Theory* 13(1):99–116.

Peng, R.; Sun, H.; and Zanetti, L. 2015. Partitioning well-clustered graphs: spectral clustering works! In *Proceedings of COLT Conference*, 1423–1455.

Phillips, J.; Verbin, E.; and Zhang, Q. 2016. Lower bounds for number-in-hand multiparty communication complexity, made easy. *SIAM Journal on Computing* 45(1):174–196.

Shi, J., and Malik, J. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8):888–905.

Spielman, D., and Srivastava, N. 2011. Graph sparsification by effective resistances. *SIAM Journal on Computing* 40(6):1913–1926.

Spielman, D., and Teng, S.-H. 2011. Spectral sparsification of graphs. *SIAM Journal on Computing* 40(4):981–1025.

Sun, H., and Zanetti, L. 2017. Distributed graph clustering and sparsification. *https://arxiv.org/abs/1711.01262*.

Tian, F.; Gao, B.; Cui, Q.; Chen, E.; and Liu, T.-Y. 2014. Learning deep representations for graph clustering. In *Proceedings of AAAI Conference*, 1293–1299.

Worldwidewebsize. 2018. http://www.worldwidewebsize.com/. [Online; accessed 05-07-2018].

Yang, W., and Xu, H. 2015. A divide and conquer framework for distributed graph clustering. In *Proceedings of ICML Conference*, 504–513.

Zhu, C., and Lam, K.-Y. 2017. Source-wise round-trip spanners. *Information Processing Letters* 124(C):42–45.

Zhu, C., and Lam, K.-Y. 2018. Deterministic improved round-trip spanners. *Information Processing Letters* 129:57–60.