Improved Dynamic Graph Learning through Fault-Tolerant Sparsification

Chun Jiang Zhu [1] **Sabine Storandt** [2] **Kam-Yiu Lam** [3] **Song Han** [1] **Jinbo Bi** [1]

Abstract

Graph sparsification has been used to improve the computational cost of learning over graphs, *e.g.*, Laplacian-regularized estimation, graph semi-supervised learning (*SSL*) and spectral clustering (*SC*). However, when graphs vary over time, repeated sparsification requires polynomial order computational cost per update. We propose a new type of graph sparsification namely fault-tolerant (*FT*) sparsification to significantly reduce the cost to only a constant. Then the computational cost of subsequent graph learning tasks can be significantly improved with limited loss in their accuracy. In particular, we give theoretical analysis to upper bound the loss in the accuracy of the subsequent Laplacian-regularized estimation, graph *SSL* and *SC*, due to the FT sparsification. In addition, FT spectral sparsification can be generalized to FT cut sparsification, for cut-based graph learning. Extensive experiments have confirmed the computational efficiencies and accuracies of the proposed methods for learning on dynamic graphs.

1. Introduction

For statistical estimation problems over graphs, an effective regularization term can be based on the underlying graph structures, specifically, the graph Laplacian (Calandriello et al., 2018; Sadhanala et al., 2016). Consider a graph $G(V, E, W)$ with n vertices and m edges, its Laplacian matrix L_G is $D_G - A_G$, where A_G and D_G are the adjacency and degree matrices of G, respectively. [1] Suppose $y = (y_1, \cdots, y_n)$ are observations over vertices in G. They are independently drawed from a model parameter $\beta^* = (\beta_1^*, \cdots, \beta_n^*)$ with Gaussian noises, *i.e.*, for every $i \in [1, n]$, $E(y_i) = \beta_i^*$. Then *Laplacian-regularized estimation* is to solve

$$\min_{\beta \in R^n} ||y - \beta||^2 + \lambda \beta L_G \beta^T, \quad (1)$$

where λ is a regularization parameter. Note that the regularization term $\beta L_G \beta^T = \sum_{(i,j)\in E} W(i,j)(\beta_i - \beta_j)$. So, the intuition is to find a vector $\hat{\beta}$ such that it will have components that vary smoothly over adjacent vertices in G, while λ controls the degree of smoothness. [2] Other spectral methods (those with a loss function based on the Laplacian) include graph Semi-Supervised Learning (*SSL*) (Zhu et al., 2003), Logistic Smoothing (Sadhanala et al., 2016), graph-regularized least squares (Belkin et al., 2005) and spectral clustering (*SC*) (Ng et al., 2001). There also exists another type of graph-based learning based on *graph cuts* and using cut-based algorithms directly, instead of spectral methods, for enhancement, *e.g.*, *Min-Cut* for *SSL* (Blum & Chawla, 2001), *Max-Cut* for *SSL* (Wang et al., 2013), *Sparsest-Cut* for hierarchical learning (Moses & Vaggos, 2017) and *Max-Flow* for *SSL* (Rustamov & Klosowski, 2018).

However, all these methods suffer from the scalability problem. *E.g.*, Equation (1) has a closed-form solution $\hat{\beta} = (I + \lambda L_G)^{-1} y$. As the matrix $I + \lambda L_G$ is Strongly Diagonally Dominant (*SDD*), one can use an optimal solver for *SDD* matrices, which requires $\tilde{O}(m)$ time (the notation $\tilde{O}$ hides a polylogarithmic factor) (Spielman & Teng, 2004). Nonetheless, even for mildly large, dense graphs with millions of edges, *e.g.*, social networks, protein interaction networks and communication networks, the above methods are not feasible. Recently, (Calandriello et al., 2018; Sadhanala et al., 2016) proposed to preprocess G into a sparse yet spectrally similar graph, called a *sparsifier* H of size $\tilde{O}(n)$ for subsequent graph learning. By using H instead of G in the Laplacian-regularized term as below,

$$\min_{\beta \in R^n} ||y - \beta||^2 + \lambda' \beta L_H \beta^T, \quad (2)$$

one can solve $\tilde{\beta} = (I + \lambda' L_H)^{-1}$ in only $\tilde{O}(n)$ time, and it has been shown in the papers that the accuracy of the estimation is not affected significantly by using H.

[1]University of Connecticut [2]University of Konstanz [3]City University of Hong Kong. Correspondence to: Jinbo Bi <jinbo.bi@uconn.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

[1]A_G and D_G are defined as $(A_G)_{u,v} = W(u, v)$ if $(u, v) \in E$ and $(A_G)_{u,v} = 0$ otherwise, $(D_G)_{u,v} = \sum_{w\in V} W(u, w)$ if $u = v$, and $(D_G)_{u,v} = 0$ otherwise, respectively.

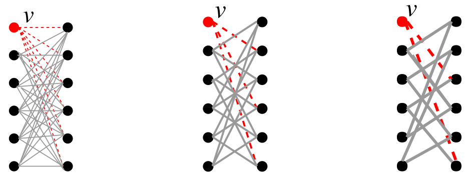
[2]Here we implicitly assume that the underlying signal β^* is smooth over G.

Nonetheless, in practice graphs vary over time and can have updates in terms of vertex/edge insertions/deletions. Reducing the computational cost in learning on dynamically changing graphs is more challenging. *E.g.*, solving Equation (1) at every time point is not feasible obviously, as recall that the computational cost for a static graph at each time point is already not practical. Then similar to the static case, one may turn to incrementally maintain, at each time point t, a sparse graph H_t spectrally similar to the current graph G_t, and then efficiently solve Equation (2) with L_{H_t}. To achieve incremental maintenance of H_t over time, there exists dynamic or streaming spectral sparsification algorithms, *e.g.*, (Abraham et al., 2016; Kelner & Levin, 2013; Kapralov et al., 2014). However, these methods have several major problems: (1) the per update computational cost is polylogarithmic w.r.t. n, which could still be high especially when the number of updates is large; (2) they are quite complicated and purely theoretical with no experimental studies, and thus may not be practical. Designing a simple and practical solution for learning on dynamic graphs is of great interest.

In this work, we propose a new type of graph sparsification to resolve the problems above for learning on dynamic graphs: the proposed fault-tolerant subgraphs have *constant* per update computational cost, work for both vertex and edge insertion/deletions, and will be shown to have both theoretical and experimental guarantees for accuracy of subsequent Laplacian-regularized estimation and graph *SSL*.

For notational convenience, it is assumed that we start from a graph $G = G_0$ such that for every time point $t \geq 1$, the vertex set V_t and edge set E_t of G_t at the time point t are a subset of V_0 and E_0, respectively. When otherwise, we can add an additional graph G' as the start graph, by including into G' all the vertices and edges in $G_0, G_1, \cdots$, as long as they are pre-defined or known in advance. This is not uncommon in applications. *E.g.*, in a topological network, the vertices and edges, which represent peers and their communication infrastructures resp., are pre-defined so that later the actual communications between peers take place and form communication networks over time. Moreover, the original assumption has its root in fault-tolerant subgraph studies (Chechik et al., 2010; Dinitz & Krauthgamer, 2011; Bodwin et al., 2018) and thus has many applications. *E.g.*, in a social network, users may prefer to temporarily unfriend some friends (for a post) or even disable their own accounts and then recover later. That is, these edges and vertices are marked as "faulty" temporarily. In a communication network, there are also frequent failed and later recovered computing nodes or communication links.

Specifically, for the spectral methods, *e.g.* Laplacian-regularized estimation and graph *SSL* (Zhu et al., 2003), we propose *fault-tolerant spectral sparsifiers*, while for cut-based methods, *e.g.* *Min-Cut* for *SSL* (Blum & Chawla, 2001) and *Max-Cut* for *SSL* (Wang et al., 2013), we propose *fault-tolerant cut sparsifiers*. Given a graph G, a (possibly re-weighted) subgraph H of G is called an f-vertex-fault-tolerant (*FT*) spectral (cut) sparsifier, if for all possible faulty vertex sets F of size $|F| \leq f$, the subgraph $H - F$ is guaranteed to be a *spectral (cut) sparsifer* of $G - F$, *i.e.*, the spectral Laplacian quadratic forms (all graph cuts) of $H - F$ and $G - F$ are the same within a factor of $1 \pm \epsilon$. The definition can be naturally extended to the edge-*FT* setting. We use Fig. 1 to illustrate the definitions, where two *FT* cut sparsifiers H_1 and H_2 of G with different ϵ are shown. We will discuss how the proposed *FT* sparsifiers can be used to reduce the computational cost.



(a) G (b) H_1 (c) H_2

Figure 1: 1-*FT* cut sparsifiers of G: H_1 and H_2. (a) G with 36 edges and edge weight 1. (b) H_1 with 18 edges and edge weight 2. (c) H_2 with 12 edges and edge weight 3. Without loss of generality consider that v is faulty. The *Min-Cut* of $G - \{v\}$ is 5, while the *Min-Cut* of $H_1 - \{v\}$ and $H_2 - \{v\}$ are 4 and 3, respectively. Then H_1 and H_2 are 1-*FT* (1 ± 0.2)-cut sparsifier and (1 ± 0.4)-cut sparsifier of G, respectively.

Our Contribution. The high computational cost of graph-based learning, *e.g.*, Laplacian-regularized estimation and graph *SSL*, has been alleviated by graph sparsification. However, when graphs are changing over time, repeated sparsification requires polynomial order computational cost per update. We design computation-efficient methods to significantly reduce the cost to only a constant. To improve the computational cost of spectral methods, we propose efficient algorithms for constructing *FT* spectral sparsifiers, including a core algorithm, a parallel algorithm and a distributed algorithm. Furthermore, for cut-based methods, we propose a new algorithm for constructing *FT* cut sparsifiers. Then the computational cost of subsequent graph learning tasks can be significantly improved with limited loss in their accuracy. In particular, we give theoretical analysis to upper bound the loss in the accuracy of the subsequent Laplacian-regularized estimation, graph *SSL* and *SC*, due to the FT sparsification. Finally, we have performed extensive experiments to confirm the computational efficiencies and accuracies of the proposed methods for learning on dynamic graphs.

2. Background

Graph SSL. In this problem, we are only given observations y_S of the signal β^* for a subset of vertices $S \subset V$ in G and the task is to predict the signals for the rest of vertices

$V - S$. A typical formulation is the *harmonic function solution (HFS)* (Zhu et al., 2003), which solves the following optimization problem.

$$\begin{aligned}\hat{\beta}_{HFS} &= \underset{\beta \in R^n}{\arg\min} \frac{1}{|S|}(\beta - y)^T l_S(\beta - y) + \lambda \beta L_G \beta^T \\ &= (\lambda |S| L_G + I_S)^+ y I_S\end{aligned} \tag{3}$$

I_S is the identity matrix with zeros at vertices not in S.

Graph Sparsification. A re-weighted subgraph H of G is called a *cut sparsifier* of G if for every vertex set S, the weight of the cut between S and $V - S$ in H is at most $1 \pm \epsilon$ times of the weight of the cut in G (Benczur & Karger, 1996). (Fung et al., 2011) proposed to construct cut sparsifiers by sampling edges according to *edge connectivities*. In a seminal work, Spielman and Teng (Spielman & Teng, 2011) generalized cut sparsifiers to spectral sparsifiers. (Spielman & Srivastava, 2011) proposed to construct spectral sparsifiers by sampling edges with probability proportional to their *effective resistances*. G can be considered as an electrical resistive network, and the effective resistance between any two vertices in G is defined as the potential difference that has to be applied between them in order to drive one unit of current through the network G.

A subgraph H of $G(V, E)$ is called a k-*spanner* of G if for every $u, v \in V$, the distance between u and v in H is at most k times of their distance in G (Peleg & Schaffer, 1989). (Kapralov & Panigrahy, 2012) showed that a spectral sparsifier can be obtained by repeated constructions of spanners. (Koutis & Xu, 2016) proposed an algorithm framework based on spanners and sampling. (Lee & Sun, 2017) proposed the state-of-the-art algorithm for constructing $O(n)$ sized spectral sparsifiers in $\tilde{O}(m)$ time. Graph sparsification has been widely used in linear algebra and algorithm design (Spielman & Teng, 2011), particularly in improving the computational cost of learning over graphs (Calandriello et al., 2018; Sadhanala et al., 2016).

Dynamic Graph Sparsification. There has been theoretical algorithms focusing on designing dynamic graph sparsificatiion. (Kelner & Levin, 2013; Cohen et al., 2016) designed dynamic sparsifications for insertion-only streams, which can support edge insertions. But their algorithms cannot process edge deletions. (Ahn et al., 2012) proposed a streaming algorithm for constructing cut sparsifiers. Later, (Kapralov et al., 2014) proposed a single pass streaming algorithm for constructing spectral sparsifiers based on the iterative sparsification (Li et al., 2013). (Abraham et al., 2016) proposed a fully dynamic sparsification by thoroughly modifying (Koutis & Xu, 2016) to guaranteeing that updates do not propagate in recursive spanner constructions. However, all these algorithms require $\Omega(polylog(n))$ computational cost per update. We reduce the cost to a constant under the assumption that the graph being maintained at every time point differs from the original graph by a bounded amount.

Fault-Tolerant Subgraphs. This topic has been extensively studied in the theorem community. For short, we will use *VFT* and *EFT* to refer to Vertex-Fault-Tolerant and Edge-Fault-Tolerant, respectively. f-*VFT* (f-*EFT*) k-spanners are a subgraph such that for all vertex (edge) faults of size at most f, the remaining part of the subgraph is always an k-spanner of the remaining part of the original graphs. They were firstly studied by (Levcopoulos et al., 1998) in the context of geometric graphs. (Chechik et al., 2010) generalized them to general graphs and proposed algorithms for constructing both *VFT* and *EFT* spanners. Recently, (Bodwin et al., 2018) proved that for a fixed k, every n-vertex graph has an f-*VFT* (f-*EFT*) spanner of stretch $2k - 1$ and size $O_k(f^{1-1/k} \cdot n^{1+1/k})$ (the O_k notation suppresses a $2^{O(k)}$ factor). Due to limit of space, more *FT* studies on graphs are referred to the Appendix. Unfortunately, the important topics of *FT* spectral sparsiers and cut sparsifiers have been greatly ignored.

Notation and Definition. A weighted undirected graph $G(V, E, W)$ consists of a vertex set V, an edge set E and a weight function W which assigns a weight $W(e)$ to each edge $e \in E$. W can be omitted from the presentation if it is clear from the context. A simple path, or a path in short, P between u and v in G is a sequence of edges $(u = v_1, v_2), \cdots, (v_l, v_{l+1} = v)$. Its distance is equal to $\sum_{i=1}^{l} W(v_i, v_{i+1})$. Each edge e in G has *resistance* $R(e) = 1/W(e)$, and the effective resistance between any two vertices u and v in G is denoted as $R_G(u, v)$. Suppose that P is a path connecting the two endpoints of an edge e, the *stretch* of e over P is equal to $\alpha_P(e) = W(e) \sum_{e' \in P}(1/W(e'))$.

For an arbitrary vertex set V_F, we use $G - V_F$ to denote the remaining graph of G after removing vertices in V_F and their edges. For an arbitrary edge set E_F, $G - E_F$ denotes the remaining graph of G after removing edges in E_F. For an arbitrary graph H, $G - H$ denotes the remaining graph of G after removing edges in H. The notation $L_A \preceq L_B$ means that for every vector $x \in R^n$, $x^T L_A x \leq x^T L_B x$, while $L_A \preceq^{\{0,1\}} L_B$ means that for every vector $x \in \{0, 1\}^n$, $x^T L_A x \leq x^T L_B x$. For an event Z, we use $Pr[Z]$ to denote the probability of the event Z happens. We say an event Z happens *with high probability (w.h.p.)* if, $Pr[Z] \geq 1 - 1/n^c$ for some constant $c > 1$. The Laplacian matrix L_G^e of an edge e in G is the Laplacian matrix of the subgraph of G containing only the edge e. It is zero elsewhere except a 2×2 submatrix.

3. FT Spectral Sparsifiers

Before we provide algorithms for constructing *FT* spectral sparsifiers, we formally define them as follows. We first describe the core algorithm in Section 3.1, and then present the parallel and distributed algorithms in Section 3.2.

Definition 1. *For a graph $G(V, E)$, a positive integer f and parameter $\epsilon \in (0, 1)$, a re-weighted subgraph $H(V, E' \subseteq E)$ is an f-**VFT** (f-**EFT**) $(1 \pm \epsilon)$-**spectral sparsifier**, if for all vertex (edge) sets $F \subseteq V$ ($F \subseteq E$) of size $|F| \leq f$, $(1-\epsilon)L_{G-F} \preceq L_{H-F} \preceq (1+\epsilon)L_{G-F}$ holds.*

3.1. Proposed Algorithms

In this section, we propose algorithms for constructing *FT* spectral sparsifier of bounded size. The result are summarized in the following theorem.

Theorem 1. *For an n-vertex m-edge graph G, a positive integer f, a parameter $\epsilon \in (0, 1)$ and $\rho > 1$, an f-VFT (f-EFT) $(1 \pm \epsilon)$-spectral sparsifier for G of expected size $O(fn \log \rho + n \log^2 n \log^3 \rho/\epsilon^2 + m/\rho)$ w.h.p. can be constructed.*

We will use *FT* spanners formally defined as follows.

Definition 2. *For positive integers f and α, a subgraph H of a graph G is an f-**VFT** (f-**EFT**) α-**spanner** of G, if for all vertex (edge) sets F of size $|F| \leq f$, we have that for every $u, v \in V$ the distance between u and v in $H - F$ is at most α times of their distance in $G - F$.*

Our algorithm is inspired by the sparsification algorithm by (Koutis & Xu, 2016). However, we need to handle the cases when there are some vertices or edges to become faulty. The idea of the algorithm is to first construct an $(f + t)$-*VFT*/*EFT* spanner for the input graph G by any *VFT*/*EFT* graph spanners algorithms to determine a set of edges with small effective resistances in G. The $(f + t)$-*VFT*/*EFT* spanner guarantees that even in the presence of at most f faults, each non-spanner edge (edge not in the spanner) has t edge-disjoint paths between its endpoints in the spanner (and thus in the input graph G), serving as a certificate for an upper bounded effective resistance of the edge. It then uniformly samples each non-spanner edge with a fixed constant probability, and scales the edge weight of each sampled edge proportionally to preserve the edge's expectation. This sampling step is inherently *VFT*/*EFT* because the sampling of each non-spanner edge is independent. By the matrix concentration bounds, we can prove that the spanner together with the sampled non-spanner edges are a moderately sparse *VFT*/*EFT* spectral sparsifier, in which the number of edges has been reduced by a constant factor. The desirable *VFT*/*EFT* spectral sparsifier can be obtained by repeating the process until we get a sufficient sparsity, which happens after a logarithmic number of iterations.

This algorithm is simple, yet works for both *VFT* and *EFT* settings. It is interesting to discover that *FT* spectral sparsifiers can be constructed via *FT* graph spanners, and thus extending the connection between spectral sparsifiers and spanners from a standard setting (Kapralov & Panigrahy, 2012) to the new *FT* setting.

Algorithm 1 *Light-FTSS*

```
Require: $G(V, E), f > 0, \epsilon \in (0, 1)$
Ensure: $H$
  Construct an $(f + 24 \log^2 n/\epsilon^2)$-FT $(\log n)$-spanner $J$ for $G$;
  $H \leftarrow J$;
  for each edge $e \in G - J$ do
    With probability 0.25, add $e$ to $H$ with a new weight $4W(e)$;
  end for
```

Algorithm 2 *FTSS*

```
Require: $G(V, E), f > 0, \epsilon \in (0, 1), \rho > 1$
Ensure: $H$
  $G_0 \leftarrow G$;
  for $i \in [1, \lceil \log \rho \rceil]$ do
    $G_i \leftarrow$ Light-FTSS$(G_{i-1}, f, \epsilon/\lceil \log \rho \rceil)$;
  end for
  $H \leftarrow G_{\lceil \log \rho \rceil}$;
```

Specifically, our algorithm is summarized in Algorithms 1 and 2. For a graph G, a positive integer f and an $\epsilon \in (0, 1)$, Algorithm 1 (*Light-FTSS*) first constructs an $(f + 24 \log^2 n/\epsilon^2)$-*FT* $(\log n)$-spanner J for G and adds edges in J to an edge set H. It then samples each edge $e \notin J$ with a fixed probability 0.25 and adds e to H with weight $4W(e)$ if it is sampled. The probability 0.25 is chosen in order to make the expected size bound hold w.h.p.. Finally, it returns H as the moderately sparse *FT* spectral sparsifier. Given a graph G, a positive integer f, an $\epsilon \in (0, 1)$ and $\rho > 1$, for every $i \in [1, \lceil \log \rho \rceil]$ Algorithm 2 calls *Light-FTSS* with parameters G_{i-1}, f and $\epsilon/\lceil \log \rho \rceil$, where $G_0 = G$ and G_j is the output of *Light-FTSS* in the iteration when $i = j$. After the iterations terminate, $G_{\lceil \log \rho \rceil}$ is returned as the final *FT* spectral sparsifier.

We first prove the following key lemma.

Lemma 1. *Let G be an n-vertex graph and J be an f-VFT (f-EFT) $(\log n)$-spanner of G. For every edge $e \in G - J$, and every vertex (edge) set F of size $|F| \leq \hat{f}$ such that $e \in G - F$, we have that*

$$W(e) \cdot R_{G-F}(e) \leq \log n/(f - \hat{f}),$$

which implies that

$$W(e) \cdot L^e_{G-F} \preceq \log n/(f - \hat{f}) \cdot L_{G-F}.$$

Proof. **The *VFT* setting.** We first show that for every edge $e \in G - J$, there are at least f vertex-disjoint paths $P_1, \cdots, P_g$ between the two endpoints of e in J, such that for every $i \in [1, g]$, $\alpha_{P_i}(e) \leq \log n$. Suppose for contradiction that for an edge $e \in G - J$, there are $g < f$ such paths in J. Then by correctly setting a faulty vertex set F, we can invalidate all these g paths in the graph $J - F$.

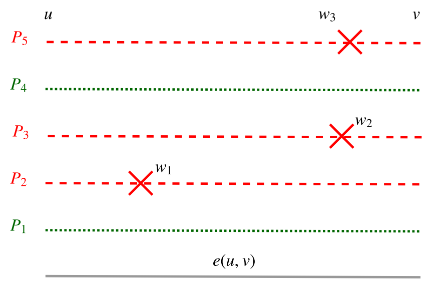


Figure 2: A faulty vertex set $F = \{w_1, \cdots, w_{\hat{f}}\}$ of size $\hat{f}$ can invalidate at most $\hat{f}$ paths out of f vertex-disjoint paths between endpoints u and v of an edge $e(u, v)$. Here $f = 5$ and $\hat{f} = 3$.

Specifically, we select an arbitrary vertex, excluding the two endpoints of e, from each of $P_1, \cdots, P_g$ to formulate F. By construction, $|F| = g < f$. For the edge $e \in G - F$, $\alpha_{J-F}(e) = \infty$ because there is no path between its two endpoints in $J - F$. This contradicts with the fact that J is an f-*VFT* $(\log n)$-spanner of G.

We then show that for every edge $e \in G - J$, and every fault set F of size $\hat{f}$ such that $e \in G - F$, there are at least $f - \hat{f}$ vertex-disjoint paths $P_1, \cdots, P_h$ $(h \geq f - \hat{f})$ between the two endpoints of e in $J - F$, where for every $i \in [1, h]$, $\alpha_{P_i}(e) \leq \log n$. This is because a fault set F of size $\hat{f}$ can invalidate at most $\hat{f}$ paths as in Fig. 2. By definition, for every $i \in [1, h]$, we have that

$$\alpha_{P_i}(e) = W(e) \sum_{e \in P_i} (1/W(e)) \leq \log n. \tag{4}$$

According to the formula for resistors connected in series, for every path P_i with $i \in [1, h]$, the effective resistance between the two endpoints of e in P_i is equal to

$$R_{P_i}(e) = \sum_{e \in P_i} R(e) = \sum_{e \in P_i} (1/W(e)) \tag{5}$$

Combining Equations (4) and (5), we have that for every $i \in [1, h]$, $R_{P_i}(e) \leq \log n / W(e)$. According to the formula for resistors connected in parallel, for a set of edge-disjoint (vertex-disjointness subsumes edge-jointness) paths $\{P_1, \cdots, P_h\}$ between e's two endpoints, and let P be the union of these paths $P = \cup_{i=1}^{h} P_i$, the effective resistance between e's two endpoints in P is equal to $R_P(e) = (\sum_{i=1}^{h} (R_{P_i}(e))^{-1})^{-1} \leq \log n / h \cdot W(e) \leq \log n / (f - \hat{f}) \cdot W(e)$. According to the Rayleigh's monotonicity law (Doyle & Snell, 2000), for any subgraph H of G and any edge $e \in G$, $R_G(e) \leq R_H(e)$ holds. Therefore,

$$R_{G-F}(e) \leq R_P(e) \leq \log n / (f - \hat{f}) \cdot W(e). \tag{6}$$

By (Spielman & Srivastava, 2011), we have $L_G^e \preceq R_G(e) L_G$. Then applying it to the graph $G - F$, we have that

$$L_{G-F}^e \preceq R_{G-F}(e) L_{G-F}. \tag{7}$$

By combining Equation (7) with Equation (6), we have that

$$W(e) \cdot L_{G-F}^e \preceq \log n / (f - \hat{f}) \cdot L_{G-F}.$$

The *EFT* setting. We show that for every edge $e \in G - J$, there are at least f edge-disjoint paths $P_1, \cdots, P_g$ between the endpoints of e in J such that for every $i \in [1, g]$, $\alpha_{P_i}(e) \leq \log n$. Suppose for contradiction that for an edge $e \in G - J$, there are $g < f$ such paths. Then, we can invalidate all g paths in $J - F$ by selecting an arbitrary edge from each of $P_1, \cdots, P_g$ and adding them into F. We have that the size of F is $g < f$. For the edge $e \in G - F$, $\alpha_{J-F}(e) = \infty$ because its endpoints are not connected in $J - F$, and thus $\alpha_{J-F}(e) > \log n$. This contradicts with the fact that J is an f-*EFT* $(\log n)$-spanner of G. The rest of the proof for the *EFT* setting is similar to that for the *VFT* setting, and thus omitted here. □

The proposed algorithm is an algorithm framework where any *VFT* (*EFT*) graph spanner algorithms can be plugged in, in Line 1 of Algorithm 1. We employ the state-of-the-art algorithms for constructing optimal *VFT* and suboptimal *EFT* spanners (for a fixed stretch) by (Bodwin et al., 2018), in order to derive the size bound in Theorem 1. We can use any other *VFT* (*EFT*) spanner algorithms here, but the bounds obtained can be slightly worse. The algorithm is a natural generalization of the greedy spanner algorithm by (Althofer et al., 1993) to the *FT* setting. Given a graph $G(V, E)$, it examines the edges of E in non-decreasing order of weight (ties are broken arbitrarily), and adds an edge e into the current spanner H if and only if there exists a fault set F such that $\alpha_{H-F}(e) > \alpha = \log n$, where the stretch factor is a fixed integer $\alpha = \log n$. We will use the following theorem.

Theorem 2. *(Bodwin et al., 2018) Let G be an n-vertex graph without negative weight cycle, and α be a fixed integer $\log n$. For any positive (possibly non-constant) integer f, G has an f-VFT (f-EFT) α-spanner of size $O(fn)$.*

For the *EFT* setting, we can also use an alternative *EFT* $(\log n)$-spanner algorithm (Chechik et al., 2010) combined with the greedy graph spanner algorithm by (Althofer et al., 1993) to get *EFT* spectral sparsifiers with the same size bound as in Theorem 1. For a graph G, the algorithm (Chechik et al., 2010) constructs an f-*EFT* $(\log n)$-spanner (the same as the f-bundle spanner) as the union of a sequence of graphs $H = \cup_{i=1}^{f} H_i$, where for every $i \in [1, f]$, H_i is a $(\log n)$-spanner of the graph $G - \cup_{j=1}^{i-1} H_j$. However, it cannot be extended to the *VFT* setting.

We will also use the following variant (Harvey, 2012) of a matrix concentration bound by Tropp (Tropp, 2012).

Theorem 3. *(Harvey, 2012) Let $Y_1, \cdots, Y_k$ be independent positive semi-definite matrices of size $n \times n$. Let $Y = \sum_{i=1}^{k} Y_i$ and $Z = E[Y]$. Suppose for every $i \in [1, k]$,*

$Y_i \preceq SZ$, where S is a scalar. Then for all $\epsilon \in [0,1]$, $Pr[\sum_{i=1}^{k} Y_i \preceq (1-\epsilon)Z] \leq n \cdot exp(-\epsilon^2/2S)$, and $Pr[\sum_{i=1}^{k} Y_i \succeq (1+\epsilon)Z] \leq n \cdot exp(-\epsilon^2/3S)$.

We are now ready to prove the following theorem summarizing properties of Algorithm 1 (*Light-FTSS*).

Theorem 4. *Given an n-vertex m-edge graph G, a positive integer f and $\epsilon \in (0,1)$, Light-FTSS constructs an f-VFT (f-EFT) $(1 \pm \epsilon)$-spectral sparsifier for G of expected size $O(fn + n\log^2 n/\epsilon^2 + m/2)$, with probability at least $1 - 1/n^2$.*

Proof. We only consider the *VFT* setting as the proof for the *EFT* setting follows similar principle. We first prove that the output H by *Light-FTSS* is an f-*VFT* $(1 \pm \epsilon)$-spectral sparsifier. Let $F \subseteq V$ be any vertex fault set of size at most f. For every edge $e \in G - J$, let X_e be the random variable defined as

$$X_e = \begin{cases} 4W(e)L^e_{G-F}, & \text{with probability } 0.25 \\ 0, & \text{otherwise} \end{cases}$$

For every $i \in [1, (\lfloor \epsilon^2/(6\log n) \rfloor)^{-1}]$, let $J_i = \lfloor \epsilon^2/(6\log n) \rfloor (J - F)$, which implies that

$$L_{J_i} = \lfloor \epsilon^2/(6\log n) \rfloor L_{J-F}.$$

We then apply Theorem 3 to the random matrix

$$\begin{aligned} Y &= \sum_{e \in G-J} X_e + \sum_{i=1}^{(\lfloor \epsilon^2/(6\log n) \rfloor)^{-1}} L_{J_i} \\ &= \sum_{e \in G-J} X_e + L_{J-F}. \end{aligned}$$

Note that

$$\begin{aligned} E(Y) &= E(\sum_{e \in G-J} X_e + L_{J-F}) = \sum_{e \in G-J} E(X_e) + L_{J-F} \\ &= \sum_{e \in G-J} L^e_{G-F} + L_{J-F} = L_{G-F}. \end{aligned}$$

By the definition of $X(e)$ and Lemma 1, for every $e \in G-J$ we have that

$$X(e) \preceq 4W(e) \cdot L^e_{G-F} \preceq \epsilon^2/(6\log n) \cdot L_{G-F}.$$

Furthermore, by definition of J_i and the fact that $L_{J-F} \preceq L_{G-F}$, we have for every $i \in [1, (\lfloor \epsilon^2/(6\log n) \rfloor)^{-1}]$,

$$L_{J_i} = \lfloor \epsilon^2/(6\log n) \rfloor \cdot L_{J-F} \preceq \epsilon^2/(6\log n) \cdot L_{G-F}.$$

Now the condition of Theorem 3 is satisfied with $S = \epsilon^2/(6\log n)$. Therefore, the inequality

$$(1-\epsilon)L_{G-F} \preceq L_{H-F} \preceq (1+\epsilon)L_{G-F} \tag{8}$$

holds with probability at least $1 - 1/2n \cdot exp(-3\log n) = 1 - 1/2n^{-2}$. This completes the proof for the spectral bound.

We then prove the size bound. The number of edges in J is $O(fn + n\log^2 n/\epsilon^2)$, according to Theorem 2. By construction, the expected number of edges in $H - J$ is $m/4$. Applying Chernoff's inequality gives that with probability at least $1 - 1/2n^{-2}$ the expected number of edges in $H - J$ is at most $m/2$, which implies the total expected number of edges is $O(fn + n\log^2 n/\epsilon^2 + m/2)$. By a union bound, the event that both the inequality (8) and the size bound hold happens with probability at least $1 - 1/n^2$. □

Given Theorem 4, the proof of Theorem 1 is referred to the Appendix, due to limit of space. Note that introducing tolerance to f vertex or edge faults costs us an extra term $fn\log\rho$ in the size of the spectral sparsifier, compared to the size of a standard combinatorial spectral sparsifier (Koutis & Xu, 2016). It is interesting to investigate how to further reduce the extra term in the size for spectral sparsifers, as (Bodwin et al., 2018) has shown that only a sub-linear term (w.r.t. f) is required when introducing the fault-tolerant property into graph spanners.

3.2. Parallel and Distributed Algorithms

In this section, we present parallel and distributed *EFT* spectral sparsification algorithms. The models of computation used by our parallel algorithm and distributed algorithm are the *CRCW PRAM* model (Reif, 1993) and the synchronized distributed model (Coulouris & Dollimore, 1988), respectively. The results are summarized in the following theorems, and their proofs have been moved to the Appendix, due to limit of space.

Theorem 5. *For an n-vertex m-edge graph G, a positive integer f, a parameter $\epsilon \in (0,1)$ and $\rho > 1$, an f-EFT $(1 \pm \epsilon)$-spectral sparsifier of size $O(fn\log n\log\rho + n\log^3 n\log^3\rho/\epsilon^2 + m/\rho)$ can be constructed in the CRCW PRAM model using $O(fm\log n\log\rho + m\log^3 n\log^3\rho/\epsilon^2)$ work in $\tilde{O}(f\log n\log\rho + \log^3 n\log^3\rho/\epsilon^2)$ time, w.h.p.*

Theorem 6. *For an n-vertex m-edge graph G, a positive integer f, a parameter $\epsilon \in (0,1)$ and $\rho > 1$, an f-EFT $(1 \pm \epsilon)$-spectral sparsifier of size $O(fn\log n\log\rho + n\log^3 n\log^3\rho/\epsilon^2 + m/\rho)$ can be constructed in the synchronized distributed model in $O(f\log n\log\rho + \log^4 n\log^3\rho/\epsilon^2)$ rounds with $O(fm\log n\log\rho + m\log^3 n\log^3\rho/\epsilon^2)$ communication complexity, using messages of $O(\log n)$ size.*

3.3. Using the FT Spectral Sparsifiers in Subsequent Graph Learning

The proposed *FT* sparsifiers have the following desirable properties by definition. At a time point $t > 0$, for each vertex v (edge e) insertion into G_{t-1}, if v (e) is in H, add v and its associated edges in H (e itself) to H_{t-1}. For each vertex v (edge e) deletion from G_{t-1}, if v (e) is in H_{t-1}, remove v and its associated edges (e) from H_{t-1}. These only incur a constant computational cost per update. More importantly, the resulting subgraph is guaranteed to be a spectral sparsifier of the graph G_t at the time point t, under the assumption that G_t differs from G_0 by a bounded amount. Then it can be used in subsequent graph learning tasks without a significant loss in their accuracy.

4. FT Cut Sparsifiers

We first formally define *FT* cut sparsifiers, and then generalize the algorithm for *FT* spectral sparsifiers to an algorithm for constructing *FT* cut sparsifiers.

Definition 3. *For a graph $G(V, E)$, a positive integer f and parameter $\epsilon \in (0, 1)$, a re-weighted subgraph $H(V, E' \subseteq E)$ is an **f-VFT (f-EFT) $(1 \pm \epsilon)$-cut sparsifier** if, for all vertex (edge) sets $F \subseteq V$ ($F \subseteq E$) of size $|F| \leq f$, $(1 - \epsilon)L_{G-F} \preceq^{\{0,1\}} L_{H-F} \preceq^{\{0,1\}} (1 + \epsilon)L_{G-F}$ holds.*

As an important building block, we define a variant of maximum spanning trees (*MST*, a spanning tree of a weighted graph having maximum weight), namely *FT α-MST*.

Definition 4. *For positive integers f and α, a subgraph H of a graph G is an **f-VFT (f-EFT) α-MST** of G, if for all vertex (edge) sets F of size $|F| \leq f$, we have that for every edge (u, v) in $G - F$, there is a path P from u to v in $H - F$ such that for every edge e' on P, $W(e) \leq \alpha W(e')$.*

Note that we relax the requirement that H is a tree. Any subgraph with the property is qualified. We also relax the requirement of an exact *MST*, which corresponds to the case when $\alpha = 1$. We will explain the reason shortly.

We observe that the algorithm framework for constructing *FT* spectral sparsifiers in Section 3.1 can be generalized to construct *FT* cut sparsifiers, if we replace *FT* spanners by *FT α-MST*, and then thoroughly set the parameters. This is inspired by sampling according to edge connectivities. *FT α-MST* can guarantee that even in the presence of at most f faults, each edge in the *MST* has a lower bound on the edge connectivity. To preserve edge connectivities, there is no requirement on the weight of edges in the cut defining the edge connectivities. Therefore, we define and use an approximate version of *FT MST*.

After replacing the first line in Algorithm 1 by constructing an $(f + C_\epsilon c \log w \log^3 n/\epsilon^2)$-*FT* $(\log n)$-*MST* J for G, where w is the maximum ratio of the weights of two edges in G, $C_\epsilon > 0$ and $c > 1$ are two new parameters, and also making according changes in Algorithm 2, we can get Algorithms 3 and 4 for constructing *FT* cut sparsifiers. Due to limit of space, we have moved the algorithms, and also the following theorem summarizing main results for *FT* cut sparsifiers, to the Appendix.

Theorem 7. *For an n-vertex m-edge graph G, a positive integer f, a parameter $\epsilon \in (0, 1)$, $\rho > 1$ a constant $C_\epsilon > 0$ and a parameter $c > 1$, Algorithm 4 constructs an f-VFT (f-EFT) $(1 \pm \epsilon)$-cut sparsifier for G of expected size $O(fn \log \rho + n \log^2 n \log^3 \rho/\epsilon^2 + m/\rho)$, with probability at least $1 - n^{-c}$.*

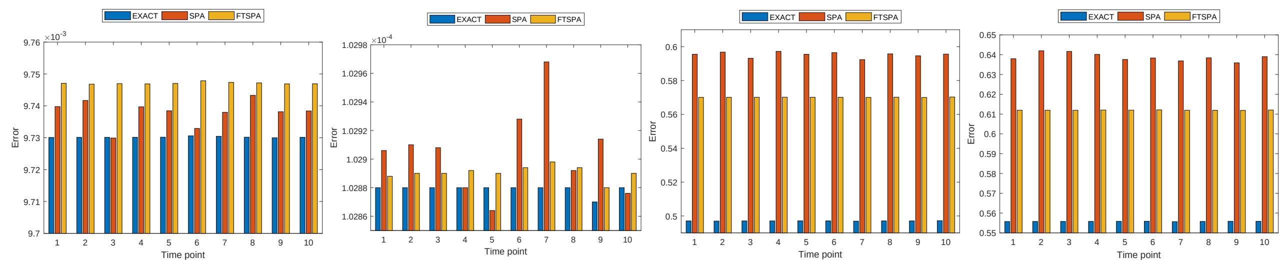
5. Stability Bounds for Subsequent Tasks

In this section, we give stability bounds to quantify the impact of the *FT* sparsification on the accuracy of subsequent graph learning tasks. The *FT* spectral sparsifiers have an important property that, for every time point t, the resulting subgraph H_t by applying all incoming updates at t on H_{t-1} is a spectral sparsifier of the current graph G_t. We then immediately obtain, for all time points, the upper bounds on the loss of the accuracy of subsequent graph learning tasks due to the *FT* sparsification, by applying existing results for static graphs to every graph G_t and its spectral sparsifier H_t. We describe the details as follows.

Laplacian-Regularized Estimation. Recall that $\hat{\beta}$ and $\tilde{\beta}$ are the estimations obtained using the exact graph G_t and the approximate subgraph H_t as in Equations 1 and 2, respectively. The bound in the loss of accuracy is upper bounded by $||\tilde{\beta} - \hat{\beta}||_2^2 \leq 2(1 + 2\epsilon)\lambda\hat{\beta}^T L_G \hat{\beta}$, if $\lambda' = 2\lambda$ (Sadhanala et al., 2016). A slightly better bound can be obtained if $\lambda' = \lambda$. As can be seen from the bound, we can smoothly tradeoff accuracy and computational cost of the estimation by setting the value of the parameter ϵ for an *FT* spectral sparsifier. A small value of ϵ results in a small accuracy loss but a large number of edges for the *FT* sparsifier, which implies a high computational cost, while a large value of ϵ results in a small sized *FT* sparsifier but a large accuracy loss.

Graph *SSL*. Because the original *HFS* in Equation (3) is not stable, we use a stable version, *Stable-HFS* for theoretical analysis.With an added regularization term $\mu = ((\frac{\gamma}{|S|}L_G + I_S)^+ yI_S)^T \mathbf{1}/(\frac{\gamma}{|S|}L_G + \mathbf{1})^+\mathbf{1}$, *Stable-HFS* has a new solution $\hat{\beta}_{STA} = (\frac{\gamma}{|S|}L_G + I_S)^+(yI_S + \mu\mathbf{1})$. Denote $\hat{R}(\beta) = \frac{1}{|S|}\sum_{i=1}^{|S|}(\beta(x_i) - y(x_i))^2$ as the empirical error and $R(\beta) = \frac{1}{n}\sum_{i=1}^{n}(\beta(x_i) - y(x_i))^2$ as the generalization error on all labeled and unlabeled vertices. It has been proven that $R(\tilde{\beta}) \leq \hat{R}(\hat{\beta}) + \beta + o(\beta)$, in Theorem 2 in (Calandriello et al., 2018). The last term $o(\beta)$ contains ϵ and increasing ϵ results in a constant increase in the bound. Then similar to Laplacian-regularized estimation, we can tradeoff accuracy and computational cost by correctly setting the value of ϵ.

There exists strong stability bounds for other spectral methods, *e.g.* Logistic Smoothing (Sadhanala et al., 2016), graph-regularized least squares, Laplacian SVM (Belkin et al., 2005) and *SC* (Ng et al., 2001). We do not review each of



(a) Estimation, $\sigma = 0.1$ (b) Estimation, $\sigma = 0.01$ (c) *SSL*, $\sigma = 0.1$ (d) *SSL*, $\sigma = 0.01$

Figure 3: Accuracy of Laplacian-regularized estimation and graph *SSL* for signals with Gaussian noise of $\sigma = 0.1$ and 0.01

them here, but give more discussions in the Appendix.

6. Experiments

We empirically show that the computational cost for maintaining spectral sparsifiers can be significantly improved for dynamic graphs, while the accuracy of subsequent Laplacian-regularized estimation and graph *SSL* are not significantly affected.

Dataset. We used the Facebook social network data from the *SNAP* [3]. The numbers of vertices and edges in the ego-combined graph are 4309 and 88234, respectively. A smooth signal β^* over the vertices was simulated following the approach in (Sadhanala et al., 2016). Then Gaussian noises from $N(0, \sigma)$ were added to form observations y. For every time point in $[1, 10]$, a random number of at most 200 insertions/deletions of randomly selected edges were generated. We can only simulate ten time points, because the baseline *SPA* (to be defined) incurs a high computational cost.

Methods. We compared our algorithm *FTSPA* with a baseline *SPA*, which constructs a spectral sparsifier from scratch at every time point. We did not include the dynamic algorithms (Kyng et al., 2017; Kapralov et al., 2014) because none of these algorithms has been implemented before. The performance measures include: (1) the update time representing the computational cost, and (2) the error $||\tilde{\beta} - \hat{\beta}||_2^2$ for Laplacian-regularized estimation, and the generalization error $R(\beta)$ for graph *SSL*. All the results were obtained by taking the average values from five runs of sparsification.

Results. We tested the errors and update times for four signals with different noises, $\sigma \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$. For Laplacian-regularized estimation, the parameter $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$, while for graph *SSL*, $\lambda \in \{10^{-6}, 10^{-4}, 10^{-2}, 10^{0}\}$. We will only report the smallest error achieved by different λ. In graph *SSL*, the labels were the sign of the signal β^*, and the number of revealed labeled vertices $l = 1000$ with 500 ones and 500 zeros, following (Calandriello et al., 2018). For our algorithm, the parameters were set to $f \in \{1, 3, 5, 7\}$, $\epsilon = 0.2$ and $\rho = 20$. Due to limit of space, we only report the results for $\sigma = \{0.1, 0.01\}$ and $f = 1$. The other results are similar and can be found in the Appendix.

[3]https://snap.stanford.edu/data/ego-Facebook.html

Methods	Update Time	Speedup	# Edges
SPA	34.2 s	1	12978 ± 30
FTSPA	0.3 ms	$> 10^5$	16502 ± 41

Table 1: Update time and # edges of *SPA* and *FTSPA*

As shown in Figure 3, for both tasks on both signals, the errors of *SPA* and *FTSPA* are not significantly affected by the sparfication. They were compared with the exact method *EXACT*, which computes the solution using the current graph G_t at a time point t. However, the average update time of *FTSPA* over all time points is only 0.3 miliseconds, significantly smaller than that of *SPA*, 34.2 seconds. The speedup is over 10^5. We observed that a small value of f, *e.g.* $f = 1$, is already enough to tolerant up to 200 edge updates on over 10^5 edges.

Moreover, the number of edges in *SPA* and *FTSPA* are only about 13K and 16.5K respectively, significantly smaller than the number of edges in the original graph, about 88K, as shown in Table 1. Then the computational cost of subsequent Laplacian-regularized estimation and graph *SSL* are significantly improved.

7. Conclusion and Future Work

Graph sparsification has been widely used in large-scale graph learning, *e.g.*, Laplacian-regularized estimation, graph *SSL* and *SC*. However, when graphs vary over time, existing methods require polynomial order computational cost per update. We reduce the cost to to only a constant by proposing *FT* spectral sparsification and cut sparsification. We then analyze the stability bounds for subsequent learning tasks, *e.g.*, Laplacian-regularized estimation and graph *SSL*. Extensive experiments have validated the computational efficiencies and accuracies of the proposed methods. As the future work, we will study the existence of *FT* sparsifiers for digraphs (Cohen et al., 2017; Zhu & Lam, 2017; 2018). Proving the size lower bounds for the *FT* spectral sparsification and cut sparsification is also an interesting direction.

Acknowledgements

We thank the four reviewers for their insightful comments. This work was supported by NSF grants: DBI-1356655, CCF-1514357 and IIS-1718738. Jinbo Bi was also supported by NIH grants 5R01DA037349-04 and 5K02DA043063-03.

References

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

Ahn, K. J., Guha, S., and McGregor, A. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of PODS Conference*, pp. 5–14, 2012.

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

Belkin, M., Niyogi, P., and Sindhwani, V. On manifold regularization. In *Proceedings of AISTATS Conference*, pp. 1, 2005.

Benczur, A. and Karger, D. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of ACM STOC Conference*, pp. 47–55, 1996.

Blum, A. and Chawla, S. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of ICML Conference*, pp. 19–26, 2001.

Bodwin, G., Dinitz, M., Parter, M., and Williams, V. Optimal vertex fault tolerant spanners (for fixed stretch). In *Proceedings of SIAM SODA Conference*, pp. 1884–1900, 2018.

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.

Chechik, S., Langberg, M., Peleg, D., and Roditty, L. Fault-tolerant spanners for general graphs. *SIAM Journal on Computing*, 39(7):3403–3423, 2010.

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

Cohen, M., Kelner, J., Peebles, J., Peng, R., Rao, A., Sidford, A., and Vladu, A. Almost-linear-time algorithms for Markov chains and new spectral primitives for directed graphs. In *Proceedings of STOC Conference*, pp. 410–419, 2017.

Coulouris, G. F. and Dollimore, J. *Distributed systems: concepts and design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.

Dinitz, M. and Krauthgamer, R. Fault-tolerant spanners: better and simpler. In *Proceedings of ACM PODC Conference*, pp. 169–178, 2011.

Doyle, P. and Snell, J. Random walks and electric networks. *https://arxiv.org/abs/math/0001057*, 2000.

Fung, W., Hariharan, R., Harvey, N. J., and Panigrahi, D. A general framework for graph sparsification. In *Proceedings of STOC Conference*, pp. 71–80, 2011.

Harvey, N. Matrix concentration and sparsification. In *Workshop on Randomized Numerical Linear Algebra: Theory and Practise*, 2012.

Kapralov, M. and Panigrahy, R. Spectral sparsification via random spanners. In *Proceedings of ITCS Conference*, pp. 393–398, 2012.

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

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

Koutis, I. and Xu, S. Simple parallel and distributed algorithms for spectral graph sparsification. *ACM Transactions on Parallel Computing*, 3(2):14, 2016.

Kyng, R., Pachocki, J., Peng, R., and Sachdeva, S. A framework for analyzing resparsification algorithms. In *Proceedings of SIAM SODA Conference*, pp. 2032–2043, 2017.

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

Levcopoulos, C., Narasimhan, G., and Smid, M. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proceedings of ACM STOC Conference*, pp. 186–195, 1998.

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

Moses, C. and Vaggos, C. Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of SODA Conference*, pp. 841–854, 2017.

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

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

Reif, J. H. *Synthesis of parallel algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

Rustamov, R. and Klosowski, J. Interpretable graph-based semi-supervised learning via flows. In *Proceedings of AAAI Conference*, pp. 3976–3983, 2018.

Sadhanala, V., Wang, Y.-X., and Tibshirani, R. J. Graph sparsification approaches for Laplacian smoothing. In *Proceedings of AISTATS Conference*, pp. 1250–1259, 2016.

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

Spielman, D. and Teng, S.-H. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of STOC Conference*, pp. 81–90, 2004.

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

Tropp, J. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

Wang, J., Jebara, T., and Chang, S.-F. Semi-supervised learning using greedy max-cut. *Journal of Machine Learning Research*, 14:771–800, 2013.

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

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

Zhu, X., Ghahramani, Z., and Lafferty, J. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML Conference*, pp. 912–919, 2003.