Improved Dynamic Graph Learning through Fault-Tolerant Sparsification - Supplement

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

A. More Studies on Fault-tolerant Subgraphs

(Parter & Peleg, 2013) studied FT-BFS tree of optimal size $O(n^{1.5})$, which contains a BFS tree from a source in the presence of 1 vertex/edge failure. The extensions to 2 failures and approximate BFS trees are referred to, e.g. (Parter, 2015; Parter & Peleg, 2014). Distance/connectivity sensitivity oracles are data structures similar to FT spanners, but not restricted to graph structures, e.g. (Demetrescu et al., 2008; Duan & Pettie, 2017). (Chechik et al., 2017) constructs an oracle of super-quadratic size capable of answering $(1 + \epsilon)$-approximate distance query resilient to $f = o(\log n / \log \log n)$ failures. Many other related research on FT graph structures include but not limited to FT routing schemes (Chechik, 2011), FT labelling (Abraham et al., 2016), and FT reachability preservers (Baswana et al., 2016).

B. Proof of Theorem 1

In this section, we prove Theorem 1, where the bounds are achieved by Algorithm 2 (FTSS).

Proof. (Theorem 1.) By Theorem 4 in the paper and Induction, for every $i \in [1, \lceil \log \rho \rceil]$, the event that the inequality

$$(1 - \epsilon / \lceil \log \rho \rceil)^j L_{G_i - F} \leq L_{G_i - F} \leq (1 + \epsilon / \lceil \log \rho \rceil)^j L_{G_i - F}$$

holds and the expected size is

$$O(fn_i + n \log^2 n \log^2 \rho / \epsilon^2 + m / 2^i),$$

happens with probability at least $(1 - 1/n^2)^i$. Since FTSS outputs $H = G_{\lceil \log \rho \rceil}$ when $i = \lceil \log \rho \rceil$ as the final FT spectral sparsifier, it satisfies the desirable properties w.h.p.

C. Parallel and Distributed Algorithms

The CRCW PRAM model is a shared memory architecture supporting concurrent read and concurrent write operations into any memory location by multiple processors. When there are simultaneous write operations by two or more processors into a memory location, both of them will succeed. The performance measures are the completion time as well as the work complexity defined as the sum of the total number of time points each processor is working. For the synchronized distributed model, there is an underlying network where each node has its own memory and a processor and any two nodes are connected by a link if there is an edge in the network. The computation operates in synchronized rounds, where each round involves passing of messages through links following by local computation at each node. The performance measures include number of rounds, total number of messages (communication complexity) and maximum length of any message.

We first show parallel and distributed algorithms for constructing $f$-EFT $(\log n)$-spanners $H$ for an input graph $G$. Combining (Chechik et al., 2010) and (Baswana & Sen, 2007), we have the following algorithms through $f$ iterations of computation. In the $i$th iteration, we construct a spanner $H_i$ for the graph $G - \sum_{j=1}^{i-1} H_j$ by the graph spanner algorithms of Baswana and Sen under either the CRCW PRAM model or the synchronized distributed model (Baswana & Sen, 2007). Edges in $H_j$ with $j < i$ will not participate in the computation by declaring themselves out. The returned $H$ is the union of the edges in each $H_i$. We then have the following two theorems by combining Theorem 5.2 from (Chechik et al., 2010) with Theorems 5.1 and 5.4 from (Baswana & Sen, 2007).

Theorem 1. For an $n$-vertex $m$-edge graph $G$, an $f$-EFT $(\log n)$-spanner for $G$ of expected size $O(fn \log n)$ can be constructed in the CRCW PRAM model, w.h.p. using $O(fm n \log n)$ work in $O(f \log n)$ time.

Theorem 2. For an $n$-vertex $m$-edge graph $G$, an $f$-EFT $(\log n)$-spanner for $G$ of expected size $O(fn \log n)$ can be constructed in the synchronized distributed model in $O(f \log^2 n)$ rounds and $O(fm n \log n)$ communication complexity, using messages of size $O(\log n)$.
Algorithms 1 and 2 can be extended to the CRCW PRAM and the synchronized distributed models by using the above algorithm in the respective model for constructing $f$-EFT $(\log n)$-spanners in Line 1 of Algorithm 1 (Light-FTSS). In the rest part of Algorithm 1, the uniform sampling for each edge is independent and can be naturally implemented in the parallel and distributed models. Algorithm 2 only calls Algorithm 1 (in the parallel or distributed model) for a few times sequentially.

We now prove the complexity summarized in Theorems 5 and 6. We first prove the complexity of Algorithm 1. By a proof similar to that of Theorem 6, we can prove that w.h.p. the returned $H$ is an $f$-VFT $((1+\varepsilon)$-spectral sparsifier for $G$. However, the size bound becomes $O(fn\log n+n\log^3 n/\varepsilon^2+m/2)$ because the size of spanners by the parallel and distributed EFT-spanner algorithms is larger by a factor of $\log n$ according to Theorems 1 and 2. For the parallel algorithm, the work complexity of Line 1 (spanner construction) is $O(fm\log n+mn\log^3 n/\varepsilon^2)$ and the work of Line 3 (random sampling) is $O(m)$, which is dominated by Line 1. The time complexity of Lines 1 and 3 are $O(f\log n+\log^3 n/\varepsilon^2)$ and $O(1)$, resp., respectively. Therefore, its work and time complexity are $O(fm\log n+mn\log^3 n/\varepsilon^2)$ and $O(f\log n+\log^3 n/\varepsilon^2)$, resp., for the distributed algorithm. The number of rounds and the communication complexity of Line 1 are $O(f\log^2 n+\log^4 n/\varepsilon^2)$ and $O(fm\log n+mn\log^3 n/\varepsilon^2)$, which also dominate one and $m$, resp., of Line 3. So the total number of rounds and the communication complexity are the same as those of Line 1. We then prove the complexity of Algorithm 2. By applying the result of Algorithm 1 a logarithmic number of times as in the proof of Theorem 1, we can prove the desirable complexity summarized in Theorems 2 and 3.

### D. Proposed Algorithms for FT Cut Sparsifiers and Their Proof

The algorithms are summarized in Algorithms 3 and 4. Now we prove the following theorem summarizing the main properties of Algorithm 3. We will use Theorem 4 in the proof.

**Theorem 3.** For an $n$-vertex $m$-edge graph $G$, a positive integer $f$, a parameter $\varepsilon \in (0,1)$, a constant $C_{\varepsilon} > 0$ and a parameter $c > 1$, Algorithm 3 constructs an $f$-VFT $(f$-EFT $((1+\varepsilon)$-cut sparsifier for $G$, with probability at least $1-n^{-c}$.

**Proof.** (Theorem 3.) Suppose without loss of generality that the maximum edge weight in $G$ is 1. Following a common technique handling weighted graphs (Roditty et al., 2008), we decompose $G$ into $\log w$ edge-joint subgraphs $G_i$ for $i \in [1,\lfloor \log w \rfloor]$, where each $G_i$ contains the edges with weights in the interval $(2^{1-(\varepsilon+1)},2^2]$ and also the edges in $J_i = J_i^{\varepsilon}/2^{-(\varepsilon+1)}$ with $J$ being the FT MST returned by Line 2 in Algorithm 3.

By the definition of $f$-FT $\alpha$-MST, even in the presence of at most $f$ faults $F$, the connectivities of each edge $G_i - F - J_i$ in $G_i - F$ is at least $C_{\varepsilon}c\log w \log^2 n/\varepsilon^2 = 4\rho c\varepsilon$, for $c' = c \log w$ and $\rho = C_{\varepsilon}c' \log^2 n/4\varepsilon^2$. Assume that all edges in $J_i$ have weights in $(2^{1-(\varepsilon+1)},2^2]$. We can then apply Theorem 4 by setting $p_e = 1$ for $e \in J_i - F$, and $p_e = 0.25$ for $e \in G_i - F - J_i$. In this way, we get that $H_i - F$ is a $(1+\varepsilon)$-cut sparsifier of $G_i - F$ with probability $1-n^{-c \log w}$. By definition, $H_i$ is an $f$-FT $(1+\varepsilon)$-cut sparsifier of $G_i$ with probability $1-n^{-c \log w}$. It is easy to see the decomposability of $FT$ $(1+\varepsilon)$-cut sparsifier. Then, $H$ is an $f$-FT $(1+\varepsilon)$-cut sparsifier of $G$ with probability $1-n^{-c \log w}$.

We can remove the assumption on $J_i$ as follows. One can find a subgraph $J_i'$ of $J_i$, by possibly splitting weights and dropping small weights, such that $J_i'$ is an $f$-FT $\alpha$-MST of $G_i$. We can then apply the previous paragraph on $G_i' = (G_i - J_i') \cup J_i'$ to get that $H_i'$ is an $f$-FT $(1+\varepsilon)$-cut sparsifier of $G_i'$. Since $G_i = G_i' \cup (J_i - J_i')$ and $H_i = H_i' \cup (J_i - J_i')$, we have that $H_i$ is an $f$-FT $(1+\varepsilon)$-cut sparsifier of $G_i$. The proof of Theorem 7 is similar to the proof of Theorem 1, and thus is omitted here.
E. More Stability Bounds for Subsequent Learning Tasks

Spectral clustering is to find \( k \) disjoint subset assignment such that the assignments are smooth over the underlying graph. Let \( \beta_c \) with \( c \in [1, k] \) be the cluster indicator vectors and \( \beta_{n \times k} \) be the matrix containing those \( k \) indicators as columns. Note that the columns in \( \beta \) are orthonormal to each other, i.e., \( \beta^T \beta = I \). SC can be formulated as the following optimization problem relaxed from the NP-hard problem of computing the minimum of Ratio-Cut (Von Luxburg, 2007).

\[
\hat{\beta} = \arg\min_{\beta : \beta^T \beta = I, \beta_c \neq 1} \text{Trace}(\beta^T L G \beta).
\]

(1)

By using a spectral sparsifier \( H \) of \( G \) instead of \( G \), we solve the following problem.

\[
\hat{\beta} = \arg\min_{\beta : \beta^T \beta = I, \beta_c \neq 1} \text{Trace}(\beta^T L H \beta).
\]

(2)

By a simple analysis, we have that for every time point \( t \),

\[
\text{Trace}(\beta^T L_{H_t} \beta) \leq (1 + \epsilon) \text{Trace}(\beta^T L_G \beta).
\]

(3)

This implies that the clusterings by running the k-means algorithm on \( \hat{\beta} \) and \( \hat{\beta} \) are similar.

F. More Experimental Results

We implemented the sparsification in Java based on the JGraphT library \(^1\), and the computation involved by matrices in Matlab. For the sparsification algorithm, we followed the insight that in practice \( O(\log n / \epsilon^2) \) spanners each with \( O(n) \) edges is often enough to obtain a \( (1 \pm \epsilon) \)-spectral sparsifier, although the theory requires \( O(\log^2 n / \epsilon^2) \) spanners each with \( O(n \log n) \) edges (Sadhanala et al., 2016).

Figure 1 summarizes the errors of Laplacian-regularized estimation and graph SSL for different values of \( f \in \{1, 3, 5, 7\} \). The average update time of 3-FTSPA, 5-FTSPA and 7-FTSPA are 0.26, 0.25 and 0.27 milliseconds, and thus the speedups remain to be over \( 10^5 \).

References


\(^1\)jgrapht.org
Figure 1: Errors of Laplacian-regularized estimation and graph SSL for signals with Gaussian noise of $\sigma = 0.1$ and $0.01$