Fault Tolerant Subgraph for Single Source Reachability: Generic and Optimal [*]

Surender Baswana
Department of CSE,
I.I.T. Kanpur, India
sbaswana@cse.iitk.ac.in

Keerti Choudhary
Department of CSE,
I.I.T. Kanpur, India
keerti@cse.iitk.ac.in

Liam Roditty
Department of Comp. Sc.
Bar Ilan University, Israel.
liam.roditty@biu.ac.il

Abstract

Let $G = (V, E)$ be an n-vertices m-edges directed graph. Let $s \in V$ be any designated source vertex. We address the problem of single source reachability (SSR) from s in presence of failures of vertices/edges.

We show that for every $k \geq 1$, there is a subgraph H of G with at most $2^k n$ edges that preserves the reachability from s even after the failure of any k edges. Formally, given a set F of k edges, a vertex $u \in V$ is reachable from s in $G \setminus F$ if and only if u is reachable from s in $H \setminus F$. We call H a k-Fault Tolerant Reachability Subgraph (k-FTRS). We prove also a matching lower bound of $\Omega(2^k n)$ for such subgraphs. Our results extend to vertex failures without any extra overhead.

Our general construction of k-FTRS is interesting from several different perspectives.

Graph theory: It reveals a separation between SSR and single source shortest paths (SSSP) in directed graphs. More specifically, in the case of SSSP in weighted directed graphs, there is a lower bound of $\Omega(m)$ even for a single edge failure [9]. In the case of unweighted graphs there is a lower bound of $\Omega(n^{3/2})$ edges, again, even for a single edge failure [16]. There is also a matching upper bound but *nothing* is known for two or more failures in the directed graphs.

Algorithms: It implies fault tolerant algorithms for other interesting problems, namely, (i) verifying if the strong connectivity of a graph is preserved after k edge or vertex failures, (ii) computing a dominator tree of a graph after k-failures.

Techniques: It makes an interesting usage of the concept of *farthest* min-cut which was already introduced by Ford and Fulkerson [12] in their pioneering work on flows and cuts. We show that there is a close relationship between the farthest min-cut and the k-FTRS. We believe that our new technique is of independent interest.

[*]This research was partially supported by Israel Science Foundation (ISF) and University Grants Commission (UGC) of India. The research of the second author was partially supported by Google India under the Google India PhD Fellowship Award.

1 Introduction

Networks in most real life applications are prone to failures. Such a network can be modeled as a graph where vertices (or edges) may change their status from active to failed, and vice versa. These failures, though unpredictable, are small in numbers and are transient due to some simultaneous repair process that is undertaken in these applications. This aspect can be captured by associating a parameter k with the network such that there are at most k vertices (or edges) that are failed at any stage, where k is much smaller than the number of vertices in the underlying graph. This motivates the research on designing fault tolerant structures for various graph problems.

In this paper we address the problem of single source fault tolerant reachability in directed graphs. The objective is to construct a sparse subgraph that preserves the reachability from a given fixed source s even after k failures. The following definition characterizes this subgraph precisely.

Definition 1.1 (k-FTRS) *Let $G = (V, E)$ be a directed graph and $s \in V$ be a designated source vertex. A subgraph H of G is said to be a k-Fault Tolerant Reachability Subgraph (k-FTRS) for G if for any subset $F \subseteq E$ of at most k edges, a vertex $v \in V$ is reachable from s in $G \setminus F$ if and only if v is reachable from s in $H \setminus F$.*

Existence of a sparse k-FTRS and its efficient construction is important from the perspective of theoretical as well as applied computer science since single source reachability is a fundamental problem. Moreover, reachability lies at the core of many other graph problems like strong-connectedness, dominators [14], double dominators [18], etc. So obtaining a sparse k-FTRS may help in obtaining fault tolerant solution for these problems as well.

Surprisingly, the only previously known result for k-FTRS was for $k = 1$. The special case of 1-FTRS is closely related to the well known concept of *dominators* presented in the seminal work of Lengauer and Tarjan [14]. Given a DFS tree T rooted at s, it is straightforward using the ideas of [14] to compute a 1-FTRS with at most $2n$ edges that contains T. In [2] we show that even if T is any arbitrary reachability tree, we can efficiently compute a 1-FTRS with at most $2n$ edges that contains T.

In this paper we present efficient construction of a sparse k-FTRS for any $k > 1$. We prove the following Theorem.

Theorem 1.1 *There exists an $O(2^k mn)$ time algorithm that for any given integer $k \geq 1$, and any given directed graph G on n vertices, m edges and a designated source vertex s, computes a k-FTRS for G with at most $2^k n$ edges. Moreover, the in-degree of each vertex in this k-FTRS is bounded by 2^k.*

We also show that the $2^k n$ bound on the size of k-FTRS is tight by proving the following theorem.

Theorem 1.2 *For any positive integers n, k with $n \geq 2^k$, there exists a directed graph on n vertices whose k-FTRS must have $\Omega(2^k n)$ edges.*

The above theorems also hold in the case when the k-FTRS is defined with respect to vertex failures instead of edge failures.

Our result on k-FTRS implies solutions to the following problems in a straightforward manner.

1. ***Strong connectedness of a graph.*** We show that it is possible to preprocess G in polynomial time to build an $O(2^k n)$ size data structure that, after the failure of any set F of k edges or vertices, can determine in $O(2^k n)$ time if the strongly connected components of graph $G \setminus F$ are the same as that of graph G.

2. ***Fault tolerant dominator tree.*** We show that any given directed graph $G = (V, E)$ with a source $s \in V$ can be preprocessed in polynomial time to build an $O(2^k n)$ size data structure that, after the failure of any set F of k edges or vertices, can report the dominator tree of $G \setminus F$ in $O(2^k n)$ time.

Besides the above applications, our techniques reveal an interesting connection between two seemingly unrelated structures: farthest min-cut and k-FTRS. The *farthest* min-cut is a very basic concept in the area of flows and cuts, and was already introduced by Ford and Fulkerson [12]. It turns out that the construction of k-FTRS employs a hierarchy of suitably constructed farthest min-cuts. We believe that this relationship might be of independent interest from the perspectives of graph theory and algorithm design.

1.1 Related Work

The most closely related problem to single source reachability is the problem of single source shortest paths. Similar to the definition of k-FTRS, one can define a k-Fault Tolerant Subgraph that preserves the shortest path tree rooted at vertex s. We denote such a subgraph with k-FTSS. Unfortunately, for weighted directed graphs, no sparse k-FTSS is possible - Demetrescu et. al [9] showed that there exist graphs whose 1-FTSS must have $\Omega(m)$ edges. For the case of unweighted graphs Parter and Peleg [16] showed that we can compute a 1-FTSS with $O(n^{3/2})$ edges; they also showed a matching lower bound. Recently, Parter [15] extended this result to 2-FTSS with $O(n^{5/3})$ edges for undirected graphs. She also showed a lower bound of $\Omega(n^{5/3})$. While the construction of a 1-FTSS is relatively simple, the construction of 2-FTSS is relatively complicated. Nothing is known for k-FTSS for the case of $k > 2$.

As opposed to the situation with FTSS, our construction of generic FTRS implies that for every constant k there is a linear size k-FTRS. Therefore, from the perspective of graph theory, our tight k-FTRS reveals an interesting separation phenomena between single source reachability and single source shortest paths in the directed graphs.

Though there are discouraging results for sparse FTSS for directed graphs, there exist sparse fault tolerant subgraphs for undirected graphs that preserve distance from the source approximately. Baswana and Khanna [3] showed that there is a subgraph with $O(n \log n)$ edges that preserves the distances from s up to a multiplicative stretch of 3 upon failure of any single vertex. Parter and Peleg [17] improved this result and showed that such a subgraph is possible even with at most $3n$ edges. For the case of edge failures, sparse fault tolerant subgraphs exist for general k. Bilò et. al [4] showed that we can compute a subgraph with $O(kn)$ edges that preserves distances from s up to a multiplicative stretch of $(2k + 1)$ upon failure of any k edges. They also showed that we can compute a data structure of $O(kn \log^2 n)$ size that is able to report the $(2k + 1)$-stretched distance from s in $O(k^2 \log^2 n)$ time.

For the case of all-pair shortest paths (APSP), Demetrescu et. al [9] showed that we can build an $O(n^2 \log n)$ size data structure that can report the distance from u to v avoiding x for any $u, v, x \in V$ in $O(1)$ time. Duan and Pettie [11] extended this result to dual failures by designing a data structure of $O(n^2 \log^3 n)$ space that can answer any distance query upon the failure of any two vertices in $O(\log n)$ time.

Fault tolerant structures for depth first search tree, graph spanners, approximate distance oracles, and compact routing schemes have been studied in [1, 6, 7, 8, 10].

1.2 Organization of the paper

We describe notations and terminologies in Section 2. In Section 3 we provide an overview of our result, and in Section 4 we describe various tools used to obtain a k-FTRS. In order to describe the ideas underlying k-FTRS gradually for a better understanding, we first present a simple construction of a 2-FTRS with $4n$ edges in Section 5. We describe the generic construction of k-FTRS for any $k \geq 1$ in Section 6. The proof for the matching lower bound is given in Section 7. We present the applications of k-FTRS in Section 8.

2 Preliminaries

Given a directed graph $G = (V, E)$ on $n = |V|$ vertices and $m = |E|$ edges, and a source vertex $s \in V$, the following notations will be used throughout the paper.

- $E(f)$: Edges of the graph G carrying a non-zero flow for a given flow f.
- $E(P)$: Edges lying on a path P.
- $E(A)$: Edges of the graph G whose both endpoints lie in set A, where $A \subseteq V$.
- $G(A)$: The subgraph of G induced by the vertices lying in a subset A of V.
- $H + (u, v)$: The graph obtained by adding an edge (u, v) to graph H.
- $H \setminus F$: The graph obtained by deleting the edges lying in a set F from graph H.
- $\textsc{max-flow}(H, S, t)$: The value of the maximum flow in graph H, where the source is a set of vertices S and destination is a single vertex t.
- $\textsc{out}(A)$: The set of all those vertices in $V \setminus A$ having an incoming edge from some vertex of A, where $A \subseteq V$.
- $\textsc{In-Edges}(v, H)$: The set of all incoming edges to v in H, when the graph H is clear from context we denote it by simply $\textsc{In-Edges}(v)$.
- $P[a, b]$: The subpath of path P lying between vertices a and b, assuming a precedes b on P.
- $P :: Q$: The path formed by concatenating paths P and Q in G. Here it is assumed that the last vertex of P is the same as the first vertex of Q.

Our algorithm for computing a k-FTRS will involve the concepts of max-flow, min-cut, and edge disjoint paths. So, at times, we will visualize the same graph G as a network with unit edge capacities. The following well known result from Graph theory shows the connection between max-flow and edge-disjoint paths.

Theorem 2.1 *For any positive integer α, there is a flow from a source set S to a destination vertex t of value α if and only if there are α edge disjoint paths originating from set S and terminating at t.*

We now give definition for (S, t)-min-cut.

Definition 2.1 *A set of edges $C \subset E$ is said to be an (S, t)-cut if each path from any $s \in S$ to t must pass through at least one edge from C. The size of a cut C is the number of edges present in C. An (S, t)-cut of smallest size is called (S, t)-min-cut.*

The following definition introduces the notion of FTRS from perspective of a single vertex $v \in V$.

Definition 2.2 *Given a vertex $v \in V$ and an integer $k \geq 1$, a subgraph $G' = (V, E')$, $E' \subseteq E$ is said to be k-FTRS(v) if for any set F of k edge failures, the following condition holds: v is reachable from s in $G \setminus F$ if and only if v is reachable from s in $G' \setminus F$.*

This definition allows us to define k-FTRS in an alternative way as follows.

Definition 2.3 *A subgraph H of G is a k-FTRS if and only if H is k-FTRS(v) for each v in G.*

3 An overview

Our starting point is a lemma (referred as Locality Lemma) that allows us to focus on a single vertex for computing k-FTRS. It essentially states that if there exists an algorithm that for any vertex v computes a k-FTRS(v) in which in-degree of v is bounded by small constant c, then on applying this algorithm recursively on an arbitrary sequence of vertices of G, we can get a k-FTRS for G in which in-degree of all the vertices is bounded by same constant c.

Thus the problem reduces to computing a k-FTRS(t) with at most 2^k incoming edges for any $t \in V$. The construction of a k-FTRS(t) employs farthest min-cut. Recall that a (s,t)-cut C partitions the vertices into two sets: one containing the source, and the other containing the sink. The farthest min-cut is the (unique) min-cut for which the set containing the source is of largest size. We now provide the main idea underlying the construction of a k-FTRS(t).

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. In order to convey the importance of farthest min-cut, let us consider the case when max-flow is exactly k. Let us suppose that there exists a path P from s to t in $G \setminus F$, 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. It follows from the properties of the farthest min-cut that if we include vertex b_i in the source then the max-flow increases (see Lemma 4.2). That is, we get at least $k+1$ edge disjoint paths from the set $\{s, b_i\}$ to t. Note that one of these paths, say Q, must be intact even after k failures. Though Q may start from b_i (instead of s), but it is not problematic as the concatenation $P[s, b_i] :: Q$ will be preserved in $G \setminus F$. 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) max-flow 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 b_i's of the edges corresponding to the farthest (S,t) min-cut. The size of the cuts in this hierarchy governs the in-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 $O(m)$ vertices and edges, so that the following assumption holds.

Assumption 1: The out-degree of all vertices in G is at most two.

It turns out that a k-FTRS(t) for the original graph can be easily obtained by a k-FTRS(t) of the transformed graph. We provide the justification for the above assumption in the appendix.

In this paper, we describe the construction of a k-FTRS with respect to edge failures only. Vertex failures can be handled by simply splitting a vertex v into an edge (v_{in}, v_{out}), where the incoming and outgoing edges of v are respectively directed into v_{in} and directed out of v_{out}.

4 The main tools

We now describe the main tools used in obtaining our k-FTRS.

4.1 Locality Lemma

We first formally state and prove the locality lemma.

Lemma 4.1 *Suppose there exists an algorithm $\mathcal{A}$ and an integer c_k satisfying the following condition: Given any graph G and a vertex v in G, $\mathcal{A}$ can compute a subgraph H of G such that*

(i) *H is a k-FTRS(v), and*

(ii) *in-degree of v in H is bounded by c_k.*

Then, we can compute a k-FTRS for G with at most $c_k \cdot n$ edges.

Proof: Let $\langle v_1, \ldots, v_n \rangle$ be any arbitrary sequence of the n vertices of G. We compute k-FTRS in n rounds as follows. Let $G_0 = G$ be the initial graph. In round i, we compute a graph G_i which is a k-FTRS with in-degree of vertices $v_1, ..., v_i$ bounded by c_k. This is done as follows - (i) We compute a k-FTRS(v_i), say H, for graph G_{i-1} using algorithm $\mathcal{A}$; (ii) We set G_i to be the graph obtained from G_{i-1} by restricting the incoming edges of v_i to only those present in H. It is easy to see that in-degree of vertices $v_1, ..., v_i$ in graph G_i would be bounded by c_k. We now show using induction that the graphs $G_0, G_1, .., G_n$ are all k-FTRS for G. The base case trivially holds true. In order to show that G_i ($i > 0$) is a k-FTRS for G, it suffices to show that G_i is a k-FTRS for G_{i-1}.

Consider any set F of k edge failures in G_{i-1}. Let x be any vertex reachable from s in $G_{i-1} \setminus F$ by some path, say P. We need to show the existence of a path Q from s to x in $G_i \setminus F$. If path P does not pass through v_i then we can simply set Q as P. If P passes through v_i, then we consider the segments $P[s, v_i]$ and $P[v_i, x]$. Since G_i and G_{i-1} may differ only in incoming edges of v_i, path $P[v_i, x]$ must be intact in $G_i \setminus F$. Now H is a k-FTRS(v_i) for G_{i-1}, thus there must exist a path, say Q, from s to v_i in $H \setminus F$. Note that G_i contains H. Thus $Q :: P[v_i, x]$ is a walk from s to x in $G_i \setminus F$, and after removal of all loops from it, we get a path from s to x in $G_i \setminus F$. Hence G_i is a k-FTRS for G_{i-1}. $\square$

4.2 Farthest Min-Cut

Definition 4.1 *Let S be a source set and t be a destination vertex. Any (S, t)-min-cut C partitions the vertex set into two sets: $A(C)$ containing S, and $B(C)$ containing t. An (S, t)-min-cut C^* is said to be the* farthest min-cut *if $A(C^*) \supsetneq A(C)$ for any (S, t)-min-cut C other than C^*. We denote C^* with* $\text{FMC}(G, S, t)$.

Ford and Fullkerson [12] gave an algorithm for constructing the farthest (S, t)-min-cut and also established its uniqueness. For the sake of completeness we state the following result from [12].

Lemma 4.2 *Let G_f be the residual graph corresponding to any max-flow f_S from S to t. Let B be the set of those vertices from which there is a path to t in G_f, and $A = V \setminus B$. Then the set C of edges originating from A and terminating to B is the unique farthest (S, t)-min-cut, and is independent of the choice of the initial max-flow f_S.*

We now state an important property of the farthest min-cut. Informally, this property claims that the max-flow in G increases by one if we add to G a new edge from the set $S \times B$. (If the new edge already exists in E, then we add one more copy of it to G).

Lemma 4.3 *Let S be a source set, t be a destination vertex, C be* $\text{FMC}(G, S, t)$*, and (A, B) be the partition of V corresponding to cut C. Let $(s, w) \in (S \times B)$ be any arbitrary edge, and $G' = G + (s, w)$ be a new graph. Then,* $\text{MAX-FLOW}(G', S, t) = 1 + \text{MAX-FLOW}(G, S, t)$*, and $C' = C \cup \{(s, w)\}$ forms a (S, t)-min-cut for graph G'.*

Proof: Let f_S be a max-flow from S to t, and G_f be the corresponding residual graph. Since $w \in B$, Lemma 4.2 implies that there exists a path from w to t in G_f. This shows that there exists a path from s to t in $G_f + (s, w)$. Note that $G_f + (s, w)$ is the residual graph for G' with respect to flow f_S. Thus $\text{MAX-FLOW}(G', S, t)$ is greater than $\text{MAX-FLOW}(G, S, t)$. Since G' is obtained by adding only one extra edge to G, the value of max-flow cannot increase by more than one, hence we get that $\text{MAX-FLOW}(G', S, t)$ is equal to $1 + \text{MAX-FLOW}(G, S, t)$.

To prove the second part, note that the existence of a path P from S to t in $G' \setminus C'$ would imply the existence of a path from S to t in G not passing through cut C. Since this cannot be possible, C' must be an (S, t)-cut for graph G'. Now $|C'| = 1 + |C| = 1 + \text{MAX-FLOW}(G, S, t) = \text{MAX-FLOW}(G', S, t)$. That is, the cardinality of C' is the same as the value of max-flow from S to t. Hence, C' is an (S,t)-min-cut. $\square$

We state two more properties of farthest-min-cut that will be used in the construction of k-FTRS .

Lemma 4.4 *Let s and t be a pair of vertices. Let $S \subseteq V$ such that $s \in S$ and $t \notin S$. Let f_S be a max-flow from S to t, C be* FMC(S,t)*, and (A,B) be the partition of V induced by cut C. Then we can find a max-flow, say f, from s to t such that $E(f) \subseteq E(A) \cup E(f_S)$.*

Proof: Let α be equal to $\textsc{max-flow}(G,S,t)$, and β be equal to $\textsc{max-flow}(G,s,t)$. Let $C = \{e_1,..,e_\alpha\}$. Let f' be any arbitrary max-flow from s to t. Note that C is also an (s,t)-cut. Thus, without loss of generality we can assume that $C \cap E(f') = \{e_1,..,e_\beta\}$. Now let $\{(P_i :: e_i :: P'_i) : i \le \alpha\}$ be a set of α edge disjoint paths from S to t corresponding to max-flow f_S. Since the edges of cut C are fully saturated with respect to flow f_S, each P_i will lie entirely in $G(A)$ and each P'_i will lie entirely in $G(B)$.

Let $\{(Q_i :: e_i :: Q'_i) : i \le \beta\}$ be set of β edge disjoint paths from s to t corresponding to flow f' such that each Q_i lies entirely in $G(A)$. Note that since C is not necessarily an (s,t)-min-cut, so a path Q'_i may pass multiple times through cut C. Now $\{(Q_i :: e_i :: P'_i) : i \le \beta\}$ forms β edge disjoint paths from s to t. This is so because Q_i's lie entirely in $G(A)$ and P'_i's lie entirely in set $G(B)$. Let f be the flow corresponding to these paths. Then f gives a max-flow from s to t such that $E(f) \subseteq E(A) \cup E(f_S)$. $\square$

Lemma 4.5 *Let s and t be a pair of vertices. Let $S \subseteq V$ such that $s \in S$ and $t \notin S$. Let $(\hat{A},\hat{B})$ be the partition of V induced by* FMC(s,t)*, and (A,B) be the partition of V induced by* FMC(S,t)*. Then $B \subseteq \hat{B}$.*

Proof: Consider any vertex $x \in B$. We first show that we can find a max-flow (say f_S) from S to t, and path (say P) from x to t, such that $E(f_S) \cap E(P)$ is empty. Consider the graph $G' = G + (s,x)$. From Lemma 4.3, we have that $\textsc{max-flow}(G',S,t) = 1 + \textsc{max-flow}(G,S,t) = 1 + \alpha$ (say). Consider any max-flow f'_S from S to t in G'. Notice that $E(f'_S)$ must contain the edge (s,x). On removal of this edge from f'_S, it decomposes into a flow f_S (from S to t in G of capacity α), and a path P (from x and to t in G). It is easy to verify that f_S is a max-flow in G, and $E(f_S) \cap E(P) = \emptyset$.

Now from Lemma 4.4 we have that there exists a max-flow, say f, from s to t such that $E(f) \subseteq E(A) \cup E(f_S)$. Also note that path P will lie entirely in graph $G(B)$, since edges of cut (A,B) are fully saturated by flow f_S, so if path P enters set A, it will not be able to return to vertex $t \in B$. Thus $E(f) \cap E(P)$ is also empty. Thus path P lies in residual graph corresponding to max-flow f from s to t in G. So vertex x must lie in $\hat{B}$. Hence we have $B \subseteq \hat{B}$. $\square$

5 A 2-FTRS with $4n$ edges

From Lemma 4.1 it follows that in order to construct a 2-FTRS for a vertex s of graph G, it is sufficient to construct for an arbitrary vertex t a subgraph H which is a 2-FTRS(t), such that the in-degree of t in H is at most 4.

By Assumption 1 stated in Section 3, we have that out-degree of s is at most 2. Thus the value of max-flow from s to t must be either one or two. Below we explain how to construct a 2-FTRS(t) in each of these cases.

(i) $\textsc{max-flow}(G,s,t) = 2$: Let $C = \{(a,b),(a',b')\}$ be the farthest (s,t) min-cut in G. (See Figure 1 (i)). So after the failure of a set F of any two edges, a path from s to t (if it exists) in $G \setminus F$, must pass through C. We construct an auxiliary graph H by adding the edge (s,b) or (s,b') to G depending upon whether this

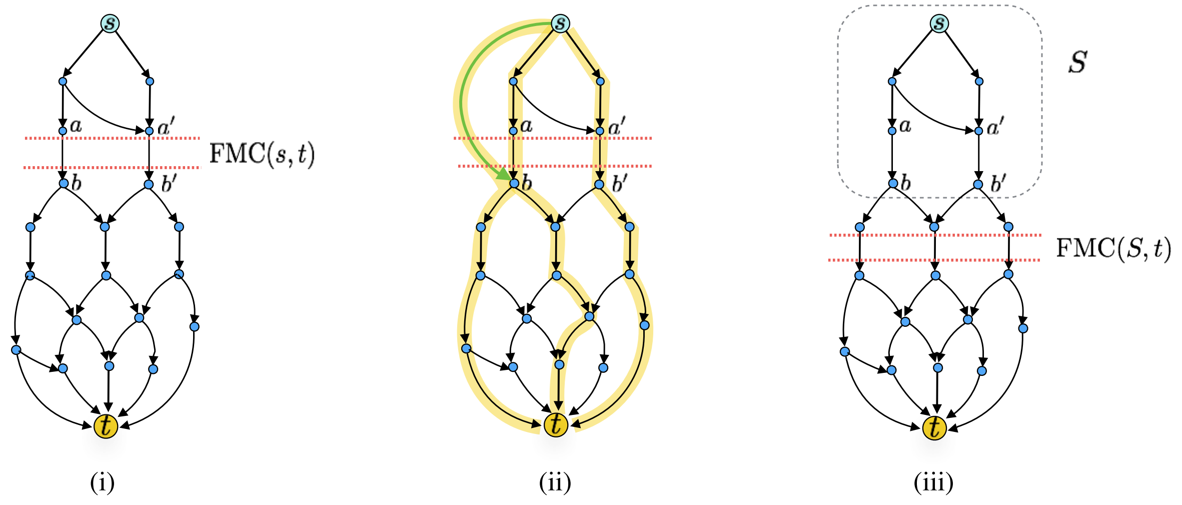

Figure 1: (i) Edges (a, b) and (a', b') represent the farthest min-cut from s to t. (ii) Paths highlighted in yellow color represent three edge-disjoint paths P_0, P_1, P_2 in graph $G + (s, b)$. (iii) As $b, b' \neq t$, source S is equal to $A \cup \{b, b'\}$, and $\text{FMC}(G, S, t) = 3$.

path passes through (a, b) or (a', b'). Since C is the farthest min-cut of G, it follows from Lemma 4.3 that the value of (s, t) max-flow in both the graphs $G + (s, b)$ and $G + (s, b')$ must be 3. So we denote P_0, P_1, P_2 to be any three edge disjoint paths from s to t in $G + (s, b)$ (see Figure 1 (ii)), and similarly P'_0, P'_1, P'_2 to be three edge disjoint paths in $G + (s, b')$. The lemma below shows how these paths can be used to compute a 2-FTRS(t).

Lemma 5.1 *Let E_t be a subset of incoming edges to t satisfying the following conditions:*

- *E_t contains last edges on paths P_0, P_1, P_2 in case $b \neq t$, and the edge (a, b) in case $b = t$.*
- *E_t contains last edges on paths P'_0, P'_1, P'_2 in case $b' \neq t$, and the edge (a', b') in case $b' = t$.*

Then the graph G^* formed by restricting the incoming edges of t in G to E_t is a 2-FTRS(t).

Proof: Consider any set F of two edge failures. Note that if t is unreachable from s in $G \setminus F$, then we have nothing to prove. So let us assume that there exists a path R from s to t in $G \setminus F$, and it passes through edge (a, b) of cut C. So, the auxiliary graph is $H = G + (s, b)$. Now if $b = t$, then R is in $G^* \setminus F$ since $(a, b) = (a, t) \in E_t$. Consider the case that $b \neq t$. Since P_0, P_1, and P_2 are edge disjoint, at least one of them is in the graph $H \setminus F$, let this path be P_0. Now either (i) P_0 is in $G \setminus F$, or (ii) the first edge on it must be (s, b). In the latter case we replace the edge (s, b) by path $R[s, b]$, so that the path $R[s, b]::P_0[b, t]$ is in $G \setminus F$. In both cases we get a path from s to t in $G \setminus F$ that enters t using only edges of the set E_t. Similar analysis can be carried out when path R passes through edge (a', b'). Hence we get that G^* is a 2-FTRS(t). $\square$

Using the above lemma we can get a 2-FTRS(t) in which the in-degree of t is bounded by 6, by including the last edges of all six paths $P_0, P_1, P_2, P'_0, P'_1, P'_2$ in E_t. But our aim is to achieve a bound of 4. For this we construct a source set $S = A \cup (\{b, b'\} \setminus \{t\})$, where A and B forms a partition of V induced by C, and compute a max flow f_S from S to t. (See Figure 1 (iii)). The following lemma shows that the graph obtained by restricting the incoming edges of t to those carrying a non-zero flow with respect to f_S is a 2-FTRS(t).

Lemma 5.2 *Let $\mathcal{E}(t)$ be the set of incoming edges to t carrying a non-zero flow with respect to f_S. Then the graph $G^* = \big(G \setminus \textsc{In-Edges}(t)\big) + \mathcal{E}(t)$ is a 2-FTRS(t).*

Proof: In order to prove this lemma it suffices to show that there exist paths $P_0, P_1, P_2, \dots, P_2'$ such that set $\mathcal{E}(t)$ satisfies the conditions required in Lemma 5.1. Here we show the existence of P_0, P_1, P_2. The proof for the existence of paths P_0', P_1', P_2' follows in a similar manner.

Note that if $b = t$, then edge (a, b) will be a direct edge from S to t, and would thus be in f_S. In this case (a, b) is in $\mathcal{E}(t)$. So consider the case $b \neq t$. In order to compute the paths P_0, P_1, P_2 we consider the graph $G_b = G + (s, b)$. Since both the endpoints of edge (s, b) lie in S, we have that f_S is a max-flow from S to t in graph G_b, as well. Let (A_S, B_S) be the partition of V induced by any (S, t) min-cut in graph G_b. Lemma 4.4 implies that we can find a max-flow, say f, from s to t in G_b such that $E(f) \subseteq E(A_S) \cup E(f_S)$. In other words, the incoming edges to t in $E(f)$ are from the set $\mathcal{E}(t)$. Recall that Lemma 4.3 implies that value of flow f is 3. So we set P_0, P_1, P_2 to be just the three paths corresponding to flow f. $\square$

We now show that the in-degree of t in G^* is bounded by 4. In order to prove this, it suffices to show that the value of (S, t) max-flow in G is at most 4. Now if

1. $b, b' \neq t$, then outgoing edges of b and b' will form an (S, t) cut,
2. $b = t$, then (a, b) along with outgoing edges of b' will form an (S, t) cut, and
3. $b' = t$, then (a', b') along with outgoing edges of b will form an (S, t) cut.

By Assumption 1, the out-degree of every vertex is bounded by two. Therefore, the value of (S, t) min-cut (and max-flow) can be at most 4.

(ii) MAX-FLOW$(G, s, t) = 1$: Let $C = \{(x, y)\}$ be the farthest min-cut from s to t. Then every path from s to t must pass through edge (x, y). Note that if $y = t$, then we can simply return the graph obtained by deleting all incoming edges of t except (x, t). If $y \neq t$, then the value of (y, t)-max-flow must be 2. So in this case we return a 2-FTRS(t) with y as a source (using **Case i.**), let this graph be G^*. It is easy to verify that G^* is a 2-FTRS(t) with s as source, and in-degree of t in G^* is bounded by 4.

This completes the construction of a 2-FTRS(t). So we have the following theorem.

Theorem 5.1 *Given any directed graph G on n vertices, we can computes a 2-FTRS for it with at most $4n$ edges.*

6 A k-FTRS with $2^k n$ edges

In this section we prove Theorem 1.1. Let t be a vertex in G for which k-FTRS(t) from s needs to be computed. (Recall that from Lemma 4.1 it follows that this is sufficient in order to prove Theorem 1.1.) Before we present the construction of k-FTRS(t) we state one more assumption on the graph G (in addition to Assumption 1).

Assumption 2: The out-degree of the source vertex s is 1.

This assumption can be easily justified by adding a new vertex s' together with an edge (s', s) to G and then setting s' as the new source vertex. Algorithm 1 constructs a k-FTRS(t) with at most 2^k incoming edges of t.

Algorithm 1 works as follows. It performs k iterations. In the ith iteration it computes the farthest min-cut C_i between a source set S_i and vertex t. For the 1st iteration, the source set S_1 consists of only vertex s. For $i \geq 1$, the source set S_{i+1} is defined by the farthest min-cut computed in the ith iteration. If (A_i, B_i) is the partition of V induced by C_i, then we set S_{i+1} as A_i union those vertices in $B_i \setminus \{t\}$ that have an incoming edge from any vertex of A_i. Notice that since $S_i \subseteq A_i$ it implies that $S_i \subseteq S_{i+1}$. After k iterations the algorithm computes a max-flow f_0 from S_{k+1} to t and sets $\mathcal{E}(t)$ to be the incoming edges of t that are in $E(f_0)$. The algorithm returns a graph G^* obtained by restricting the incoming edges of t to $\mathcal{E}(t)$.

Algorithm 1: Algorithm for computing k-FTRS(t)

```
1 $S_1 \leftarrow \{s\}$;
2 for $i = 1$ to $k$ do
3     $C_i \leftarrow$ FMC$(G, S_i, t)$;
4     $(A_i, B_i) \leftarrow$ Partition$(C_i)$;
5     $S_{i+1} \leftarrow (A_i \cup \text{OUT}(A_i)) \setminus \{t\}$;
6 end
7 $f_0 \leftarrow$ max-flow from $S_{k+1}$ to $t$;
8 $\mathcal{E}(t) \leftarrow$ Incoming edges of $t$ present in $E(f_0)$;
9 Return $G^* = (G \setminus \text{IN-EDGES}(t)) + \mathcal{E}(t)$;
```

We show in the next subsection that G^* is a k-FTRS(t) with in-degree(t) bounded by 2^k. We must note that after some iteration $i < k$, the source set may become $V \setminus \{t\}$. In this case, all subsequent iterations will be redundant.

For a better understanding of the hierarchy of cuts and the source sets constructed in our algorithm, refer to Figure 2 (i). Note that some of the edges in a cut C_i ($i \in [1, k]$) may terminate to t, we denote this set by E_i^t. The following lemma shows that these edges are always included in our FTRS G^*.

Lemma 6.1 *For every* $1 \leq i \leq k$, $E_i^t \subseteq \mathcal{E}(t)$.

Proof: Consider any edge $(u, t) \in E_i^t$. Since u belongs to A_i it follows from the algorithm that u is added to the source set $S_{i+1} \subseteq S_{k+1}$. Thus (u, t) is a direct edge from source S_{k+1} to t, and must be in every max-flow from S_{k+1} to t. $\square$

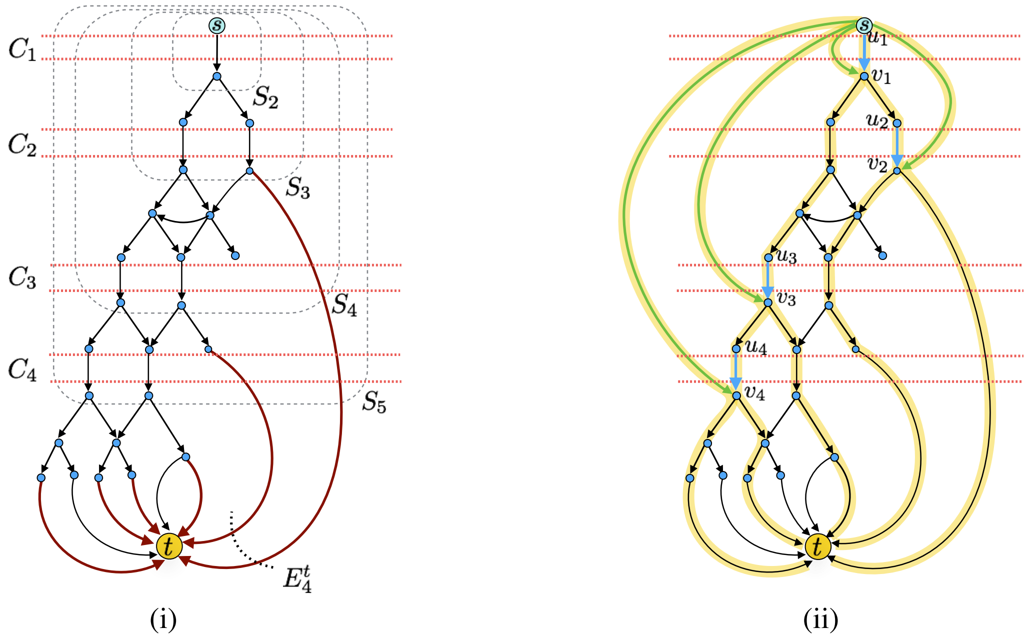

Figure 2: (i) The edges in brown color constitute the set $\mathcal{E}(t)$ when $k = 4$. (ii) Paths highlighted in yellow color represent 5 edge-disjoint paths in graph H.

6.1 Analysis

We now show that G^* is a k-FTRS(t). Let F be any set of k failed edges. Assume that there exists a path R from s to t in $G \setminus F$. We shall prove the existence of a path $\hat{R}$ from s to t in $G^* \setminus F$. Let $i \in [1, k]$. Since each cut C_i is an (s,t)-cut, the path R must pass through an edge, say (u_i, v_i), in C_i. Let us first consider the case when $(u_i, v_i) \in E_i^t$ for some $i \in [1, k]$. Since Lemma 6.1 implies that (u_i, v_i) is present in G^* and $v_i = t$, the path R is contained in G^*. So we can set $\hat{R}$ to R. We now turn to the case when the edge (u_i, v_i) belongs to the set $C_i \setminus E_i^t$ for every $i \in [1, k]$. In order to show the existence of a path $\hat{R}$ in G^*, we introduce a sequence of auxiliary graphs H_i's (for every $i \in [1, k+1]$) as follows.

$$H_1 = G, \quad H_i = G + (s, v_1) + \ldots + (s, v_{i-1}), \quad i \in [2, k+1]$$

Let $H = H_{k+1}$ be the graph obtained by adding all the edges $(s, v_1), ..., (s, v_k)$ to graph G. (See Figure 2 (ii)). We will show using induction that H_i contains exactly i edge disjoint paths from s to t. Before presenting the proof, we show that for any i, the cut C_i, which is the farthest min-cut between S_i and t in G, is also the farthest min-cut between S_i and t in H_i.

Lemma 6.2 $C_i = \text{FMC}(H_i, S_i, t)$

Proof: The cut C_i is defined as $\text{FMC}(G, S_i, t)$ and $H_i = G + \sum_{j<i}(s, v_j)$. Since the algorithm adds to the source set S_{j+1} all the vertices in $\text{OUT}(A_j) \setminus \{t\}$, the vertex $v_j \in \text{OUT}(A_j) \setminus \{t\}$ is added to $S_{j+1} \subseteq S_i$. This implies that the graph H_i is formed by adding edges such that both of their endpoints are in S_i. Hence, C_i is also equal to $\text{FMC}(H_i, S_i, t)$, and A_i, B_i form the partition of V induced by C_i in H_i. □

Lemma 6.3 $\text{MAX-FLOW}(H, s, t) = k + 1$

Proof: We show by induction that $\text{MAX-FLOW}(H_i, s, t) = i$, for any $i \in [1, k+1]$. Note that $H_1 = G$. The base case holds since the out-degree of s is one and t is reachable from s, thus $\text{MAX-FLOW}(H_1, s, t) = 1$.

Recall that H_{i+1} is formed by adding the edge (s, v_i) to H_i, where (u_i, v_i) was an edge present in cut C_i. It follows from Lemma 6.2 that C_i is the farthest min-cut in H_i with S_i as source, and (A_i, B_i) is the partition of V induced by C_i. Let $(\hat{A}, \hat{B})$ be the partition of V induced by $\text{FMC}(H_i, s, t)$. It follows from Lemma 4.5 that $B_i \subseteq \hat{B}$. Therefore, using Lemma 4.3 we get $\text{MAX-FLOW}(H_{i+1}, s, t) = 1 + \text{MAX-FLOW}(H_i, s, t) = i + 1$. □

Next, we define a graph H^* to be the graph H where the incoming edges of t are only those present in the set $\mathcal{E}(t)$, that is, $H^* = (H \setminus \text{IN-EDGES}(t)) + \mathcal{E}(t)$. The following Lemma shows that the value of max-flow remains unaffected by restricting the incoming edges of t to $\mathcal{E}(t)$.

Lemma 6.4 $\text{MAX-FLOW}(H^*, s, t) = k + 1$.

Proof: Recall that f_0 is a max-flow from S_{k+1} to t in G. Since both endpoints of the edges $(s, v_1), \ldots, (s, v_k)$ are in S_{k+1}, we get that f_0 is a max-flow from S_{k+1} to t in graph $H = G + \sum_{i=1}^{k}(s, v_i)$ as well. From Lemma 4.4 it follows that we can always find an (s,t) max-flow, say f, in H such that $E(f) \subseteq E(f_0) \cup E(A_{k+1})$. The flow f terminates at t using only edges from $\mathcal{E}(t)$, therefore, it is a flow in graph $H^* = (H \setminus \text{IN-EDGES}(t)) + \mathcal{E}(t)$ as well. Since f is an (s,t) max-flow in H and max-flow cannot increase on edge removal, it follows from Lemma 6.3 that $\text{MAX-FLOW}(H^*, s, t) = k + 1$. □

Note that graph H^* defined above is also equal to $G^* + \sum_{j=1}^{k}(s, v_j)$. Next, using the $k+1$ edge disjoint paths in H^* we show that G^* is a k-FTRS(t).

Lemma 6.5 *For any set F of k edges, if t is reachable from s in $G \setminus F$, then t is reachable from s in $G^* \setminus F$ as well.*

Proof: Recall that we started with assuming that R is a path from s to t in $G \setminus F$. We need to show that there exists a path $\hat{R}$ from s to t in $G^* \setminus F$. Consider the graph H^*. By Lemma 6.4 we get that there exists $k+1$ edge disjoint paths from s to t in H^*, let these be $P_0, P_1, \ldots, P_k$. Since $|F| = k$, we have that at least one of these $k+1$ paths, say P_0, must be intact in $H^* \setminus F$. Now if P_0 lies entirely in $G^* \setminus F$, then we can set $\hat{R}$ to be P_0. Thus let us assume that P_0 does not lie in $G^* \setminus F$. Since H^* is formed by adding edges $(s, v_1), .., (s, v_k)$ to G^*, we will have that the first edge on P_0 is one of the newly added edges, say (s, v_j), and the remaining path $P_0[v_j, t]$ will lie entirely in $G^* \setminus F$. Now we can simply replace edge (s, v_j) by path $R[s, v_j]$ to get path $\hat{R} = R[s, v_j]{::}P_0[v_j, t]$ from s to t in $G^* \setminus F$. $\square$

We shall now establish a bound on the number of incoming edges of t in G^*.

Bounding the size of set $\mathcal{E}(t)$.

Let $C_{k+1} = \text{FMC}(G, S_{k+1}, t)$. We now prove using induction that $|C_i|$ is bounded by 2^{i-1}, where $i \in [1, k]$, thus achieving a bound of 2^k on $|\mathcal{E}(t)| = |C_{k+1}|$. For the base case of $i = 1$, $|C_1| = 1$ is obvious, since the out-degree of s is one. In the following lemma we prove the induction step.

Lemma 6.6 *For any $i \geq 1$ and $i \leq k$, $|C_{i+1}| \leq 2|C_i|$.*

Proof: Let D denote the set of edges originating from S_{i+1} and terminating to $V \setminus S_{i+1}$. Since S_{i+1} contains A_i, all the edges in set E_i^t must lie in D. Now consider an edge in $(u, v) \in D \setminus E_i^t$. Note that vertex u cannot lie in A_i, because then v must be either t or lie in $(\text{OUT}(A_i) \setminus \{t\}) \subseteq S_{i+1}$. Thus edges of $D \setminus E_i^t$ must originate from vertices of the set $\text{OUT}(A_i) \setminus \{t\}$. Since $|\text{OUT}(A_i) \setminus \{t\}| \leq |C_i \setminus E_i^t|$ and the out-degree of every vertex is at most two, so we get that $|D \setminus E_i^t| \leq 2|C_i \setminus E_i^t|$. Thus, $|D| \leq |E_i^t| + 2|C_i \setminus E_i^t| \leq 2|C_i|$. Since D is an (S_i, t) cut, we get that the size of (S_i, t) min-cut must be bounded by $2|C_i|$. $\square$

Remark: The fact that the out-degree of each vertex is upper bounded by 2 played a crucial role in bounding the size of k-FTRS in the proof of Lemma 6.6.

We now analyze the running time of our algorithm to compute a k-FTRS(t) for any $t \in V$. The first step in computation of k-FTRS(t) is to transform G into a graph with $O(m)$ vertices and edges such that out-degree of each vertex is bounded by two. This takes $O(m)$ time (see Appendix). Next we apply Algorithm 1 on this transformed graph. The time complexity of Algorithm 1 is dominated by the time required for computing the k farthest min-cuts which is $O(\sum_{i=1}^{k} m \times |C_i|) = O(2^k m)$ time (see [12]). Finally a k-FTRS(t) for original graph can be extracted form a k-FTRS(t) of transformed graph in $O(m)$ time (see Appendix). Thus a k-FTRS(t) for any vertex t can be computed in $O(2^k m)$ time.

Since computation of a k-FTRS requires n rounds, where in each round we compute k-FTRS(v) for some $v \in V$, the total time complexity is $O(2^k mn)$ (see proof of Lemma 4.1). We thus conclude with the following theorem.

Reminder of Theorem 1.1 *There exists an $O(2^k mn)$ time algorithm that for any given integer $k \geq 1$, and any given directed graph G on n vertices, m edges and a designated source vertex s, computes a k-FTRS for G with at most $2^k n$ edges. Moreover, the in-degree of each vertex in this k-FTRS is bounded by 2^k.*

7 Lower bound

We shall now show that for each k, n ($n \geq 2^k$), there exists a directed graph G with $O(n)$ vertices whose k-FTRS must have $\Omega(2^k n)$ edges. Let T be a balanced binary tree of height k rooted at s. Let X be the set of leaf nodes of T, thus $|X| = 2^k$. Let Y be another set of n vertices. Then the graph G is obtained by adding an edge from each $x \in X$ to each $y \in Y$. In other words, $V(G) = V(T) \cup Y$ and $E(V) = E(T) \cup (X \times Y)$. Figure 3 illustrates the graph for $k = 3$.

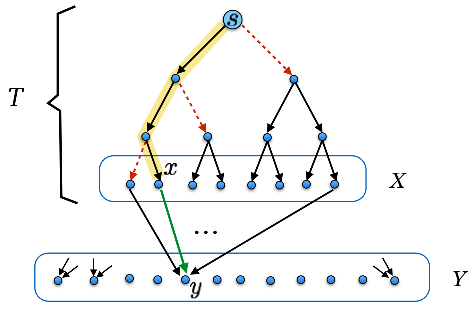

Figure 3: For each edge (x, y), there exist 3 edges (shown dotted) whose failure will render y unreachable from s unless (x, y) is kept in the 3-FTRS.

We now show that a k-FTRS for G must contain all edges of G. It is easy to see that all edges of T must be present in a k-FTRS of G. Thus let us consider an edge $(x, y) \in X \times Y$. Let P be the tree path from s to leaf node x. Let F be the set of all those edges $(u, v) \in T$ such that $u \in P$ and v is the *child* of u not lying on P. Clearly $|F| = k$. Observe that x is the only leaf node of T reachable from s on the failure of the edges in set F. Thus $P{::}(x, y)$ is the unique path from s to y in $G \setminus F$. This shows that edge (x, y) must lie in a k-FTRS for G. This establishes a lower bound of $\Omega(2^k n)$ on the size of k-FTRS for G.

Reminder of Theorem 1.2 *For any positive integers n, k with $n \geq 2^k$, there exists a directed graph on n vertices whose k-FTRS must have $\Omega(2^k n)$ edges.*

8 Applications

In this section we present a few simple applications of k-FTRS.

1. Detecting if a set of k edge/vertex failures alters the strong connectivity of a graph.

Georgiadis et. al [13] gave an $O(n)$ size oracle that, after any edge or vertex failure, can report the strongly connected components (SCCs) of the remaining graph in $O(n)$ time. Here we consider the more restricted problem where we are only interested in finding out if a set of k edge/vertex failures alters the SCCs of G. Our construction works as follows. Let G^R denote the graph obtained by reversing each edge of graph G. Let $S_1, \ldots, S_t$ be the partition of V corresponding to the SCCs of G, and let s_i be any arbitrary vertex in S_i. For $1 \leq i \leq t$, let G_i (G_i^R) denote the graph induced by set S_i in G (G^R). For each $i \in [1, t]$, let H_i and H_i^R denote the k-FTRS for G_i and G_i^R respectively with s_i as the source vertex. Our data structure is simply the collection of the subgraphs $H_1, H_1^R, ..., H_t, H_t^R$. Given any query set F of at most k failing edges/vertices, we perform traversal from s_i in graphs $H_i \setminus F$ and $H_i^R \setminus F$, where $i \in [1, t]$. The SCC S_i is preserved if and only if the set of vertices reachable from s_i in both $H_i \setminus F$ and $H_i^R \setminus F$ is the same as the set S_i. Note that the total space used, and the query time after any k failures are both bounded by $O(2^k n)$. Thus we get the following theorem.

Theorem 8.1 *Given any directed graph $G = (V, E)$ on n vertices, a source $s \in V$, and a positive integer k, we can preprocess G in polynomial time to build an $O(2^k n)$ size data structure that, after the failure of any set F of k edges or vertices can determine in $O(2^k n)$ time whether the SCCs of graph $G \setminus F$ are the same as that of graph G.*

2. Fault Tolerant algorithm for reporting Dominator Tree of a graph.

Given a directed graph G and a source vertex s we say that vertex v dominates vertex w if every path from s to w contains v [14]. The vertex v is the immediate dominator of w if every dominator of w (other than w itself) is also a dominator of v. A dominator tree is a tree rooted at s where each node's children are those nodes that it immediately dominates [14]. Buchsbaum et al. [5] gave an $O(m)$ time algorithm for computing dominators and dominator tree. Here we show how this algorithm can be combined with the concept of k-FTRS to obtain a fault tolerant algorithm for reporting the dominator tree after the failure of any k edges or vertices. Let H be a $(k+1)$-FTRS for graph G. It is easy to see that on failure of any set F of k edges or vertices, the graph $H \setminus F$ is still a 1-FTRS for graph $G \setminus F$. Thus dominators of a vertex w in graph $H \setminus F$ are identical to that in graph $G \setminus F$. So in order to compute the dominator tree of $G \setminus F$ it suffices to run the algorithm of Buchsbaum et al. [5] on graph $H \setminus F$. This would take time of the order of the number of edges in H, that is, $O(2^k n)$. The space used is also $O(2^k n)$ as it suffices to store just the graph H. Thus we get that the following theorem.

Theorem 8.2 *Given any directed graph $G = (V, E)$ on n vertices, a source $s \in V$, and a positive integer k, we can preprocess G in polynomial time to build an $O(2^k n)$ size data structure that, after the failure of any set F of k edges of vertices, can compute the dominator tree of $G \setminus F$ in $O(2^k n)$ time.*

References

[1] S. Baswana, S. R. Chaudhury, K. Choudhary, and S. Khan. Dynamic DFS in undirected graphs: breaking the O(*m*) barrier. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 730–739, 2016.

[2] S. Baswana, K. Choudhary, and L. Roditty. Fault tolerant reachability for directed graphs. In Y. Moses, editor, *Distributed Computing - 29th International Symposium, DISC 2015, Tokyo, Japan, October 7-9, 2015, Proceedings*, volume 9363 of *Lecture Notes in Computer Science*, pages 528–543. Springer, 2015.

[3] S. Baswana and N. Khanna. Approximate shortest paths avoiding a failed vertex: Near optimal data structures for undirected unweighted graphs. *Algorithmica*, 66(1):18–50, 2013.

[4] D. Bilò, L. Gualà, S. Leucci, and G. Proietti. Multiple-edge-fault-tolerant approximate shortest-path trees. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, pages 18:1–18:14, 2016.

[5] A. L. Buchsbaum, L. Georgiadis, H. Kaplan, A. Rogers, R. E. Tarjan, and J. Westbrook. Linear-time algorithms for dominators and other path-evaluation problems. *SIAM J. Comput.*, 38(4):1533–1573, 2008.

[6] S. Chechik. Fault-tolerant compact routing schemes for general graphs. *Inf. Comput.*, 222:36–44, 2013.

[7] S. Chechik, M. Langberg, D. Peleg, and L. Roditty. Fault tolerant spanners for general graphs. *SIAM J. Comput.*, 39(7):3403–3423, 2010.

[8] S. Chechik, M. Langberg, D. Peleg, and L. Roditty. f-sensitivity distance oracles and routing schemes. *Algorithmica*, 63(4):861–882, 2012.

[9] C. Demetrescu, M. Thorup, R. A. Chowdhury, and V. Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, 2008.

[10] M. Dinitz and R. Krauthgamer. Fault-tolerant spanners: better and simpler. In C. Gavoille and P. Fraigniaud, editors, *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 169–178. ACM, 2011.

[11] R. Duan and S. Pettie. Dual-failure distance and connectivity oracles. In *SODA'09: Proceedings of 19th Annual ACM -SIAM Symposium on Discrete Algorithms*, pages 506–515, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[12] J. Ford and D. Fullkerson. *Flows in networks*. Princeton University Press, Princeton, 1962.

[13] L. Georgiadis, G. F. Italiano, and N. Parotsidis. A new framework for strong connectivity and 2-connectivity in directed graphs. *CoRR*, abs/1511.02913, 2015.

[14] T. Lengauer and R. E. Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Trans. Program. Lang. Syst.*, 1(1):121–141, 1979.

[15] M. Parter. Dual failure resilient BFS structure. In C. Georgiou and P. G. Spirakis, editors, *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 481–490. ACM, 2015.

[16] M. Parter and D. Peleg. Sparse fault-tolerant BFS trees. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 779–790, 2013.

[17] M. Parter and D. Peleg. Fault tolerant approximate BFS structures. In C. Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1073–1092. SIAM, 2014.

[18] M. Teslenko and E. Dubrova. An efficient algorithm for finding double-vertex dominators in circuit graphs. In *2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005), 7-11 March 2005, Munich, Germany*, pages 406–411, 2005.

A Justification for Assumption 1

Let G be the input graph. We construct a new graph $H = (V', E')$ from G such that out-degree of each vertex in it is bounded by two as follows.

(i) For each u in V, construct a binary tree B_u such that the number of leaves in B_u is exactly equal to out-degree (say $d(u)$) of u in G. Let u^r be the root of this tree, and $u^\ell_1, .., u^\ell_{d(u)}$ be its leaves.

(ii) Insert the binary tree B_u in place of out-edges of u in G as follows. We delete all the out-edges, say $(u, v_1), \ldots, (u, v_{d(u)})$, of vertex u. Next we connect u to u^r, and u^ℓ_i to v_i, for each $i \leq d(u)$.

Observe that H constructed in this manner will have $O(m)$ edges and vertices. In this process, the out-degree of each vertex in H gets bounded by 2, though in-degree remains unchanged. Notice that an edge in G is mapped to a path in H as follows.

$$(u, v_i) \mapsto (u, u^r)\text{::(path from } u^r \text{ to } u^\ell_i \text{ in } B_u\text{)::}(u^\ell_i, v_i)$$

We now show how to construct a k-FTRS(t) (say G^*) for G. Let H^* be a k-FTRS(t) for graph H. For each out-neighbor v_i of a vertex u in G, include edge (u, v_i) in G^* if and only if edge (u^ℓ_i, v_i) is present in H^*. Now consider any set F of k failed edges in G. Define a set F' of failed edges in H by adding edge (u^ℓ_i, v_i) to F' for each $(u, v_i) \in F$. It can be inferred from the mapping defined above that there is a path from s to t in $G^* \setminus F$ if and only if there is a path from s to t in $H^* \setminus F'$. Thus graph G^* is a k-FTRS(t) for graph G with in-degree(t, G^*) equal to in-degree(t, H^*). This shows that computing a k-FTRS(t) for $t \in V$ in G is equivalent to computing k-FTRS(t) in graph H which has $O(m)$ edges and vertices, and out-degree of each vertex is bounded by two. Hence Assumption 1 is justified.