Fault Tolerant Distance Preservers
SRIP 2018

Debanuj Nayak & Dr. Manoj Gupta (Advisor)

Department of Computer Science and Engineering, IIT Gandhinagar

July 8, 2018

Acknowledgements

I am very grateful to my SRIP advisor Dr. Manoj Gupta for guiding me through the whole research experience. I would also like to thank IIT Gandhinagar and the SRIP organizing team for providing us with this opportunity.

Contents

Abstract

Preservers are subgraphs that preserve the distance between a given pair of vertices in a graph or network. Real world networks however are prone to failures, thus it is important to study fault tolerant versions of these subgraphs.

Fault Tolerant Distance Preservers can preserve distance even after a set of edge failures in the graph. These have been studied before, but the construction has been very complex.

The problem is that given a graph G , we have to construct a k - Fault Tolerant Distance preserver , in other words a subgraph of G which preserves distances after k edge failures.

In this paper we present a simple and elegant way of constructing k Fault Tolerant Distance preservers for k=1 and k=2.

1 Introduction

A k - fault tolerant distance preserver also called a k - FT preserver is a subgraph of the original graph that has the property that the distance between a single source vertex s to any other vertex t $\in$ V after k edge failures in this subgraph is same as the distance between the s and t after k edge failures in the original graph. The precise definition is as follows

Definition 1.1 (Fault Tolerant Distance Preserver)**.** Given an n-node directed graph G =(V,E) and a designated source vertex s, a subgrapgh H $\subseteq$ G is called a k- fault tolerant distance preserver if for every set of failed edges F , the following is true

$$distance_{H-F}(s,t) = distance_{G-F}(s,t)\ \forall t \in V \text{ and } F \subseteq E, |F| < k$$

In this paper we provide a simple and elegant way of constructing k - FT presevers for k=1 and k=2.

2 Prior Work on this problem

Fault Tolerant Preservers have been studied before. Parter and Peleg provided the construction of a 1-FT preserver in $O(n^{3/2})$space.[7] Parter later also constructed a 2-FT preserver in $O(n^{5/3})$space. Parter also gave a lower bound of $\Omega(n^{2-1/k+1})$.[6] Bodwin et.al. provided the construction of k-FT preservers for any general k in $O(kn^{2-1/2^k})$.[2]

3 High Level Overview of our Technique

We solve the problem of fault tolerant distance preservers by extending the idea of fault tolerant reachability.

Fault tolerant reachability is a similar problem where one has to construct a fault tolerant reachability subgraph or FTRS. A k FTRS preserves the reachability from a source vertex s to any other vertex t $\in$ V after k edge failures in the subgraph if and only if t is reacahble from s in the original graph after k failures. Baswana et.al solved the problem of constructing a k-FTRS for any general k.[1]

Let the shortest distance from source vertex s to t be d and let the shortest distance from s to t after k failures be $d + j$ for some j.

Rather than maintaining reachability on the original graph G as done by baswana et.al. , we separately maintain reachability from s to t on subgraphs G_i where G_i represents the union of $d+i$ length paths from s to t.

The new distance from s to t is $d + j$ which means t is reachable from s on the graph G_j after k failures but is not reachable from s for any other G_i for $i < j$. Thus k-FTRS of G_j will preserve this path and k-FTRS of all G_i for $i < j$ will not preserve this path. The union of the k-FTRS of all the G_i 's gives a k-FT distance preserver.

4 k-Fault Tolerant Reachability Subgraph

In this section we will see how to construct a k-FTRS in general.

4.1 Definition of k-FTRS

The precise definition of a k-Fault Tolerant Reachability Subgraph or k-FTRS is as follows

Definition 4.1 (Fault Tolerant Reachability Subgraph)**.** Given an n-node directed graph G =(V,E) and a designated source vertex s, a subgrapgh H $\subseteq$ G is called a k- fault tolerant reachability subgraph or k-FTRS if for every set of k failed edges F , the following is true

A vertex t $\in$ V is reachable from s in G $-$ F if and only if t is reachable from s in H $-$ F

Definition 4.2 (k - FTRS(t))**.** Given a vertex t $\in$ V, a subgraph H $\subseteq$ G is called a k-FTRS(t) if for every set F of k edges t is reachable from s in G-F if and only if v is reachable from s in H-F.

This definition thus helps us to define k-FTRS in an alternate way. We can say that a subgraph H is a k-FTRS if it is a k-FTRS(t) $\forall$ t $\in$ V. The following lemma called the locality lemma gives us a way of calculating the k - FTRS.

Lemma 4.1 (Locality Lemma)**.** *If there is an algorithm that can compute k-FTRS(t) where the in-degree of each vertex is bounded by a constant c then on applying this algorithm on all the vertices of G will give us a k-FTRS where the degree of each vertex is bounded by a c.*

Thus all we need to do is to create an algorithm for computing k - FTRS(t) for any t $\in$ V. From here on unless otherwise specified , we will be working on k- FTRS(t).

4.2 Fault Tolerant Reachability and its relation to Max Flow Min Cut

Theorem 4.2 (Max Flow Min Cut Theorem)**.** *This theorem states that the maximum flow that can be achieved from s to t in a graph G is equal to the size of the minimum cut separating s and t.*

Corollary 4.2.1. *For an unweighted graph G , for any positive integer α, there is a flow of value α from a vertex set S to a destination vertex t, if and only if there are α edge disjoint paths from S to t.*

We define another term called as the Farthest Min Cut, which is very crucial for computing the k -FTRS and also k - FT preserver.

Definition 4.3 (Farthest Min Cut)**.** Let S be a source set and t be a destination vertex in any graph G and suppose for any (S-t) min cut C, (A(C),B(C)) = Partition(G,C). Any (S, t)-min-cut C_{far} is called farthest min-cut, denoted by Farthest Min Cut(G, S, t), if for any other (S, t)-min-cut C, it holds that A(C) $\subseteq$ A(C_{far})

4.3 Brief Overview of computing k-FTRS

The Farthest Min Cut is unique for a a given graph , source set S and destination vertex. If the max-flow from s to t in G is k+1 or greater, then we can define k-FTRS(t) to be any k + 1 edge disjoint paths from s to t.

When the max flow is exactly k , let us suppose that there exists a path P from s to t in G-F, where F is the set of failing edges. Then P must pass through an edge, say $(a_i; b_i)$, of the farthest min-cut. If we include b_i in the source then the max flow increases. Thus we get at least k + 1 edge disjoint paths from the set $\{s; b_i\}$ to t. This suggests that a subgraph H of G that contains k + 1 edge disjoint paths from $\{s; b_i\}$ to t for each i, will serve as a k-FTRS(t).

For the case when (s; t) maxflow in G is less than k, we compute a series of farthest min-cuts built on a hierarchy of nested source sets. Our construction consists of k rounds. It starts with a source set S containing the singleton vertex s. In each iteration we add to the previous source S, the endpoints bi's of the edges corresponding to the farthest (S; t) min-cut. The size of the cuts in this hierarchy governs the degree of t in k-FTRS(t). In order to get a bound on the size of these cuts, we transform G into a new graph with H with O(m) vertices and edges.

We construct this new graph H =(V',E') in the following way.For each u in V we construct a binary tree B_u such that the number of leaves in B_u is exactly equal to degree (say d(u)) of u in G. Now the out-degree of each vertex in H is bound by two.

4.4 Algorithm for computing k-FTRS

The procedure briefly explained in the previous section creates a k - FTRS, where the in-degree of each vertex is at most 2^k. There are n vertices, so the k-FTRS requires $2^k n$ many edges. The k - FTRS algorithm is as follows.[1]

Algorithm 1 k-FTRS(t)

1: $S_1 \leftarrow \{s\}$
2: **for** *i = 1 to k* **do**
3: $C_1 \leftarrow$ *Farthest Min Cut(G,*S_i*,t)*
4: (A_i,B_i) $\leftarrow$ *Partition(*C_1*)*
5: $S_{i+1} \leftarrow (A_i \cup OUT(A_i)) - \{t\}$
6: $f \leftarrow$ *Max Flow from S to t*
7: *Add only those edges to t whaich are adjacent to t and are present in E(f)*

5 Computing k-Fault Tolerant Distance Preservers for k=1,2

Just like K-FTRS could be defined in an alternate way using k-FTRS(t), k=FT distance preserver can be defined using k-FT distance preserver(t).

Definition 5.1 (k-FT distance preserver(t))**.** Given a vertex t $\in$ V, a subgraph H $\subseteq$ G is called a k-FTRS(t) if for every set F of k edges

$$distance_{H-F}(s,t) = distance_{G-F}(s,t)\ \forall t \in V$$

A subgraph H is a k-FT distance preserver for a graph G if and only if H is a k-FT distance preserver(t) $\forall$ t in V.The same Locality Lemma can also be applied in the context of k-FT distance preservers.

Lemma 5.1 (Locality Lemma for k-FT distance preservers)**.** *If there is an algorithm that can compute k-FT distance preserver(t) where the in-degree of each vertex is bounded by a constant c then on applying this algorithm on all the vertices of G will give us a k-FT distance preserver where the degree of each vertex is bounded by a c.*

Thus we need to develop an algorithm that can compute k-FT distance preserver(t).
The subgraphs k-FTRS(t) only preserve reachability after k faults. However, a k-FT distance preserver has to preserve the distance.To achieve this k-Fault Tolerant reachability property is applied individually on subgraphs G_i where G_i represents the union of $d + i$ length paths where d is the length of the shortest path from s to t. The algorithm thus tries to compute edge disjoint paths of each and every length possible.

5.1 Algorithm for 1-Fault Tolerant Distance Preserver

Algorithm 2 1-FT distance preserver(t)

$G_0 \leftarrow G$
for i from 0 to $\sqrt{n}$ **do**
    G' $\leftarrow$ *Shortest Path Subgraph*(G_i)
    $C_1 \leftarrow$ *Farthest Min Cut(G',s,t)*
    if *Size of* C_1 *is 2* **then**
        Let C_1 *be* $\{u_1$-v_1, u_2-$v_2\}$
        $S \leftarrow \{s, v_1, v_2\}$
        $f \leftarrow$ *Max Flow from S to t*
        Add the new ending edges of E(f) on t
        EXIT
    else
        Let C_1 *be* $\{u_1$-$v_1\}$
        $S \leftarrow \{s, v_1\}$
        $C_2 \leftarrow$ *Farthest Min Cut(G',S,t)*
        Let C_1 *be* $\{u_2$-v_2, u_3-$v_3\}$
        $f \leftarrow$ *Max Flow from S to t*
        Add the new ending edges of E(f) on t
        $G_{i+1} \leftarrow G$ - C_1

After the application of the above algorithm , still some paths may be left. These remaining paths from s to t will have a distance > d+$\sqrt{n}$ .Here we can use Parter and Peleg's method[7] , and can conclude that at most $\sqrt{n}$ new ending paths from s to t will remain. Add the last edges of these $\sqrt{n}$ paths on t.

5.2 Algorithm for 2-Fault Tolerant Distance Preserver

Just like the 1-FT distance preserver case, After the application of the above algorithm , still some paths may be left. These remaining paths from s to t will have a distance > d+ $n^{2/3}$ and so we can conclude that at most $n^{1/3}$ new ending paths from s to t will remain using Parter's method.[6] Add the last edges of these $n^{1/3}$ paths on t.

Algorithm 3 2-FT distance preserver(t)

$G_0 \leftarrow G$
for i from 0 to $\sqrt{\text{n}}$ **do**
    G' $\leftarrow$ *Shortest Path Subgraph*(G_i)
    $C_1 \leftarrow$ *Farthest Min Cut(G',s,t)*
    if *Size of* C_1 *is 2* **then**
        Let C_1 *be* $\{u_1$-v_1, u_2-$v_2\}$
        $S \leftarrow \{s, v_1, v_2\}$
        $C_2 \leftarrow$ *Farthest Min Cut(G',S,t)*
        Let C_2 *be* $\{u_3$-v_3, u_4-v_4, u_5-$v_5\}$
        $S \leftarrow \{s, v_1, v_2, , v_3, v_4, v_5\}$
        $f \leftarrow$ *Max Flow from S to t*
        Add the new ending edges of E(f) on t
        $G_{i+1} \leftarrow G$ - C_1
    else
        Let C_1 *be* $\{u_1$-$v_1\}$
        $S \leftarrow \{s, v_1\}$
        $C_2 \leftarrow$ *Farthest Min Cut(G',S,t)*
        Let C_2 *be* $\{u_2$-v_2, u_3-$v_3\}$
        $S \leftarrow \{s, v_1, v_2, v_3\}$
        $C_3 \leftarrow$ *Farthest Min Cut(G',S,t)*
        Let C_3 *be* $\{u_4$-v_4, u_5-v_5, u_6-$v_6\}$
        $S \leftarrow \{s, v_1, v_2, v_3, v_4, v_5, v_6\}$
        $f \leftarrow$ *Max Flow from S to t*
        Add the new ending edges of E(f) on t
        $G_{i+1} \leftarrow G$ - C_2

5.3 Proof of the above algorithms

Both the above algorithms are based on the idea of constructing FTRS on subgraphs which contain only paths of a specific lengths from s to t. In other words we are applying the FTRS algorithm repeatedly on these subgraphs. The FTRS algorithm has been proven by Baswana et.al. in their paper. To prove that the algorithm is correct we have to show that G_{i+1} that we are constructing from G_i actually gives us the subgraph that we want.

An important observation here would be that we are interested in new ending paths only. New ending paths are paths that end on t using an edge that has not been added to t uptil now.

We are computing K-FT distance preserver with k-FT distance preserver(t) for each t $\in$ V. From the algorithm we can see, in each k-FT distance preserver(t), we add some edges to t, which are the last edges of some edge disjoint paths.The basic idea is suppose there is a path P from s to t after k edge failures, and the the path P uses the edge (u-t) to end at t. This means there is a path P' = P - (u-t) from s to u. Thus the path P will be preserved as a consequence of computing K-FT distance preserver(u) and the edge (u-t).

This shows that suppose paths of greater length which terminate on t using an edge that has been already added need not be considered while computing k-FT distance preserver(t). Using the same example as before, suppose the path P ends on t using the edge (u-t) but it has been added on t before, then this path P need not be considered while computing k-FT distance preserver(t). The path which must be considered are new ending paths.

Lemma 5.2. *The graph G_{i+1} constructed from G_i contains all valid $(i+1)^{th}$ shortest new ending paths.*

Proof. All the graphs until G_i have already handled i^{th} shortest new ending paths. A $(i+1)^{th}$ shortest new ending path is a valid one if it is the shortest path from s to t after some failures and no shorter path is available. Looking at the algorithms for k=1 and k=2 , the cut that is being removed is the Farthest Min Cut of size 1 for k=1 and the Farthest Min Cut of size 2 for k=2.

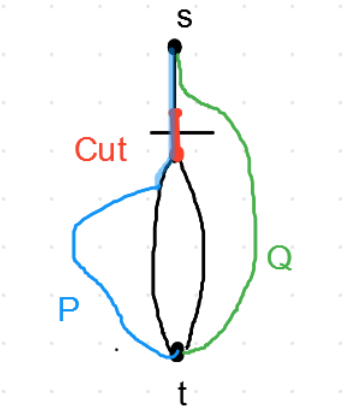

Figure 1: k = 1

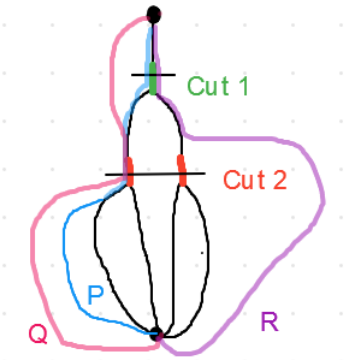

Figure 2: k = 2

In the figure k=1, the cut is removed. Any new ending path of greater length which passes through the cut such as P can't be a valid shortest path after 1 edge failure as two edge disjoint paths are present below the cut. Thus the path avoiding the failure will pass through one of the two edge disjoint paths. However paths such as Q which can possibly be shortest paths avoiding 1 failure and will be considered by our 1 FT distance preserver algorithm.

In the figure k=2, the cut of size 2 or cut 2 is removed. Any new ending path of greater length which passes through the cut 2 can't be a valid shortest path after 2 edge failures.

Consider the case of a valid shortest arbitrary path P. It follows the original shortest path, passes through one edge of cut 2 and then deviates away from the 2 edge disjoint paths below cut 2 to join at t. This will happen only when the 2 edge failures lie on these 2 edge disjoint paths. However in that case, a shorter path passing through the other edge of cut 2 is preserved and thus P is not a valid shortest path.

Consider again the case of path Q. It deviates from the original path once, joins the path again, passes through cut 2 and again deviates below cut 2 to join at t. If it deviated from the original path once above cut 2, means a failure took place on the path above cut 2. Now only one more failure is remaining and 2 edge disjoint paths are present below cut 2, thus the second deviation in path Q is not necessary and Q is not a valid shortest path.

However paths such as R can possibly be shortest paths avoiding 2 failures and will be considered by our 2 FT distance preserver algorithm □

Lemma 5.3 (Remaining paths for k=1)**.** *The paths remaining are those whose lengths are more than d+$\sqrt{n}$. The paths we need to consider are only new ending edge disjoint paths and the number of such long new ending edge disjoint remaining is $\sqrt{n}$*

Proof. Suppose there are two paths P and Q initially edge disjoint which avoid a failure and later combine at a vertex and then reach t. Since both of them are doing the exact same thing keeping only one of them is sufficient. On the other hand, consider two paths P and Q who were same initially but later deviated from each other and met at t.the path P is a shortest path avoiding a failed edge. If Q was following it from the start, deviating from P was not necessary as their is only 1 failure. Thus only new ending edge disjoint paths have to be considered. There could many long paths whose length is more than d+$\sqrt{\text{n}}$. however there are at most n vertices and each edge disjoint path has at least d+$\sqrt{\text{n}}$ vertices in it. Then by simple division we see that there can be at max $\sqrt{\text{n}}$ such paths. □

This was the argument used in Parter and Peleg's paper. A similar argument also exist for the case k=2.[7][6]

6 Summary, Further Work and Conclusion

The two algorithms developed as a result of this SRIP research project are thus able successfully compute k - Fault Tolerant Distance Preservers for k =1 and 2 in a much simpler way than procedures mentioned before without considering too many cases

For k = 1
In each stage we add 2 edges to t and this goes on for $\sqrt{n}$ many times. After this another $\sqrt{n}$ edges are added. Thus in total $3\sqrt{n}$ edges are added to each t. The overall space used by the subgraph is thus $3n\sqrt{n}$ which is $O(n\sqrt{n})$ or $O(n^{3/2})$ which matches with the space bound given by Parter and Peleg.

For k = 2
In each stage we add 4 edges to t and this goes on for $n^{2/3}$ many times. After this another $n^{1/3}$ edges are added. Thus in total 4 $n^{2/3} + n^{1/3}$ edges are added to each t. The overall space used by the subgraph is thus $O(nn^{2/3})$ which is $O(n^{5/3})$ which again matches with the space bound proved by Parter.

Further research would be to extend this simple idea to construct k -FT distance preservers for any k or to extend 1 FT and 2 FT distance preservers to multiple source.

This SRIP project gave me a first hand experience of how academic research takes place and was a great learning experience for me. Other activities, especially the poster presentation allowed me to know more about research in other disciplines as well.

References

[1] S. Baswana, K. Choudhary, and L. Roditty, "Fault tolerant subgraph for single source reachability: Generic and optimal", in *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '16, Cambridge, MA, USA: ACM, 2016, pp. 509–518, ISBN: 978-1-4503-4132-5. DOI: 10.1145/2897518.2897648. [Online]. Available: http://doi.acm.org/10.1145/2897518.2897648.

[2] G. Bodwin, F. Grandoni, M. Parter, and V. V. Williams, "Preserving distances in very faulty graphs", *CoRR*, vol. abs/1703.10293, 2017. arXiv: 1703.10293. [Online]. Available: http://arxiv.org/abs/1703.10293.

[3] D. Chakraborty and D. Das, "Near optimal sized weight tolerant subgraph for single source shortest path", *CoRR*, vol. abs/1707.04867, 2017. arXiv: 1707.04867. [Online]. Available: http://arxiv.org/abs/1707.04867.

[4] M. Gupta and S. Khan, "Multiple source dual fault tolerant BFS trees", *CoRR*, vol. abs/1704.06907, 2017. arXiv: 1704.06907. [Online]. Available: http://arxiv.org/abs/1704.06907.

[5] M. Gupta and A. Singh, "Generic single edge fault tolerant exact distance oracle", *CoRR*, vol. abs/1805.00190, 2018. arXiv: 1805.00190. [Online]. Available: http://arxiv.org/abs/1805.00190.

[6] M. Parter, "Dual failure resilient BFS structure", *CoRR*, vol. abs/1505.00692, 2015. arXiv: 1505.00692. [Online]. Available: http://arxiv.org/abs/1505.00692.

[7] M. Parter and D. Peleg, "Sparse fault-tolerant BFS trees", *CoRR*, vol. abs/1302.5401, 2013. arXiv: 1302.5401. [Online]. Available: http://arxiv.org/abs/1302.5401.