Spectral Vertex Sparsifiers and Pair-Wise Spanners Over Distributed Graphs

Chun Jiang Zhu [1] **Qinqing Liu** [2] **Jinbo Bi** [2]

Abstract

Graph sparsification is a powerful tool to approximate an arbitrary graph and has been used in machine learning over graphs. As real-world networks are becoming very large and naturally distributed, distributed graph sparsification has drawn considerable attention. In this work, we design communication-efficient distributed algorithms for constructing spectral vertex sparsifiers, which closely preserve effective resistance distances on a subset of vertices of interest in the original graphs, under the well-established message passing communication model. We prove that the communication cost approximates the lower bound with only a small gap. We further provide algorithms for constructing pair-wise spanners which approximate the shortest distances between each pair of vertices in a target set, instead of all pairs, and incur communication costs that are much smaller than those of existing algorithms in the message passing model. Experiments are performed to validate the communication efficiency of the proposed algorithms under the guarantee that the constructed sparsifiers have a good approximation quality.

1. Introduction

Vertex sparsifiers and edge sparsifiers of a graph are a smaller graph that has a reduced number of vertices and a reduced number of edges respectively, but preserving important properties such as the spectral property, flow/cut values or distances of the input graph. Recently they have been used to improve the computational or communication complexity of machine learning methods (Chen et al., 2016; Loukas & Vandergheynst, 2018; Loukas, 2019; Zhu et al., 2019a; Bravo Hermsdorff & Gunderson, 2019; Rong et al., 2020; Zheng et al., 2020; Chen et al., 2020).

Given a graph $G(V, E)$ where V and E denote the sets of vertices and edges, a subset of *terminal* vertices $T \subseteq V$ represents the vertices of interest in the graph G. In practice, a community with a shared interest in social networks, or abnormal nodes in communication networks, can form the terminal set T for the respective graphs. The effective resistance between two vertices u and v is the voltage differential between them when we regard the graph as an electrical network of resistors with one unit of current injected at u and extracted at v. Effective resistances can be defined based on graph Laplacian and are useful in many applications that seek to cluster nodes in a network (von Luxburg et al., 2014). For example, they have deep connections to random walks on graphs (Tetali, 1991). The *Schur complement*[1] (Zhang, 2005) of the Laplacian L of G with respect to (w.r.t.) T is a pivotal concept, and can help calculate a *spectral vertex sparsifier* $H(T, E')$ of G w.r.t. T (Durfee et al., 2019) that preserves effective resistance between the vertices in T. The concept of Schur complement is also used in solving Laplacian linear systems $Lx = b$ when only a portion of b is concerned. Schur complements have found a wide range of applications including semi-supervised learning, Markov chains and finite-element analysis, and computer vision (Wagner et al., 2018; Kyng et al., 2016; Koutis et al., 2011; Dorfler & Bullo, 2013; Zhang, 2005).

Similar to Schur complements that preserve crucial properties only for pairs of terminals in T, another research thread works on constructing graph structures which preserve distances for pre-specified pairs of vertices, instead of all pairs. For a set of vertex pairs $P \subseteq V \times V$, a *P-pairwise spanner* $F(V, E' \subseteq E)$ of G only approximates distances between pairs $(u, v) \in P$ in G up to a small stretch. For example, a P-pairwise spanner of stretch $(+2)$ (or called $(+2)$-P-pairwise spanner) of G, is a subgraph F of G where the distance between every pair in P is at most their distance in G plus two. When P has certain special structures, there might be better trade-offs between the stretch and the size (the number of edges) of the spanner. There exist two main variants: *S-sourcewise spanners* for a subset of vertices

[1]Department of Computer Science, University of North Carolina at Greensboro, Greensboro, NC, USA [2]Department of Computer Science and Engineering, University of Connecticut, Storrs, CT, USA. Correspondence to: Chun Jiang Zhu <chunjiang.zhu@uncg.edu>, Jinbo Bi <jinbo.bi@uconn.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

[1]We will use Schur complements and spectral vertex sparsifiers interchangably, although strictly speaking, the former is the Laplacian of the latter.

$S \subseteq V$ called *sources* when $P = S \times V$, and T-*subsetwise spanners* for terminals T when $P = T \times T$. Pair-wise spanners and their variants were first studied in the theory community with applications in distributed computing, network routing, and computational biology (Coppersmith & Elkin, 2006; Cygan et al., 2013; Parter, 2014; Kavitha & Varma, 2015; Kavitha, 2017; Ahmed et al., 2020). They can be useful for learning over graphs such as graph neural networks (You et al., 2019; Yang et al., 2019), spectral methods (Chen et al., 2016; Zhu et al., 2019a), and a series of graph kernel methods based on shortest paths and connectivity (Costa & Grave, 2010; Shervashidze et al., 2011; Borgwardt & Kriegel, 2005; Hermansson et al., 2015).

However, graph-based learning on modern large-scale graphs imposes high computational and storage demands, which are too expensive, if not impossible, to meet by a single machine. Thus, distributed computing clusters and server storage are a popular and cost-effective way to meet the requirements (Balcan et al., 2013; 2014). Distributed graph sparsification has received considerable research interests, *e.g.*, (Chen et al., 2016; Woodruff & Zhang, 2017; Sun & Zanetti, 2019; Fernandez et al., 2020). In these works, the well-established *message passing* communication model represents a distributed system with point-to-point communication. There is a communication channel between each of s remote sites and a coordinator. Each site can send a message to another site by first sending to the coordinator, who then forwards the message to the destination. For an n-vertex graph $G(V, E)$, each site S_i holds a subset of edges $E_i \subset E$ on a common vertex set V and their union is $E = \cup_{i=1}^{s} E_i$. The model can be separated into the message passing *with and without duplication* models, based on whether two remote sites have edge duplicates. The major objective is to minimize the communication cost, which is usually measured by the total number of bits communicated.

Unfortunately, there is no known study of constructing Schur complements in any distributed computational model. The task appears challenging as it involves a matrix inversion which is inherently indecomposable. For pair-wise spanners and their variants, although they have been studied in the non-distributed model (Coppersmith & Elkin, 2006; Cygan et al., 2013; Kavitha & Varma, 2015; Kavitha, 2017) and the CONGEST distributed model (Censor-Hillel et al., 2018), they have not been systematically studied in the message passing model. The only existing work in the message passing model is a recent study on standard all-pairs spanners, which can imply pair-wise spanners (Fernandez et al., 2020). However, the communication costs $\tilde{O}(n^{3/2}s)$ ($\tilde{O}$ hides a poly-logarithmic factor) and $\tilde{O}(\sqrt{s}n^{3/2} + ns)$ of constructing standard spanners in the message passing models respectively with and without duplication could be very large, relatively to the situation when the cardinality of pairs $|P|$ is small.

Table 1. The communication costs of distributed algorithms for constructing approximate Schur complements. Given an n-vertex graph $G(V, E)$ with edges distributed at s remote sites, T is a subset of V and $B \subseteq V$ is the boundary vertex set (defined in Sec. 2). We have the following results for the message passing without duplication model.

LocalSS	LocalSC	Lower Bound
$\tilde{O}(ns)$ (Thm. 4)	$\tilde{O}(\|T \cup B\|s)$ (Thm. 8)	$\Omega(\|T\|s)$

Our Contributions. We initiate the investigation of distributed learning of Schur complements. (1) We propose two algorithms, namely the *local spectral sparsification* (LocalSS) and the *local Schur complement* (LocalSC) methods in the message passing without duplication model. The former utilizes the decomposability of spectral sparsifiers and incurs a communication cost $\tilde{O}(ns)$. Unfortunately, the cost does not depend on the size of the terminals $|T|$ and appears to be wasteful when $|T| \ll n$. This motivates the latter method that enjoys a communication cost $\tilde{O}(|T \cup B|s)$ (See Table 1), where B is the set of *boundary* vertices. Informally, a boundary vertex in B is a vertex with at least two of its edges linking to different sites, *e.g.*, a boundary router in communication networks and a highway hub in road networks, and usually $|B| \ll n$. Unlike spectral sparsifiers, Schur complement (w.r.t. T) is indecomposible. However, importantly, we discover and prove that after adding boundary vertices B to the terminal set, Schur complement w.r.t. $T \cup B$ becomes decomposible. This property immediately implies a distributed algorithm. Considering the wide utility of Schur complements, the decomposability of Schur complement w.r.t. $T \cup B$ can be of independent interest.

(2) We generalize the decomposability of spectral sparsifiers to the multi-graph setting. The property enables us to apply spectral sparsification to reduce the size of Schur complements to nearly linear from quadratic. We analyze that the communication lower bound of distributed Schur complement is $\Omega(|T|s)$ and the gap between the achieved upper bound and the lower bound is $\tilde{O}(|B|s)$ which can be small because in practice B is small. Finally, we show that the proposed algorithms can be extended to the dynamic setting.

(3) We propose a serials of distributed constructions of pair-wise spanners and the variants of different stretch-size trade-offs as summarized in Theorems 1, 2 and Corollary 1. We emphasize that all algorithms work for both the message passing with and without duplication models. In the message passing with duplication model, compared to the communication cost $\tilde{O}(n^{3/2}s)$ of the existing work (Fernandez et al., 2020), when $|S| < \sqrt{n}$ and $|\mathcal{H}(P)| < \sqrt{n}$, our algorithms for constructing S-sourcewise spanners and P-pairwise spanners incur a smaller communication complexity. In the message passing without duplication model, when $|S| < \sqrt{\frac{n}{s}}$ and $|\mathcal{H}(P)| < \sqrt{\frac{n}{s}}$, our algorithms incur

a communication cost smaller than $\tilde{O}(\sqrt{s}n^{3/2} + ns)$ of the existing method (Fernandez et al., 2020). Moreover, the sizes of the sourcewise and pairwise spanners match those of the same spanners constructed by a corresponding sequential algorithm up to a poly-logarithmic factor (Kavitha & Varma, 2015; Abboud & Bodwin, 2016; Kavitha, 2017).

Theorem 1 (**Sourcewise Spanner**). *For a graph $G(V, E)$ and a subset of source vertices $S \subseteq V$, there exists an algorithm that constructs a $(+2)$-S-sourcewise spanner of size $\tilde{O}(n^{5/4}|S|^{1/4})$ with high probability (w.h.p.) using communication cost $\tilde{O}(|S|ns)$.*

Theorem 2 (**Pairwise Spanner**). *For a graph $G(V, E)$ and a set of vertex pairs $P \subseteq V \times V$, there exists an algorithm that constructs a $(+2)$-P-pairwise spanner ($(+4)$-P-pairwise spanner) of size $\tilde{O}(n|P|^{1/3})$ (resp., $\tilde{O}(n|P|^{2/7})$) w.h.p. using communication cost $\tilde{O}(|\mathcal{H}(P)|ns)$, where $\mathcal{H}(P)$ is P's hitting set, $\mathcal{H}(P) = \{X \subseteq V \,|\, \forall (u, v) \in P, u \in X \text{ or } v \in X\}$.*

Corollary 1 (**Subsetwise Spanner**). *For a graph $G(V, E)$ and a subset of vertices $T \subseteq V$, there exists an algorithm that constructs a $(+2)$-T-subsetwise spanner ($(+4)$-T-subsetwise spanner) of size $\tilde{O}(n|T|^{2/3})$ (resp., $\tilde{O}(n|T|^{4/7})$) w.h.p. using communication cost $\tilde{O}(|T|ns)$.*

(4) Empirical studies are designed using both synthetic and real-world datasets to confirm that the proposed algorithms are effective in constructing Schur complements with a good approximation quality and also efficient communication in the message passing model.

Related Work. Schur complements are also known as Kron reduction, and have been compared to other graph reduction techniques (Loukas, 2019; Bravo Hermsdorff & Gunderson, 2019). However, it is unclear how these techniques can be adapted in the distributed environment. (Saad & Sosonkina, 1999) studied how to use Schur complements to solve a distributed sparse linear system, but they did not work on distributed constructions of Schur complements. There exist several works on using edge sparsifiers in graph-based learning (Rong et al., 2020; Zheng et al., 2020; Chen et al., 2020), though they are designed neither for vertex sparsifiers nor the distributed setting. Exact pair-wise and source-wise spanners (called preservers) were firstly studied in (Coppersmith & Elkin, 2006; Bollobas et al., 2005) and then in (Bodwin & Williams, 2016; Bodwin, 2017). Further studies focused on approximate pair-wise and source-wise spanners with different stretch *e.g.*, $(+2)$, $(+4)$ and $(+6)$ (Cygan et al., 2013; Parter, 2014; Kavitha, 2017; Abboud & Bodwin, 2016). (Abboud & Bodwin, 2016) considered lower bounds for pair-wise spanners and their connection with lower bounds for pair-wise preservers. (Censor-Hillel et al., 2018) studied distributed constructions of pair-wise spanners in the CONGEST model, but they are not for the message passing model. (Forster et al., 2021) proposed a communication-efficient algorithm in the CONGEST model (not in message passing model) for solving graph Laplacian linear systems. It is based on an algorithm for constructing approximate minor Schur complements, and is a randomized algorithm. However, our algorithm is deterministic and can employ a deterministic algorithm for constructing spectral sparsifiers, *e.g.*, (Baston et al., 2012). Their method aims to parallelize the Li-Schild algorithm (Li & Schild, 2018) by replacing the random spanner trees that cannot be implemented in parallel. Particularly, it identifies a subset of (steady) edges that can be sampled independently of each other. In contrast, we study and exploit the decomposability theorem of Schur complements. A more comprehensive set of related works is given in the Appendix.

2. Distributed Schur Complements

In this section, we present our first set of results, distributed spectral vertex sparsification. We first formally define the notations and the studied problem, and then propose two distributed algorithms for constructing approximate Schur complements and analyze their communication complexity. Finally, we provide distributed algorithms in the dynamic setting. All missing proofs can be found in the Appendix.

Notations and Definitions. We consider a weighted undirected graph $G(V, E, W)$, where $n = |V|$, $m = |E|$ and $W(e)$ defines the weight of each edge $e \in E$. Let A_G be the adjacency matrix of G, *i.e.*, $(A_G)_{u,v} = W(u, v)$ if $(u, v) \in E$ and zero otherwise. Let the weighted degree of a vertex $u \in V$, $d(u) = \sum_{v \in V} W(u, v)$. Let D_G be the degree matrix of G defined as $(D_G)_{u,u} = d(u)$, and zero otherwise. The *Laplacian matrix* of G is $L_G = D_G - A_G$. For an arbitrary vertex set $X \subseteq V$ and $\bar{X} = V - X$ in G, L_G can be represented as block matrices: $L_G = \begin{bmatrix} L_{[\bar{X},\bar{X}]} & L_{[\bar{X},X]} \\ L_{[X,\bar{X}]} & L_{[X,X]} \end{bmatrix}$. The *Schur complement* of G w.r.t. X, denoted as $SC(G, X)$, is the matrix obtained by eliminating vertices in $\bar{X}$ using Gaussian elimination (Zhang, 2005). Its closed-form is $SC(G, X) = L_{[X,X]} - L_{[X,\bar{X}]} L^{-1}_{[\bar{X},\bar{X}]} L_{[\bar{X},X]}$. A $(1 + \epsilon)$-$SC(G, T)$, H, is an approximate Schur complement of G w.r.t. T such that for every $x \in R^{|T|}$, it holds that $(1-\epsilon)x^T \cdot SC(G, T) \cdot x \leq x^T L_H x \leq (1+\epsilon)x^T \cdot SC(G, T) \cdot x$. In the message passing without duplication model, there are s remote sites $S_1, \cdots, S_s$ and a coordinator. Each site S_i holds a subset of edges $E_i \subset E$ on a common vertex set V, defining its local graph $G_i(V, E_i)$. It holds that $E = \cup_{i=1}^{s} E_i$, and $E_i \cap E_j = \emptyset$ for $1 \leq i \neq j \leq s$. We study the problem of reducing the communication cost for distributively constructing a $(1 + \epsilon)$-$SC(G, T)$ in the coordinator.

Local Spectral Sparsification Method. Considering an arbitrary graph problem P, a centralized method is to transmit edges in all local graphs G_i to the coordinator, and then apply any centralized algorithm for P in G. However, this

method incurs a communication cost $O(m)$, which could be prohibitively large for large-scale graphs. In contrast, a principal method is that every site S_i performs *local* computations to get a succinct *synopsis* H_i of its local graph G_i, and then only transmits H_i, instead of G_i, to the coordinator (Chen et al., 2016; Zhu et al., 2019b). If H_i has a much smaller number of edges than G_i, the communication cost can be significantly reduced.

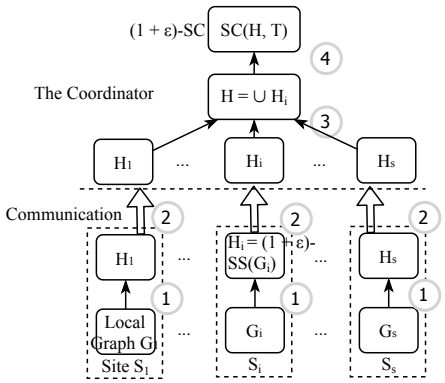


Figure 1. The idea of the local spectral sparsification for constructing $(1+\epsilon)$-SC.

We will follow this strategy, and start by introducing the notion of spectral sparsification: a $(1+\epsilon)$-*spectral sparsifier* of G, denoted as $(1+\epsilon)$-$SS(G)$, 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. It has wide usage in graph clustering, linear system solvers, and semi-supervised learning (Calandriello et al., 2018; Chen et al., 2016; Zhu et al., 2019a). Because spectral sparsifiers can well approximate the Laplacian matrix of the original graphs, based on which the Schur complement is calculated, we propose to first construct a spectral sparsifier H of G in the coordinator and then construct $SC(H,T)$ as an approximate Schur complement (Fig. 1). Specifically, every site S_i constructs a spectral sparsifier of its local graph G_i as the synopsis H_i and then transmits H_i to the coordinator. Upon receiving H_i from all sites, the coordinator first takes their union, $H = \cup_{i=1}^{s} H_i$, and then constructs $SC(H,T)$. The distributed construction of a spectral sparsifier in the coordinator was introduced in (Abraham et al., 2016; Chen et al., 2016). It is based on the *decomposability* property of spectral sparsifiers (Thm. 3): the union of spectral sparsifers of a subgraph of G is a spectral sparsifier of G. Because each sparsifier H_i has size $O(n)$, this method incurs a communication cost $\tilde{O}(ns)$. However, the cost could be large, relatively when the number of terminals is much smaller than $|V|$, *i.e.*, $|T| \ll n$.

Theorem 3 (Decomposability of Spectral Sparsifiers (Abraham et al., 2016)). *Let $E_1, \cdots, E_s$ be any partition of the set of edges E in a weighted undirected graph $G(V,E)$. It holds that $\cup_{i=1}^{s} SS(G_i(V,E_i)) = SS(G)$.*

Theorem 4. *For a subset of terminals $T \subseteq V$ in a graph $G(V,E)$, using communication cost $\tilde{O}(ns)$ the local spectral sparsification method outputs a $(1+\epsilon)$-$SC(G,T)$.*

Local Schur Complement Method. It is natural to consider directly using the Schur complement as the synopsis H_i of the local graph G_i. That is, each site S_i constructs $H_i = SC(G_i, T)$ and then transmits it to the coordinator, who then takes the union of all the received H_i, $H = \cup_{i=1}^{s} H_i$. However, we prove that unlike spectral sparsifiers, the Schur complement does not satisfy the *decomposability* property in Theorem 5. The main reason is that the matrix inversion in the definition of Schur complement does not satisfy the decomposability property. Hence we have $H \neq SC(G,T)$, rendering impossible to employ the Schur complements of local graphs.

Theorem 5 (Indecomposability of Schur Complements). *Let $E_1, \cdots, E_s$ be any partition of the edge set E in a graph $G(V,E)$. There exists a subset of vertices $T \subseteq V$ such that $\cup_{i=1}^{s} H_i \neq SC(G,T)$, where $H_i = SC(G_i(V,E_i),T)$.*

Proof. By definition, for every $i \in [1,s]$ we have

$$H_{1i} = SC(G_i,T) = L_{G_i[T,T]} - L_{G_i[T,\bar{T}]} L^{-1}_{G_i[\bar{T},\bar{T}]} L_{G_i[\bar{T},T]}.$$

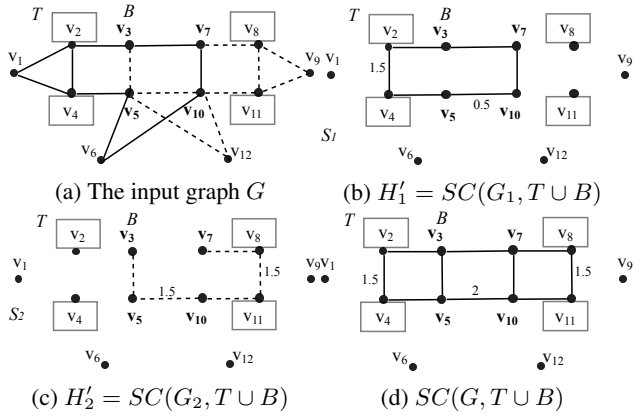
Summing all equalities for $i \in [1,s]$, we get that

$$\cup_{i=1}^{s} H_{1i} = L_{G[T,T]} - \sum_{i=1}^{k} L_{G_i[T,\bar{T}]} L^{-1}_{G_i[\bar{T},\bar{T}]} L_{G_i[\bar{T},T]}.$$

Because at the right hand sides $\sum_{i=1}^{k} L_{G_i[T,\bar{T}]} L^{-1}_{G_i[\bar{T},\bar{T}]} L_{G_i[\bar{T},T]} \neq L_{G[T,\bar{T}]} L^{-1}_{G[\bar{T},\bar{T}]} L_{G[\bar{T},T]}$, we have $\cup_{i=1}^{s} H_{1i} \neq SC(G,T)$. □

We now introduce the notion of boundary vertices: a vertex $v \in V$ is called a *boundary* vertex if v has two edges observed at two different sites. Formally, the boundary vertex set of G, $B = \{v \,|\, \exists (u,v) \in E_i, (v,w) \in E_j \; s.t. \; i \neq j\}$. Boundary vertices are well motivated in practice, *e.g.*, boundary routers for sending cross-region packages in communication networks, members in charge of outreach activities in social networks, and highway hubs connecting to another region in road networks. Because in practice the number of boundary vertices in a network is much smaller than the total number of vertices, we assume that $|B| \ll n$.

We make an interesting observation that after taking the union of boundary vertices B and terminals T as the vertex set w.r.t. which a Schure complement of G_i is constructed, the Schur complement will satisfy the decomposability property. Specifically, each site S_i constructs $H_i = SC(G_i, T \cup B)$ and then transmits it to the coordinator. Upon receiving all the H_i, the coordinator takes their union $H = \cup_{i=1}^{s} H_i$. Note that the SC operator can introduce multiple edges between the same two vertices. Here



(a) The input graph G (b) $H_1' = SC(G_1, T \cup B)$

(c) $H_2' = SC(G_2, T \cup B)$ (d) $SC(G, T \cup B)$

Figure 2. An illustrating example for the decomposabiity in Thm. 6. All edge weights are one unless stated explicitly. In Fig. 2a, the edges of G are distributed at two sites: solid edges are in the site S_1 and dash edges are in S_2. The terminals in $T = \{v_2, v_4, v_8, v_{11}\}$ are surrounded by a rectangle. The boundary vertices in $B = \{v_3, v_5, v_7, v_{10}\}$ are in bold. In Fig. 2b and 2c, we plot the Schur complements $H_1' = SC(G_1, T \cup B)$ and $H_2' = SC(G_2, T \cup B)$ at the sites S_1 and S_2, respectively. Fig. 2d shows that the union of H_1' and H_2' is equal to $SC(G, T \cup B)$.

the union of two parallel edges $e_1(u, v)$ of weight $W(e_1)$ and $e_2(u, v)$ of weight $W(e_2)$ is an edge (u, v) of weight $W(e_1) + W(e_2)$. Then it is guaranteed that H is equal to $SC(G, T \cup B)$. The decomposability is summarized in Thm. 6 and an illustrating example is provided in Fig. 2.

Theorem 6. *Under the notations of Thm. 5, let $B \subseteq V$ be the set of boundary vertices in G. For all subsets of vertices $T \subseteq V$, it holds that $\cup_{i=1}^{s} H_i' = SC(G, T \cup B)$, where $H_i' = SC(G_i, T \cup B)$ for $1 \leq i \leq s$.*

We rigorously prove Thm. 6 by using the *star-mesh* operator for constructing Schur complements (Zhang, 2005). The operator is performed at each vertex v not in $T \cup B$ and basically removes v and then updates the edges between each pair of its neighbors. For every pair of v's original neighbors u, w, the star-mesh transform adds an edge (u, w) with weight $W(u, v)W(v, w)/d(v)$ to E if $(u, w) \notin E$, or otherwise increases $W(u, w)$ by $W(u, v)W(v, w)/d(v)$. As in Lemma 1, it has been proved that one can apply the star-mesh operator in the vertices in $V - T - B$ with an arbitrary order to calculate $SC(G, T \cup B)$ (Zhang, 2005). Then it suffices to prove that the union set of star-mesh operators for constructing $\cup_{i=1}^{s} H_i'$ and the set for constructing $SC(G, T \cup B)$ are equivalent. We emphasize that Thm. 6 holds for Schur complements constructed by an arbitrary algorithm, although we prove it using the star-mesh transforms method.

Lemma 1. *(Zhang, 2005) For a vertex set $V' \subseteq V$ in a graph $G(V, E)$, $SC(G, V')$ can be computed by sequentially performing star-mesh transforms on the vertices in $V - V'$ in an arbitrary order.*

Proof. (Theorem 6) Let $O_1, \cdots, O_r$ for $r = |R|$ be any sequence of the vertices in $R = V - T - B$ that is to be reduced by star-mesh transforms to compute $SC(G, T \cup B)$ (suppose that G is directly available in the coordinator). Because each vertex v in R is not a boundary vertex, all its edges will be observed only at a single site which we denote as $S(v)$. Let $V(S_i)$ of each site S_i be the vertices v in R with $S(v)$ being S_i, *i.e.*, $V(S_i) = \{v \in R \,|\, S(v) = S_i\}$. We can easily get that $V(S_1), \cdots, V(S_s)$ are a partitioning of the vertex set R, *i.e.*, $\cup_{i=1}^{s} V(S_i) = R$ and $V(S_i) \cap V(S_j) = \emptyset$ for $1 \leq i < j \leq s$. Further let $O_{i_1}, \cdots, O_{i_r}$ be any sequence of the vertices in R that is to be reduced by star-mesh transforms to compute $H_i' = SC(G_i, T \cup B)$ at each site S_i. It suffices to prove that the sequences $O_1, \cdots, O_r$ and $\cup_{i=1}^{s}(O_{i_1}, \cdots, O_{i_r})$ are equivalent in constructing Schur complements.

We observe that for each vertex u in $R - V(S_i)$ at site S_i, the star-mesh transform does nothing because u has no edges in the local graph G_i. We then remove all vertices in $R - V(S_i)$ from the sequence $O_{i_1}, \cdots, O_{i_r}$ and only keep the vertices in $V(S_i)$ in the sequence. This results in an effective sub-sequence $O_{i_1'}, \cdots, O_{i_{h_i}'}$ at S_i, where $h_i = |V(S_i)|$ and each operation in the sequence corresponds to a vertex in $V(S_i)$. Because $V(S_1), \cdots, V(S_s)$ are a partitioning of the vertex set R, it is easy to see the two sets $\{O_1, \cdots, O_r\}$ and $\{\cup_{i=1}^{s}(O_{i_1'}, \cdots, O_{i_{h_i}'})\}$ which are induced by the respective sequence are the same. Based on Lemma 1 and the fact that the union in $\cup_{i=1}^{s} H_i'$ takes summation of the weights of multiple edges between the same two vertices as the new edge weight, we have the two sequences $O_1, \cdots, O_r$ and $\cup_{i=1}^{s}(O_{i_1'}, \cdots, O_{i_{h_i}'})$ are equivalent in constructing Schur complements. This implies that $O_1, \cdots, O_r$ and $\cup_{i=1}^{s}(O_{i_1}, \cdots, O_{i_r})$ are equivalent, completing the proof. □

Algorithm 1 *LocalSC*

Input: Local graph G_i at each site S_i, terminals T, boundary vertices B, and a parameter ϵ
Output: $(1+\epsilon)$-$SC(G, T)$
1: **for** site S_i **do**
2: Constructs $H_i' = SC(G_i, T \cup B)$;
3: Constructs $H_i'' = (1+\epsilon)$-$SS(H_i')$ by any standard spectral sparsification algorithm, and then transmits it to the coordinator;
4: Upon receiving all H_i'', the coordinator takes their union $H'' = \cup_{i=1}^{s} H_i''$ and then returns $SC(H'', T)$;

Sparsification of Multi-Graphs. A potential problem is that the constructed structure $H_i' = SC(G_i, T \cup B)$ at each site can be dense and have up to $O(|T \cup B|^2)$ edges. It can be too dense to be leveraged as a sparsifier, and can also incur a large communication cost. One may consider to apply spectral sparsification on each H_i' to reduce its size. However, the SC operation could add multiple edges incident to the same two vertices in H_i' at different sites.

For example in Fig. 2b and 2c, H_1' contains edge (v_5, v_{10}) of weight 0.5 while H_2' contains edge (v_5, v_{10}) of weight 1.5. This prevents us from using Thm. 3, which does not hold for multi-graphs, to prove the quality of the union of spectral sparsifiers of local graphs.

We first generalize the decomposability of spectral sparsifiers for simple graphs (Thm. 3) to multi-graphs (Thm. 7). At the first glance, it appears challenging because parallel edges in multi-graphs will be processed and sampled independently. With a careful analysis of the union operation, we are able to prove the decomposability in the multi-graph setting. We further propose to apply a $(1+\epsilon)$-spectral sparsification in H_i' to get $H_i'' = (1+\epsilon)\text{-}SS(SC(G_i, T\cup B))$ and then transmit H_i''. This can reduce the size of the sparsifer from quadratic to nearly linear, $\tilde{O}(|T \cup B|)$. More importantly, because each site S_i transmits H_i'' of size $\tilde{O}(|T\cup B|)$ to the coordinator, the communication cost of this approach is only $\tilde{O}(|T \cup B|s)$.

Theorem 7 (Decomposability of Spectral Sparsifiers for Multi-Graphs). *Let $E_1, \cdots, E_s$ be any partition of the set of edges E in a weighted undirected multi-graph $G(V, E)$. It holds that $\cup_{i=1}^{s} SS(G_i(V, E_i)) = SS(G)$.*

Theorem 8. *For a subset of terminals $T \subseteq V$ and boundary vertices $B \subseteq V$ in a graph $G(V, E)$, Algorithm 1 constructs a $(1+\epsilon)$-$SC(G, T)$ using communication cost $\tilde{O}(|T\cup B|s)$, which is close to the communication lower bound $\Omega(|T|s)$.*

When all the vertices in a graph are included in T (*i.e.*, $|T| = n$), the problem degenerates to distributed spectral sparsification problem, which has a communication lower bound $\Omega(ns) = \Omega(|T|s)$ (Chen et al., 2016). Because in practice the number of boundary vertices $|B|$ is usually small, the gap between Alg. 1's upper bound $\tilde{O}(|T \cup B|s)$ and this lower bound is small.

Extensions to Dynamic Graphs. Most of the massive graphs in the real world, such as social networks and the Web graph, are subject to frequent changes over time. We show that Alg. 1 can be adapted to a dynamic graph undergoing both edge insertions and deletions. Given an initial graph, each site S_i first initializes a (dynamic) Schur complement (Durfee et al., 2019) in $\tilde{O}(m\epsilon^{-4})$ time. Then for the updates observed locally at a subsequent time point, S_i maintains a Schur complement H_i' of size $\tilde{O}(m\epsilon^{-2})$ for the graph at the time point dynamically in $\tilde{O}(1)$ amortized time per update (Durfee et al., 2019). Next, it utilizes the dynamic spectral sparsification algorithm (Abraham et al., 2016) to maintain for H_i' a sparsifier H_i'' of size $\tilde{O}(t\epsilon^{-2})$ in $\tilde{O}(\epsilon^{-2})$ amortized update time. Finally, S_i transmits only the updates of H_i'' compared to the previous time point to the coordinator. Assuming that the number of updated edges in H_i'' is roughly linear in the update time, the amortized communication cost per edge update is $\tilde{O}(\epsilon^{-2})$.

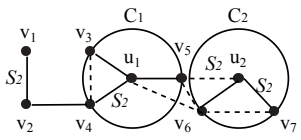
Challenges in the Blackboard Model. Consider another communication model called the blackboard model, where there is a broadcast channel. We can still construct SC of the local graph G_i w.r.t. $T \cup B$, $H_i' = SC(G_i, T \cup B)$, at each site S_i as in Step 2 of Alg. 1. However, we cannot employ existing spectral sparsification algorithms (Chen et al., 2016; Zhu et al., 2019b) to construct $(1+\epsilon)$-$SS(\cup_{i=1}^{s} H_i')$ using communication cost of $\tilde{O}(|T \cup B| + s)$. This is because the algorithms do not work for multi-graphs. It is unclear how the sampling-based algorithms perform sampling on multiple edges incident to the same two vertices at different sites. Therefore, it is an open problem to construct a $(1+\epsilon)$-$SC(G, T)$ in the blackboard model using communication cost $o(|T \cup B|^2 s, n + s)$, where the latter comes from the simple method that extends the local spectral sparsification method to this model. Note that a trivial lower bound is $\Omega(|T| + s)$. This further shows the importance of the generalization of sparsification to multi-graphs in the message passing model as in Thm. 7. More discussions can be found in the Appendix.

3. Distributed Pair-Wise Spanners

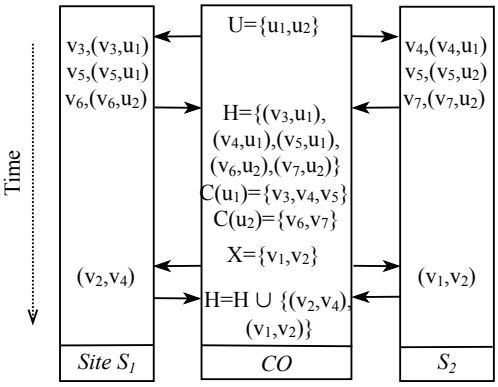
In this section, we present the second group of results, distributed algorithms for constructing pair-wise spanners and the variants. We first define notations we will use, and then outline high-level ideas of the proposed distributed algorithms. The missing algorithms and the proofs can be found in the Appendix.

Notations. In a graph $G(V, E)$, the neighbors of a vertex u in V are denoted as $N(u) = \{v \,|\, (u, v) \in E\}$, and the neighbors of a subset of vertices $U \subseteq V$ are $N(U) = \cup_{v\in U} N(v)$. Let $E(u, U) = \{(u, v) \,|\, (u, v) \in E \text{ and } v \in U\}$, and $e(u, U)$ be an arbitrary edge in $E(u, U)$. For a subset of vertices U', $e(U', U) = \cup_{u\in U'} e(u, U)$. For a subset of edges $F \subseteq E$, let $E(u, U, F) = E(u, U) \cap F$ and $e(u, U, F)$ be an arbitrary edge in $E(u, U, F)$. The distance between two vertices $u, v \in V$ in a subgraph G' of G is denoted by $d(u, v, G')$ and the shortest path realizing it is $\Pi(u, v, G')$, where ties are broken arbitrarily. The function $Send(X, Dest)$ sends data X to the destination $Dest$ and $Dest$ can be the coordinator (CO) or all sites (AS). Similarily, $Send(X, Y, Dest)$ sends data X and Y to $Dest$. All steps in the pseudo-code are executed in CO, unless stated explicitly in AS.

Overview. The proposed algorithms work for both the message passing with and without duplication models, and are inspired by existing pair-wise and source-wise spanners (Kavitha & Varma, 2015; Kavitha, 2017; Censor-Hillel et al., 2018). Basically there are three procedures, the clustering, the breadth-first search (BFS) trees constructions and the path-buying procedure. Given a set of cluster centers, the clustering procedure organizes vertices of the input graph



(a) A graph observed at two sites and a clustering $C = \{C_1, C_2\}$ for cluster centers $U = \{u_1, u_2\}$. Edges marked by S_2 are at site S_2 and otherwise at S_1. Solid edges are included in H while dash edges are in G.



(b) distributed clustering

Algorithm 2 *Clustering*

Input: A vertex set U and a graph G
Output: A set C of clusters and edge set H

$C = \{C(u) = \emptyset \mid u \in U\}$; $H = \emptyset$;
$Send(U, AS)$;
AS: $Send(N(U), e(N(U), U), CO)$;
$N = \cup_{AS}(N(U))$;
for each vertex $v \in N$ **do**
  $(u, v) = e(v, U)$;
  $H = H \cup (u, v)$;
  $C(u) = C(u) \cup v$;
$X = V - N - U$; $Send(X, AS)$;
AS: $Send(E(X, V), CO)$;
$H = H \cup (\cup_{AS} E(X, V))$;
return (C, H);

Figure 4. The clustering algorithm and an illustrating example.

into clusters of radius one around the cluster centers and there is a set of unclustered vertices. For example in Fig. 3a, there are two clusters C_1 and C_2 and v_1 and v_2 are unclustered. For each cluster C_i, edges connecting its cluster center and its vertices are added to the spanner H, along with all edges of the unclustered vertices (See the solid edges). With the edges in C_i, the distance between any two vertices in C_i can be well approximated. More importantly, if the distance between a vertex v outside C_i and a vertex in C_i is known to be preserved in H, the distances between v and all other vertices in C_i are immediately well approximated with a stretch of at most two.

Next, the BFS trees rooted at a sampled subset of cluster centers are added to the spanner H. Consider a shortest path $\Pi(u, v, G)$ we need to approximate for $u \in S$, $v \in V$ in an S-sourcewise spanner, or $(u, v) \in P$ in a P-pairwise spanner. When $\Pi(u, v, G)$ contains a large number of missing edges in H, it will pass through a large number of clusters (because all edges of unclustered vertices are already in H) and thus w.h.p. it will pass through a cluster C_i whose cluster center is sampled to grow a BFS tree T and all edges of the tree are added to H. Then $\Pi(u, v, G)$ is well approximated by the BFS tree T and edges of the cluster C_i. When $\Pi(u, v, G)$ contains only a few missing edges in H, we are afford to add all the missing edges or a carefully selected subset of edges to H through a path-buying procedure. We will discuss the cluster procedure and the path-buying procedure for source-wise spanners. The construction of a BFS tree from a given root is already available in (Fernandez et al., 2020).

The Clustering Procedure. Given a set of cluster centers U, the coordinator sends the centers U to all remote sites. Upon receiving U, each site sends U's neighbors $N(U)$ in its local graph together with an arbitrary edge from each neighbor in $N(U)$ to U, $e(N(U), U)$, to the coordinator. The coordinator then takes union N of the neighbors at all sites, and assign each vertex v in N to the cluster $C(u)$ of an arbitrary neighboring cluster center u, and also add the edge (u, v) to the spanner H. Finally, the coordinator requests the edges of unclustered vertices $X = V - N - U$ from all sites and then adds them to H. The algorithm is provided in Alg. 2 and an example is given in Fig. 4.

The Path-Buying Procedure. The path-buying method was introduced in (Baswana et al., 2010) and has been used in several spanner algorithms (Cygan et al., 2013; Parter, 2014; Kavitha & Varma, 2015). However, it is unknown how to implement this procedure in the message passing model. For S-sourcewise spanners [2] , our path-buying procedure adds, for every pair of a source vertex $u \in S$ and a cluster $C_i \in C$, the shortest path between them to an edge set F, if it contains at most f missing edges in H. Specifically, for each vertex u in S, the coordinator grows a BFS-like tree T from u. It maintains a current layer L of vertices, initialized to contain only u, and repeats the following until all clusters in C have been reached or L is empty.

In each iteration, the coordinator sends L to every remote site, who maintains the set of visited vertices L^* by adding L to it and then sends L's unvisited neighbors $N(L) - L^*$ and edges $e(N(L) - L^*, L)$ back to the coordinator. Upon receiving all L's neighbors and edges, the coordinator takes union of the neighbors $N = \cup_{AS}(N(L) - L^*)$ as vertices in the next layer. For each vertex v in N, the coordinator adds v and its edge already in H, $e(v, L, H)$ if such an edge exists and otherwise $e(v, L)$ to T. Giving a priority to edges in H can reduce the size of the constructed spanner. If $\Pi(u, v, T)$ already has more than f missing edges in H, v can be safely pruned from the next layer N because all potential paths through v have more than f missing edges. Otherwise, it checks if the cluster $C(v)$ of v has not been reached before, and maintains F and terminates the construction of T if necessary. Finally, N is assigned to L as the current layer. The algorithm is provided in Alg. 3.

Discussions. For constructing $(+4)$-P-pairwise spanners,

[2] In principle the path-buying procedures for the pair-wise spanners are similar to that of source-wise spanners.

Algorithm 3 *PathBuying-S*

Input: A vertex set S, a graph G, an edge set H, a parameter f and a set C of clusters
Output: An edge set F

$F = \emptyset$;
for each vertex $u \in S$ **do**
  $L = \{u\}, T = (\{u\}, \emptyset)$; AS: $L^* = \emptyset$;
  while $L \neq \emptyset$ **do**
    $Send(L, AS)$;
    AS: $L^* = L^* \cup L$, $Send(N(L) - L^*, e(N(L) - L^*, L), CO)$;
    $N = \cup_{AS}(N(L) - L^*)$;
    for each vertex $v \in N$ **do**
      if $e(v, L, H) \neq \emptyset$ **then**
        $T = T \cup (v, e(v, L, H))$;
      else
        $T = T \cup (v, e(v, L))$;
      if $|\{e \mid e \in \Pi(u, v, T)$ and $e \notin H\}| > f$ **then**
        $N = N - \{v\}$; **Continue**;
      if $C(v)$ has not been reached before **then**
        $F = F \cup \Pi(u, v, T)$;
        if all clusters in C have been reached **then**
          Go to Line 2;
    $L = N$;
return F;

there is an additional prefix-suffix-buying procedure before the path-buying procedure: for every pair (u, v) in P, it adds the first l and the last l missing edges in $\Pi(u, v, G)$ to an edge set F. This can be achieved by growing a BFS-like tree from each vertex u in P's hitting set, $\mathcal{H}(P) = \{X \subseteq V \mid \forall (u, v) \in P, u \in X$ or $v \in X\}$ and when a vertex v which forms a pair with u in P is reached, maintaining F accordingly. With the algorithms for constructing pair-wise spanners in Thm. 2, we immediately get algorithms for constructing T-subsetwise spanners (See Cor. 1). A $(+2)$-T-subsetwise and $(+4)$-subsetwise-spanner of G can be constructed by setting $P = T \times T$ and then applying a $(+2)$-P-pairwise and $(+4)$-P-pairwise spanner in G, resp.

The communication costs of our algorithms for constructing S-soursewise spanners and P-pairwise spanners are $\tilde{O}(|S|ns)$ and $\tilde{O}(|\mathcal{H}(P)|ns)$, respectively. As mentioned earlier, when the size of S and $\mathcal{H}(P)$ are small, i.e., smaller than $\sqrt{n}$ and $\sqrt{\frac{n}{s}}$ under the message passing with and without duplication models resp., the communication costs are smaller than those of the existing algorithms (Fernandez et al., 2020), resp. $\tilde{O}(n^{3/2}s)$ and $\tilde{O}(\sqrt{s}n^{3/2} + ns)$. The communication cost in the without duplication model might be smaller than that in the with duplication model because edge duplicates incur additional communications. We leave this improvement as an important future work.

Compared to the sequential algorithms (Kavitha & Varma, 2015; Abboud & Bodwin, 2016; Kavitha, 2017), we stress that our distributed algorithms construct source-wise and pair-wise spanners of the same number of edges up to only a poly-logarithmic factor. One may notice that 0-S-sourcewise and 0-P-pairwise spanners H of G can be constructed by growing a BFS tree from each vertex in S and $\mathcal{H}(P)$ in G resp. and then adding all edges in the BFS trees to H. Our algorithms have the same asymptotical communication costs as these algorithms. However, ours achieve a significantly smaller number of edges in the spanner (at the expense of a slightly larger stretch). To see this, the sizes of the above pair-wise spanners of stretch 0, and our spanners of stretch $(+2)$ and $(+4)$ are $\tilde{O}(n|\mathcal{H}(P)|) = \tilde{O}(n|P|)$, $\tilde{O}(n|P|^{1/3})$, and $\tilde{O}(n|P|^{2/7})$, respectively. Similarly, the sizes of the above source-wise spanner of stretch 0 and our spanner of stretch $(+2)$ are $\tilde{O}(n|S|)$ and $\tilde{O}(n^{5/4}|S|^{1/4})$, respectively. Therefore, our algorithms provide a smooth trade-off between the stretch and the size of a spanner.

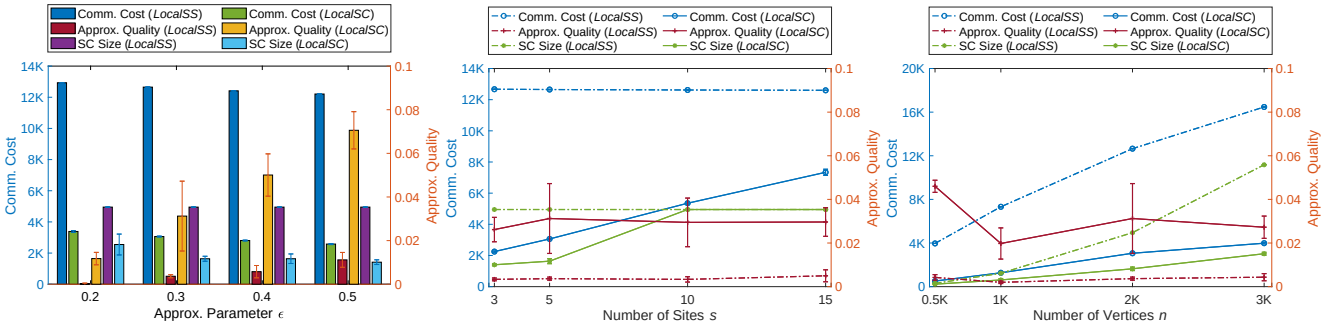
4. Experiments

In this section, we empirically evaluate the proposed algorithms for constructing Schur complements. The algorithms were implemented using Matlab and Julia programs, and all experiments were performed in a machine with Intel i7-9750H 2.6GHz CPU and 16G RAM.

Datasets and Methods. We use two synthetic datasets, *Circles* and *Gaussians*, and four real-world datasets *Sculpture*, *Sculpture-1M*, *Sculpture-11M*, and *Beach*. In the *Circles* dataset consisting of $2K$ vertices and about $17K$ edges, the vertices are sampled from two circles of the same origin but different radii 1 and 1.05, respectively. We connect two vertices if they are mutually k-nearest neighbors of each other for $k = 20$, and use the standard RBF similarity $W(u, v) = exp\{-||u - v||_2^2/2\sigma^2\}$ for $\sigma = 10$. For the *Beach* dataset, we use a 524×88 version of a beach photo resulting in $46,112$ vertices and $73,159,983$ edges with $k = 4000$ and $\sigma = 5$. The statistics of other datasets can be found in the Appendix. Each graph edge is assigned to a site S_i for $i \in [1, s]$ by a rule described in the Appendix. We independently sample each vertex in the graph with probably r and then use the sampled vertices as the terminals. We independently sample each vertex in the graph with probably r and then use the sampled vertices as the terminals.

For the spectral sparsification, we employ the implementation of Spielman and Srivastava (Spielman & Srivastava, 2011) [3]. The computation of approximation quality for checking how well one graph approximates another is also from the same code. For simplicity, the communication cost is the total number of edges communicated, which approximates the total number of bits by a logarithmic factor. The size of the constructed Schur complement (or called *SC size*) is also recorded as it will affect the computational complexity of the subsequent tasks. All performance measures are averaged over five runs and reported together with their standard deviation.

[3] github.com/danspielman/Laplacians.jl



(a) The baseline results for *Circles* (b) Effects of varied s (c) Effects of varied n

Figure 5. The experimental results on a synthetic dataset *Circles*.

Results. In the baseline setting, the number of sites $s = 5$, the sampling rate $r = 0.05$, (*i.e.*, $|T| = 0.05n$) and the approximation parameter $\epsilon = 0.3$. As shown in Fig. 5a, for the *Circles* dataset, both *LocalSS* and *LocalSC* have communication costs smaller than the centralized method ($\sim 17K$). The size of the boundary vertices $|B| = 107$, the average size of $|T \cup B| = 201$, and both are much smaller than $n = 2K$. Then the communication costs of *LocalSC* are consistently significantly smaller than those of *LocalSS*, matching the theoretical analysis. When ϵ increases, the communication costs of both methods decrease as a worse approximation is allowed. Although *LocalSC* has a relatively worse approximation quality than *LocalSS*, both methods result in an error always smaller than 0.07, much better than the required quality $\epsilon = [0.2, 0.5]$. Moreover, *LocalSC* has a significantly smaller SC size than *LocalSS*, which is favorable for subsequent applications.

In the *Gaussians* and *Sculpture* datasets, the observations are generally similar except when ϵ is small, *e.g.* 0.2, the communication cost of *LocalSC* is larger than that of *LocalSS*. This may be due to that in *LocalSC* the Schur complements constructed at all sites have edge duplicates. When ϵ becomes larger, the cost of *LocalSC* becomes much smaller than that of *LocalSS* again. As shown in Table 2 and 3, for larger datasets *LocalSC* consistently achieves small communication cost and low approximation error (in comparison with the centralized method where all m edges in the original graph are communicated). The advantages of communication cost becomes more obvious in larger datasets.

On the *Circles* dataset, we also evaluate the effects when the number of sites increases but other parameters remain the same. Fig. 5b shows that the communication cost of *LocalSC* increases faster than that of *LocalSS*. We observe that in the theoretical communication cost $\tilde{O}(|T \cup B|s)$ of *LocalSC*, when s is larger, the number of boundary vertices usually also increases accordingly. In Fig. 5c, with an increasing number of vertices $n \in [0.5K, 1K, 2K, 3K]$, the communication cost of *LocalSS* increases roughly linearly in n, supporting our theoretical result. More complete discussions can be found in the Appendix.

ϵ	Comm. Cost (cost/m*100%)	Approx. Quality
0.2	63 (0.06)	0.05 (0.004)
0.3	31 (0.01)	0.1 (0.02)
0.4	18 (0.04)	0.15 (0.02)
0.5	11 (0.02)	0.21 (0.02)
0.6	8 (0.01)	0.29 (0.02)
0.7	6 (0.01)	0.31 (0.03)
0.8	5 (0.01)	0.4 (0.04)
0.9	4 (0.01)	0.42 (0.04)

Table 2. Performance of *LocalSC* on *Beach* dataset in the baseline setting. Numbers in the parentheses are standard deviations.

ϵ	Comm. Cost (cost/m*100%)	Approx. Quality
0.2	43 (0.07)	0.03 (0.001)
0.3	33 (0.07)	0.06 (0.005)
0.4	26 (0.07)	0.09 (0.01)
0.5	20 (0.05)	0.14 (0.01)
0.6	15 (0.05)	0.27 (0.1)
0.7	11 (0.04)	0.35 (0.09)
0.8	9 (0.03)	0.39 (0.04)
0.9	7 (0.01)	0.51 (0.09)

Table 3. Performance of *LocalSC* on *Sculpture-11M* dataset in the baseline setting.

5. Conclusion and Future Work

We propose both distributed constructions of approximate Schur complements and pair-wise spanners under the well-established message-passing communication model. Rigorous theoretical analysis is performed for all the algorithms developed in this paper. We also conduct experiments to evaluate the communication efficiency of the distributed Schur complement algorithms. In the future, we will investigate how to extend our algorithms to other communication models such as the blackboard model. It is also interesting to study distributed constructions of other graph sparsification techniques, *e.g.*, spectral and cut sparsifiers.

Acknowledgements

We thank anonymous reviewers for their constructive comments. Chun Jiang Zhu was supported by UNC Greensboro start-up funds. This work was partially supported by NSF grants CCF-1514357 and IIS-1718738, and NIH grants R01DA051922 and R01MH119678 to J. Bi.

References

Abboud, A. and Bodwin, G. Error amplification for pairwise spanner lower bounds. In *Proceedings of SODA Conference*, pp. 841–856, 2016.

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

Ahmed, R., Bodwin, G., Sahneh, F. D., Hamm, K., Jebelli, M. J. L., Kobourov, S., and Spence, R. Graph spanners: a tutorial review. In *arXiv:1909.03152*, 2020.

Balcan, M.-F., Ehrlich, S., and Liang, Y. Distributed k-means and k-median clustering on general communication topologies. In *Proceedings of NIPS Conference*, pp. 1995–2003, 2013.

Balcan, M.-F., Kanchanapally, V., Liang, Y., and Woodruff, D. P. Improved distributed principal component analysis. In *Proceedings of NIPS Conference*, pp. 3113–3121, 2014.

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

Baswana, S., Kavitha, T., Mehlhorn, K., and Pettie, S. Additive spanners and (α, β)-spanners. *ACM Transactions on Algorithms*, 7(1), 2010.

Bodwin, G. Linear size distance preservers. In *Proceedings of SODA Conference*, pp. 600–615, 2017.

Bodwin, G. and Williams, V. V. Better distance preservers and additive spanners. In *Proceedings of SODA Conference*, pp. 855–872, 2016.

Bollobas, B., Coppersmith, D., and Elkin, M. Sparse distance preservers and additive spanners. *SIAM Journal on Discrete Mathematics*, 19(4):1029–1055, 2005.

Borgwardt, K. M. and Kriegel, H.-P. Shortest-path kernels on graphs. In *Proceedings of ICDM Conference*, pp. 74–81, 2005.

Bravo Hermsdorff, G. and Gunderson, L. M. A unifying framework for spectrum-preserving graph sparsification and coarsening. In *Proceedings of NeurIPS Conference*, pp. 7736–7747, 2019.

Calandriello, D., Koutis, I., Lazaric, A., and Valko, M. Improved large-scale graph learning through ridge spectral sparsification. In *Proceedings of ICML Conference*, pp. 688–697, 2018.

Censor-Hillel, K., Kavitha, T., Paz, A., and Yehudayoff, A. Distributed construction of purely additive spanners. *Distributed Computing*, 31(3):223–240, 2018.

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

Chen, Y., Wu, L., and Zaki, M. J. Iterative deep graph learning for graph neural networks: better and robust node embeddings. In *Proceedings of NeurIPS Conference*, 2020.

Coppersmith, D. and Elkin, M. Sparse sourcewise and pairwise preservers. *SIAM Journal on Discrete Mathematics*, 20(2):463–501, 2006.

Costa, F. and Grave, K. D. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of ICML Conference*, pp. 255–262, 2010.

Cygan, M., Grandoni, F., and Kavitha, T. On pairwise spanners. In *Proceedings of STACS Conference*, pp. 209–220, 2013.

Dorfler, F. and Bullo, F. Kron reduction of graphs with applications to electrical networks. *IEEE Transactions on Circuits and System*, 60(1):150–163, 2013.

Durfee, D., Gao, Y., Goranci, G., and Peng, R. Fully dynamic spectral vertex sparsifiers and applications. In *Proceedings of STOC Conference*, pp. 914–925, 2019.

Fernandez, M. V., Woodruff, D. P., and Yasuda, T. Graph spanners in the message-passing model. In *Proceedings of ITCS Conference*, 2020.

Forster, S., Goranci, G., Liu, Y., Peng, R., Sun, X., and Ye, M. Minor Sparsifiers and the Distributed Laplacian Paradigm. *https://arxiv.org/abs/2012.15675*, 2021.

Hermansson, L., Johansson, F. D., and Watanabe, O. Generalized shortest path kernel on graphs. In *Proceedings of International Conference on Discovery Science*, pp. 78–85, 2015.

Kavitha, T. New pairwise spanners. *Theory of Computing Systems*, 61:1011–1036, 2017.

Kavitha, T. and Varma, N. Small stretch pairwise spanners and approximate D-preservers. *SIAM Journal on Discrete Mathematics*, 29(4):2239–2254, 2015.

Koutis, I., Miller, G. L., and Tolliver, D. Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. *Computer Vision and Image Understanding*, 115(12):1638–1646, 2011.

Kyng, R., Lee, Y., Peng, R., Sachdeva, S., and Spielman, D. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of STOC Conference*, pp. 842–850, 2016.

Li, H. and Schild, A. Spectral subspace sparsification. In *Proceedings of FOCS Conference*, pp. 385–396, 2018.

Loukas, A. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20:1–42, 2019.

Loukas, A. and Vandergheynst, P. Spectrally approximating large graphs with smaller graphs. In *Proceedings of ICML Conference*, pp. 3243–3252, 2018.

Parter, M. Bypassing Erdos' girth conjecture: Hybrid spanners and sourcewise spanners. In *Proceedings of ICALP Conference*, pp. 608–619, 2014.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *Proceedings of ICLR Conference*, 2020.

Saad, Y. and Sosonkina, M. Distributed Schur complement techniques for general sparse linear systems. *SIAM Journal of Scientific Computing*, 21(4):1337–1356, 1999.

Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011.

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

Sun, H. and Zanetti, L. Distributed graph clustering and sparsification. *ACM Transactions on Parallel Computing*, 6(3):17, 2019.

Tetali, P. Random walks and effective resistance of networks. *Journal of Theoretical Probability*, 1:101–109, 1991.

von Luxburg, U., Radl, A., and Hein, M. Hitting and commute times in large random neighborhood graphs. *Journal of Machine Learning Research*, 15(1):1751–1798, 2014.

Wagner, T., Guha, S., Kasiviswanathan, S. P., and Mishra, N. Semi-supervised learning on data streams via temporal label propagation. In *Proceedings of ICML Conference*, pp. 5082–5091, 2018.

Woodruff, D. P. and Zhang, Q. When distributed computation is communication expensive. *Distributed Computing*, 30(5):309–323, 2017.

Yang, Y., Wang, X., Song, M., Yuan, J., and Tao, D. SPAGAN: shortest path graph attention network. In *Proceedings of IJCAI Conference*, pp. 4099–4015, 2019.

You, J., Ying, R., and Leskovec, J. Position-aware graph neural networks. In *Proceedings of ICML Conference*, pp. 7134–7143, 2019.

Zhang, F. The Schur complement and its applications. *Numerical Methods and Algorithms*, 2005.

Zheng, C., Zong, B., Cheng, W., Song, D., Ni, J., Yu, W., Chen, and Wang, W. Robust graph representation learning via neural sparsification. In *Proceedings of ICML Conference*, pp. 11458–11468, 2020.

Zhu, C., Storandt, S., Lam, K.-Y., Han, S., and Bi, J. Improved dynamic graph learning through fault-tolerant sparsification. In *Proceedings of ICML Conference*, pp. 7624–7633, 2019a.

Zhu, C., Zhu, T., Lam, K.-Y., Han, S., and Bi, J. Communication-optimal distributed dynamic graph clustering. In *Proceedings of AAAI Conference*, pp. 5957–5964, 2019b.