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

$$(1-\epsilon/\lceil \log \rho \rceil)^i L_{G-F} \preceq L_{G_i-F} \preceq (1+\epsilon/\lceil \log \rho \rceil)^i L_{G-F}$$

holds and the expected size is

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

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.. □

[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 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

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 $O(fm\log n)$ work in $\tilde{O}(f\log n)$ 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 $O(fm\log n)$ communication complexity, using messages of size $O(\log n)$.*

Algorithm 1 *Light-FTCS*

Require: $G(V, E), f > 0, \epsilon \in (0, 1), C_\epsilon > 0, c > 1$
Ensure: H
  Set w as the maximum ratio of the weights of two edges in G;
  Construct an $(f + C_\epsilon c \log w \log^3 n/\epsilon^2)$-*FT* $(\log n)$-MST 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 *FTCS*

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

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* (f-*EFT*) $(1 \pm \epsilon)$-spectral sparsifier for G. However, the size bound becomes $O(fn \log n + n \log^3 n/\epsilon^2 + m/2)$ 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 $O(fm \log n + m \log^3 n/\epsilon^2)$ 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 $\tilde{O}(f \log n + \log^3 n/\epsilon^2)$ and $O(1)$, resp.. Therefore, its work and time complexity are $O(fm \log n + m \log^3 n/\epsilon^2)$ and $\tilde{O}(f \log n + \log^3 n/\epsilon^2)$, resp.. For the distributed algorithm, the number of rounds and the communication complexity of Line 1 are $O(f \log^2 n + \log^4 n/\epsilon^2)$ and $O(fm \log n + m \log^3 n/\epsilon^2)$, 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 $\epsilon \in (0, 1)$, a constant $C_\epsilon > 0$ and a parameter $c > 1$, Algorithm 3 constructs an f-VFT (f-EFT) $(1 \pm \epsilon)$-cut sparsifier for G, with probability at least $1 - n^{-c}$.*

Theorem 4. *(Fung et al., 2011) Let H be obtained from a graph G with weights in $(1/2, 1]$ by independently sampling edge e with probability $p_e \geq \rho/\lambda_G(e)$, where $\rho = C_\epsilon c \log^2 n/4\epsilon^2$, and $\lambda_G(e)$ is the local edge connectivity of edge e, C_ϵ is an explicitly known constant. Then H is a $(1 \pm \epsilon)$-cut sparsifier with probability $1 - n^{-c}$.*

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 $i \in [1, \lceil \log w \rceil]$, where each G_i contains the edges with weights in the interval $(2^{-(i+1)}, 2^i]$ and also the edges in $J_i = J/2^{-(i+1)}$ with J being the *FT MST* returned by Line 2 in Algorithm 3.

By the definition of f-*FT* α-*MST*, even in the presence of at most f faults F, the connectivies of each edge of $G_i - F - J_i$ in $G_i - F$ is at least $C_\epsilon c \log w \log^2 n/\epsilon^2 = 4\rho c'$, for $c' = c \log w$ and $\rho = C_\epsilon c' \log^2 n/4\epsilon^2$. Assume that all edges in J_i have weights in $(2^{-(i+1)}, 2^i]$. 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 $(1 \pm \epsilon)$-cut sparsifier of $G_i - F$ with probability $1 - n^{-c \log w}$. By definition, H_i is an f-*FT* $(1 \pm \epsilon)$-cut sparsifier of G_i with probability $1 - n^{-c \log w}$. It is easy to see the decomposibility of *FT* $(1 \pm \epsilon)$-cut sparsifier. Then, H is an f-*FT* $(1 \pm \epsilon)$-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* α-*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* $(1 \pm \epsilon)$-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* $(1 \pm \epsilon)$-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 β_c with $c \in [1, k]$ be the cluster indicator vectors and $\boldsymbol{\beta}_{n\times k}$ be the matrix contaning those k indicators as columns. Note that the columns in $\boldsymbol{\beta}$ are orthonormal to each other, *i.e.*, $\boldsymbol{\beta}^T\boldsymbol{\beta} = I$. *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).

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}:\boldsymbol{\beta}^T\boldsymbol{\beta}=I, \beta_c \perp \mathbf{1}}{\arg\min} Trace(\boldsymbol{\beta}^T L_G \boldsymbol{\beta}). \tag{1}$$

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

$$\tilde{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}:\boldsymbol{\beta}^T\boldsymbol{\beta}=I, \beta_c \perp \mathbf{1}}{\arg\min} Trace(\boldsymbol{\beta}^T L_H \boldsymbol{\beta}). \tag{2}$$

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

$$Trace(\boldsymbol{\beta}^T L_{H_t} \boldsymbol{\beta}) \leq (1+\epsilon) Trace(\boldsymbol{\beta}^T L_{G_t} \boldsymbol{\beta}). \tag{3}$$

This implies that the clusterings by running the k-means algorithm on $\tilde{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\beta}}$ 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 miliseconds, and thus the speedups remain to be over 10^5.

References

Abraham, I., Chechik, S., Gavoille, C., and Peleg, D. Forbidden-set distance labels for graphs of bounded doubling dimension. *ACM Transactions on Algorithms*, 12 (2):22, 2016.

Baswana, S. and Sen, S. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, 30(4):532–563, 2007.

Baswana, S., Choudhary, K., and Roditty, L. Fault tolerant subgraph for single source reachability: generic and optimal. In *Proceedings of SIAM SODA Conference*, pp. 509–518, 2016.

Chechik, S. Fault-tolerant compact routing schemes for general graphs. In *Proceedings of ICALP Conference*, pp. 101–112, 2011.

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

Chechik, S., Cohen, S., Fiat, A., and Kaplan, H. $1+\epsilon$-approximate f-sensitive distance oracles. In *Proceedings of SIAM SODA Conference*, pp. 1479–1496, 2017.

Demetrescu, C., Thorup, M., Chowdhury, R., and Ramachandran, V. Oracles for distances avoiding a failed node or link. *SIAM Journal on Computing*, 37(5):1299–1318, 2008.

Duan, R. and Pettie, S. Connectivity oracles for graphs subject to vertex failures. In *Proceedings of SIAM SODA Conference*, pp. 490–509, 2017.

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.

Parter, M. Dual failure resilient BFS structure. In *Proceedings of ACM PODC Conference*, pp. 481–490, 2015.

Parter, M. and Peleg, D. Sparse fault-tolerant BFS trees. In *Proceedings of ESA Conference*, pp. 779–790, 2013.

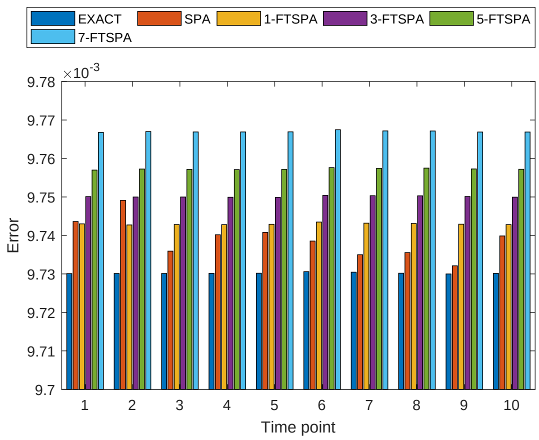
Parter, M. and Peleg, D. Fault tolerant approximate BFS structures. In *Proceedings of SIAM SODA Conference*, pp. 1073–1092, 2014.

Roditty, I., Thorup, M., and Zwick, U. Roundtrip spanners and roundtrip routing in directed graphs. *ACM Transactions on Algorithms*, 4(3), 2008.

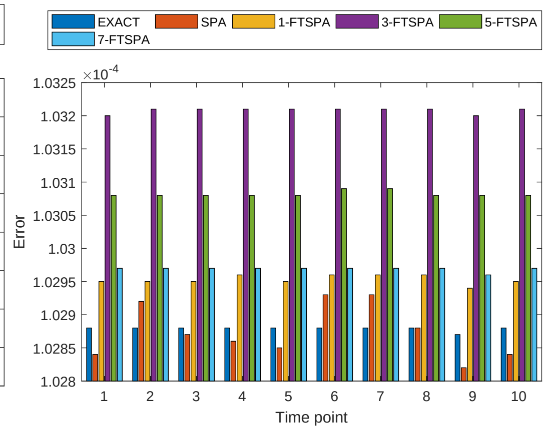
Sadhanala, V., Wang, Y.-X., and Tibshirani, R. J. Graph sparsification approaches for Laplacian smoothing. In *Proceedings of AISTATS Conference*, pp. 1250–1259, 2016.

Von Luxburg, U. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

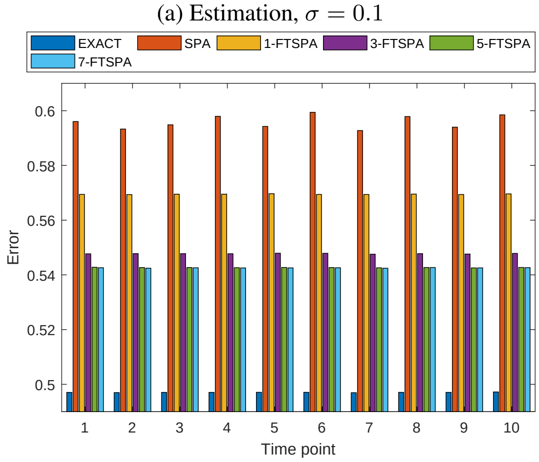
[1] jgrapht.org



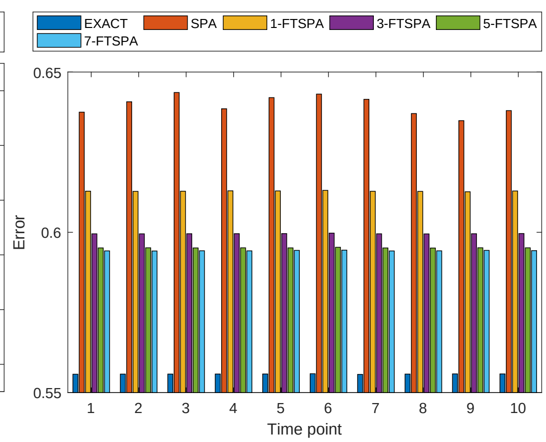
(a) Estimation, $\sigma = 0.1$



(b) Estimation, $\sigma = 0.01$



(c) *SSL*, $\sigma = 0.1$



(d) *SSL*, $\sigma = 0.01$

Figure 1: Errors of Laplacian-regularized estimation and graph *SSL* for signals with Gaussian noise of $\sigma = 0.1$ and 0.01