SIAM J. COMPUT.
Vol. 47, No. 1, pp. 80–95

FAULT-TOLERANT SUBGRAPH FOR SINGLE-SOURCE REACHABILITY: GENERAL AND OPTIMAL*

SURENDER BASWANA†, KEERTI CHOUDHARY†, AND LIAM RODITTY‡

Abstract. Let G be a directed graph with n vertices, m edges, and a designated source vertex s. We address the problem of single-source reachability (SSR) from s in the 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 $v \in V(G)$ is reachable from s in $G \setminus F$ if and only if v is reachable from s in $H \setminus F$. We call H a k-fault tolerant reachability subgraph (k-FTRS). We also prove a matching lower bound of $\Omega(2^k n)$ edges for such subgraphs that holds for all n, k with $2^k \leq n$. Our results extend to vertex failures without any extra overhead. The construction of k-FTRS is interesting from several different perspectives. From the *Graph theory* perspective 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 and undirected graphs, Demetrescu et al. showed that there is a lower bound of $\Omega(m)$ edges even for a single edge failure [*SIAM J. Comput.,* 37 (2008), pp. 1299–1318]. In the case of unweighted graphs Parter and Peleg gave a lower bound of $\Omega(n^{3/2})$ edges, again, even for a single edge failure [*Proc. Algorithms—21st Annual European Symposium*, 2013, pp. 779–790]. From the *Algorithms* perspective 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, and (ii) computing a dominator tree of a graph after k-failures. From the perspective of *techniques* it makes an interesting usage of the concept of farthest min-cut which was already introduced by Ford and Fulkerson in their pioneering work on flows and cuts [*Flows in Networks*, Princeton University Press, 1962; reprinted 2011]. 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.

Key words. fault-tolerant, subgraph, reachability, min-cut

AMS subject classifications. 05C20, 05C85, 68W05, 68W40

DOI. 10.1137/16M1087643

1. Introduction.

Networks in most real-life applications are prone to failures. Such networks can be modeled as graphs 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 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. Our objective is to construct a sparse subgraph that preserves the reachability from a given fixed source s even after k failures. The following definition

*Received by the editors August 3, 2016; accepted for publication (in revised form) September 25, 2017; published electronically January 11, 2018. A preliminary version of this paper appeared in *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing (STOC* 2016*), pp.* 509–518.

http://www.siam.org/journals/sicomp/47-1/M108764.html

Funding: 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 Ph.D. Fellowship Award.

†Department of Computer Science and Engineering, I.I.T. Kanpur, Kalyanpur, Kanpur, Uttar Pradesh 208016, India (sbaswana@cse.iitk.ac.in, keerti@cse.iitk.ac.in).

‡Department of Computer Science, Bar Ilan University, Ramat Gan, 5290002, Israel (liam.roditty@biu.ac.il).

precisely characterizes this subgraph.

DEFINITION 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 every 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 $H \setminus F$.*

The reachability problem is fundamental in many applications of both theoretical and applied computer science (see [27] and references therein). Therefore, a result on the existence of a sparse k-FTRS and its efficient construction will also be very important and could have many real-world applications. Moreover, the reachability problem lies at the core of many other graph problems, namely, strong-connectedness, dominators [20], double-dominators [26], etc. Thus obtaining a sparse k-FTRS should help in obtaining fault-tolerant solution for these problems.

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 [20]. Given a depth-first-search (DFS) tree T rooted at s, using the ideas of [20] it is straightforward to compute a 1-FTRS with at most $2n$ edges that contains T. In [2], Baswana, Choudhary, and Roditty show that even if T is any arbitrary reachability tree,[1] 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 2. *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 3. *For any positive integers n, k with $n \geq 3 \cdot 2^k$, there exists a directed graph $G = (V, E)$ on n vertices and a source vertex $s \in V$ whose k-FTRS with respect to s must have $(2^k n/3)$ 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 a data structure of $O(2^k n)$ words that, after the failure of any set F of k edges or vertices, can determine in $O(2^k n)$ time whether the strongly connected components of graph $G \setminus F$ are the same as those 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 a data structure of $O(2^k n)$ words 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.

In addition to the above applications, our techniques reveal an interesting connec-

[1] The term "reachability tree" in [2] is used for denoting a subgraph of G which (i) is an arborescence (directed tree) rooted at source s, and (ii) contains all the vertices reachable from s in G.

tion 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 (for more details, see subsection 4.2). 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.

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 each 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}.

1.1. Related work. The problem most closely related to single-source reachability (SSR) is that of single-source shortest paths (SSSP). Similarly 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 by k-FTSS. Unfortunately, for weighted graphs (directed or undirected), no sparse k-FTSS is possible—Demetrescu et al. [13] showed that there exist graphs whose 1-FTSS must have $\Omega(m)$ edges. For the case of unweighted graphs, Parter and Peleg [25] showed that we can compute a 1-FTSS with $O(n^{3/2})$ edges; they also showed a matching lower bound. Recently, Parter [23] extended this result to 2-FTSS with $O(n^{5/3})$ edges for unweighted 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.

In contrast to the situation with FTSS, our construction of general 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 SSR and SSSP in directed graphs.

For undirected weighted graphs, there exist various results for sparse fault-tolerant subgraphs that preserve an approximate distance from the source. 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. Nardelli, Proietti, and Widmayer [21] showed that that there is a subgraph with $2n$ edges that preserves the distances from s up to a multiplicative stretch of 3 upon failure of a single edge. Bilò et al. [5] showed that we can compute a subgraph with $O(n \log n/\epsilon^2)$ edges that preserves a $(1+\epsilon)$-shortest path after failure of an edge as well as a vertex. For the case of edge failures, sparse fault-tolerant subgraphs exist for general k. Bilò et al. [6] 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 gave a data structure of $O(kn \log^2 n)$ words that can report distances from s stretched by a factor of at most $(2k+1)$ in $O(k^2 \log^2 n)$ time.

For the case of all-pair shortest paths (APSP) in weighted directed graphs, Demetrescu et al. [13] 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 [15] 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 DFS trees, graph spanners, approximate distance oracles, and compact routing schemes have been studied in [1, 10, 11, 12, 14, 22, 4, 24].

1.2. Organization of the paper. We describe notation and terminology 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, for better understanding we first present a simple construction of a 2-FTRS with $4n$ edges in section 5. We describe the general 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 notation will be used throughout the paper:

- $E(f)$: Edges of the graph G carrying a nonzero flow for a given flow f.
- $E(P)$: Edges lying on a path P.
- $E(A)$: Edges of the graph G whose endpoints both 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.
- $H \setminus F$: The graph obtained by deleting the edges lying in set F from graph H.
- $H + (u, v)$ (respectively, $H + F$): The graph obtained by adding an edge (u, v) (respectively, a set of edges F) to graph H.
- MAX-FLOW(H, S, t): The value of the maximum flow in graph H, where the source is a set of vertices S and the destination is a single vertex t.
- OUT(A): The set of all those vertices in $V \setminus A$ having an incoming edge from some vertex of A, where $A \subseteq V$.
- IN-EDGES(v, H): The set of all incoming edges to v in H. When the graph H is clear from the context, we denote it simply by 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. Thus, 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 4. *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.*

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

DEFINITION 5. *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 every 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 6. *A subgraph H of $G = (V, E)$ is a k-FTRS if and only if H is a k-FTRS(v) for every $v \in V$.*

3. Overview. Our starting point is a lemma (referred to as the 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 the in-degree of v is bounded by some integer c_k, then on applying this algorithm recursively on an arbitrary sequence of vertices of G, we can get a k-FTRS for G in which the in-degree of all the vertices is bounded by c_k.

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 an (s,t)-cut C is a partition of 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) as 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 9). 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), this is not problematic because 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 2.

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. Below we provide the justification for the above assumption.

3.1. Justification for Assumption 1. Let G be the input graph. We construct a new graph $H = (V', E')$ from G such that the out-degree of each vertex in H is bounded by 2 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 the out-degree (say $d(u)$) of u in G. Let u^r be the root of this tree, and let $u^\ell_1, \ldots, u^\ell_{d(u)}$ 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 the in-degree of the original vertices remains unchanged. Notice that an edge in G is mapped to a path in H as follows:

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

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_i^ℓ, 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 the out-degree of each vertex is bounded by 2. Hence Assumption 1 is justified.

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 7. *Suppose there exist an algorithm $\mathcal{A}$ and an integer c_k satisfying the following condition: Given any graph G and a vertex v in G, $\mathcal{A}$ computes a subgraph H of G such that*

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

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

Then, we can compute a k-FTRS for G with at most $c_k n$ 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 $v_1, \ldots, v_i$ 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 $\mathcal{A}$; and (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 the in-degree of vertices $v_1, \ldots, v_i$ in graph G_i would be bounded by c_k. We now show, using induction, that the graphs $G_0, G_1, \ldots, G_n$ 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}, and 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}. ∎

4.2. Farthest min-cut.

DEFINITION 8. *Let S be a source set and t be a destination vertex. Any (S,t)-cut C is a partition of the vertex set into two sets, $A(C)$ and $B(C)$, where $S \subseteq A(C)$ and $t \in B(C)$. An (S,t)-min-cut C^* is said to be the* farthest min-cut *if $A(C^*) \supsetneq A(C)$ for every (S,t)-min-cut C other than C^*. We denote C^* by* FMC(G, S, t).

In 1962 Ford and Fulkerson [16] showed the construction of the farthest (S,t)-min-cut and also gave an algorithm for constructing it. For the sake of completeness we state the following result from the 2011 reprint of [16] (for the proof see Theorem 5.5 of Chapter 1, which is freely available on the Web).

LEMMA 9. *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 at 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 1 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 10. *Let S be a source set, t be a destination vertex, C be* $\mathrm{FMC}(G,S,t)$*, 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,* $\textsc{max-flow}(G',S,t) = 1 + \textsc{max-flow}(G,S,t)$*, and $C' = C \cup \{(s,w)\}$ forms an (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 9 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 $\textsc{max-flow}(G',S,t)$ is greater than $\textsc{max-flow}(G,S,t)$. Since G' is obtained by adding only one extra edge to G, the value of max-flow cannot increase by more than 1, and hence we get that $\textsc{max-flow}(G',S,t)$ is equal to $1 + \textsc{max-flow}(G,S,t)$.

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 $|C'| = 1 + |C| = 1 + \textsc{max-flow}(G,S,t) = \textsc{max-flow}(G',S,t)$. 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. ∎

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

LEMMA 11. *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* $\mathrm{FMC}(S,t)$*, 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 $\alpha = \textsc{max-flow}(G,S,t)$ and $\beta = \textsc{max-flow}(G,s,t)$. Let $e_1,\ldots,e_\alpha$ be the edges lying in the cut C, and 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 $C \cap E(f') = \{e_1,\ldots,e_\beta\}$. Now let $\{(P_i :: e_i :: P_i') : i \le \alpha\}$[2] 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 a 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, 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 because the Q_i's lie entirely in $G(A)$ and the 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 $E(f) \subseteq E(A) \cup E(f_S)$. ∎

LEMMA 12. *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* $\mathrm{FMC}(s,t)$*, and (A,B) be the partition of V induced by* $\mathrm{FMC}(S,t)$*. Then $B \subseteq \hat{B}$.*

[2]With slight abuse of notation, while concatenating, we can view an edge as a path of length 1.

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 10, we have that MAX-FLOW$(G', S, t) = 1 +$ 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 11 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. Hence path P lies in the residual graph corresponding to max-flow f from s to t in G. So vertex x must lie in $\hat{B}$. Hence we have $B \subseteq \hat{B}$. ∎

5. A 2-FTRS with $4n$ edges. From Lemma 7 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 the out-degree of s is at most 2. Thus, the value of max-flow from s to t must be either 1 or 2. Below we explain how to construct a 2-FTRS(t) in each of these cases.

Case 1. MAX-FLOW$(G, s, t) = 2$: Let $C = \{(a, b), (a', b')\}$[3] 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 on whether this path passes through (a, b) or (a', b'). Since C is the farthest min-cut of G, it follows from Lemma 10 that the value of (s, t) max-flow in both graphs $G + (s, b)$ and $G + (s, b')$ must be 3. So we denote by P_0, P_1, P_2 any three edge disjoint paths from s to t in $G+(s, b)$ (see Figure 1(ii)) and similarly denote by P'_0, P'_1, P'_2 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 13. *Let E_t be a subset of incoming edges to t satisfying the following two conditions:*

(i) *E_t contains the last edges on paths P_0, P_1, P_2 in case $b \neq t$ and contains the edge (a, b) in case $b = t$.*
(ii) *E_t contains the last edges on paths P'_0, P'_1, P'_2 in case $b' \neq t$ and contains 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. Thus, 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 when $b \neq t$. 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 P_0 must be (s, b). In the latter case we replace the edge (s, b) by

[3]Note that (a, b) and (a', b') are two different edges; however, it may happen that either $a = a'$ or $b = b'$.

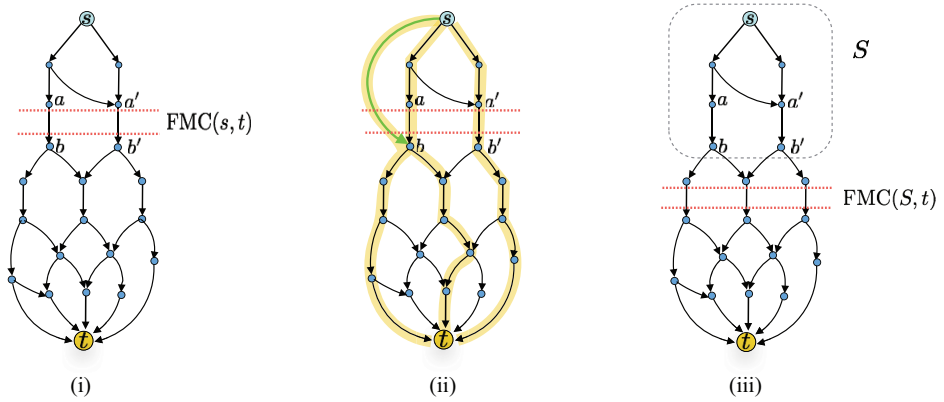

FIG. 1. (i) *Edges* (a,b) *and* (a',b') *represent the farthest min-cut from* s *to* t. (ii) *Paths highlighted in yellow represent three edge disjoint paths* P_0, P_1, P_2 *in graph* $G + (s,b)$. (iii) *As* $b, b' \neq t$, *source* S *is equal to* $A \cup \{b, b'\}$, *and* $\text{FMC}(G, S, t) = 3$.

path $R[s,b]$, 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). ∎

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 form 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 nonzero flow with respect to f_S is a 2-FTRS(t).

LEMMA 14. *Let* $\mathcal{E}(t)$ *be the set of incoming edges to* t *carrying a nonzero flow with respect to* f_S. *Then the graph* $G^* = \big(G \setminus \text{IN-EDGES}(t)\big) + \mathcal{E}(t)$ *is a* 2-FTRS(t).

Proof. In order to prove this lemma it suffices to show that $\mathcal{E}(t)$ satisfies the conditions required in Lemma 13. Here we show that $\mathcal{E}(t)$ satisfies condition (i) of Lemma 13. The proof of condition (ii) 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 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 11 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 10 implies that the 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. ∎

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 2. Therefore, the

value of (S,t)-min-cut (and max-flow) can be at most 4.

Case 2. 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 1); let this graph be G^*. It is easy to verify that G^* is a 2-FTRS(t) with s as source, and the in-degree of t in G^* is bounded by 4.

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

THEOREM 15. *There exists a polynomial time algorithm that, for any given directed graph G on n vertices, computes a* 2-FTRS *for G with at most $4n$ edges.*

6. Computing a k-FTRS. In this section we prove Theorem 2. Let t be a vertex in G for which k-FTRS(t) from s needs to be computed. (Recall that from Lemma 7 it follows that this is sufficient in order to prove Theorem 2.) 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.

The above 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. Algorithm for computing k-FTRS(t).

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

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 unioned with those vertices in $B_i\setminus\{t\}$ that have an incoming edge from any vertex of A_i. Notice that since $S_i\subseteq A_i$, this 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 $\mathcal{E}(t)$. 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

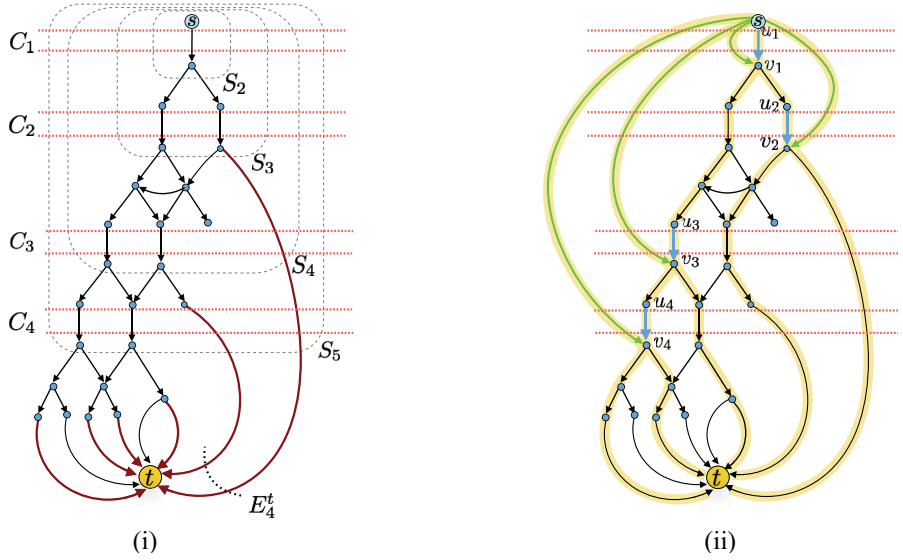

FIG. 2. (i) *The edges in brown constitute the set* $\mathcal{E}(t)$ *when* $k = 4$. (ii) *Paths highlighted in yellow represent five edge disjoint paths in graph* H.

$(i \in [1, k])$ may terminate at t; we denote this set by E_i^t. The following lemma shows that these edges are always included in our FTRS G^*.

LEMMA 16. *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. ∎

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 16 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, namely, H_i's (for every $i \in [1, k + 1]$) as follows:

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

Let $H = H_{k+1}$ be the graph obtained by adding all the edges $(s, v_1), \ldots, (s, v_k)$ 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 17. $C_i = \text{FMC}(H_i, S_i, t)$.

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

Proof. We show by induction that $\text{MAX-FLOW}(H_i, s, t) = i$ for each $i \in [1, k+1]$. Note that $H_1 = G$. The base case holds since the out-degree of s is 1 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 17 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 12 that $B_i \subseteq \hat{B}$. Therefore, using Lemma 10 we get $\text{MAX-FLOW}(H_{i+1}, s, t) = 1 + \text{MAX-FLOW}(H_i, s, t) = i + 1$. ∎

Next, we define H^* to be a graph obtained from 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 19. $\text{MAX-FLOW}(H^*, s, t) = k + 1$.

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 11 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 18 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-$\text{FTRS}(t)$.

LEMMA 20. *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 19 we get that there exist $k+1$ 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 $(s, v_1), \ldots, (s, v_k)$ 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 $G^* \setminus F$. ∎

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 1. In the following lemma we prove the induction step.

LEMMA 21. *For each $i \geq 1$ 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, 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 $2|C_i|$. ∎

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 21.

Analysis of running time. We now analyze the running time of our algorithm to compute a k-FTRS(t) for any $t \in V$. The first step in the computation of k-FTRS(t) is to transform G into a graph with $O(m)$ vertices and edges such that the out-degree of each vertex is bounded by 2. This takes $O(m)$ time (see subsection 3.1). 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 $O(\sum_{i=1}^{k} m \times |C_i|) = O(2^k m)$ (see [16]). Finally, a k-FTRS(t) for the original graph can be extracted from a k-FTRS(t) of the transformed graph in $O(m)$ time (see subsection 3.1). 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 the proof of Lemma 7).

We conclude with the following theorem.

REMINDER OF THEOREM 2. *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. A matching lower bound. We shall now show that for each k, n ($n \geq 3 \cdot 2^k$), there exists a directed graph G with 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 $n - |V(T)|$ 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 $k = 3$.

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. Notice that G contains at least $2^k|Y| > 2^k(n - 2^{k+1}) \geq 2^k n/3$ edges. This establishes a lower bound of $(2^k n/3)$ on the size of k-FTRS for G.

REMINDER OF THEOREM 3. *For any positive integers n, k with $n \geq 3 \cdot 2^k$, there exists a directed graph $G = (V, E)$ on n vertices and a source vertex $s \in V$ whose k-FTRS with respect to s must have $(2^k n/3)$ edges.*

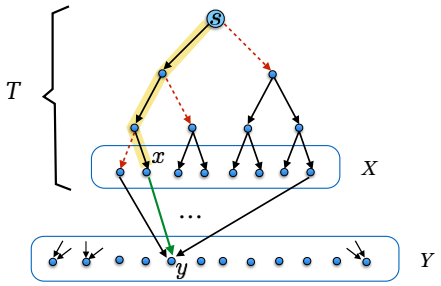

FIG. 3. *For each edge (x, y), there exist three edges (shown dotted) whose failure will render y unreachable from s unless (x, y) is kept in the* 3-FTRS.

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

8.1. Detecting if a set of k edge/vertex failures alters the strong connectivity of a graph. Georgiadis, Italiano, and Parotsidis [17] 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 $H_1, H_1^R, \ldots, H_t, H_t^R$. 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 22. *Given any directed graph $G = (V, E)$ on n vertices and a positive integer k, we can preprocess G in polynomial time to build a data structure of $O(2^k n)$ words that, after the failure of any set F of at most k edges or vertices, can determine in $O(2^k n)$ time whether the SCCs of graph $G \setminus F$ are the same as those of graph G.*

8.2. 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 [20]. 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 the children of each node are those nodes that it immediately dominates [20]. Buchsbaum et al. [9] 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 those 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. [9] 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 the following theorem.

THEOREM 23. *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 a data structure of $O(2^k n)$ words that, after the failure of any set F of at most k edges or vertices, can compute the dominator tree of $G \setminus F$ in $O(2^k n)$ time.*

9. Future work. In this paper, we showed the construction of a sparse subgraph with at most $2^k n$ edges that preserves reachability from a designated source s to each $v \in V$ after a failure of any set of at most k edges or vertices. There are two natural extensions of this problem. The first is to extend it from single-source to multiple sources: given a source set $S \subset V$ and any integer k, compute a sparse subgraph that preserves reachability for each $v \in V$ from each $s \in S$ upon failure of any k edges or vertices. The second is to compute a pairwise FTRS structure: given a set of pairs $P \subset V \times V$, and any integer k, compute a sparse subgraph that preserves reachability from u to v for each $(u, v) \in P$ upon failure of any k edges or vertices. These extensions have been previously studied for the well-known problem of spanners for undirected graphs [18, 19, 8, 7]. Another important direction of study is computing a sparse subgraph that preserves a strong-connectivity or a biconnectivity relation among the vertices of a given graph upon failure of any k edges or vertices.

REFERENCES

[1] S. BASWANA, S. R. CHAUDHURY, K. CHOUDHARY, AND S. KHAN, *Dynamic DFS in undirected graphs: Breaking the O(m) barrier*, in Proc. Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016), Arlington, VA, SIAM, 2016, pp. 730–739, https://doi.org/10.1137/1.9781611974331.ch52.

[2] S. BASWANA, K. CHOUDHARY, AND L. RODITTY, *Fault tolerant reachability for directed graphs*, in Proc. Distributed Computing—29th International Symposium (DISC 2015), Tokyo, Japan, 2015, pp. 528–543.

[3] S. BASWANA AND N. KHANNA, *Approximate shortest paths avoiding a failed vertex: Near optimal data structures for undirected unweighted graphs*, Algorithmica, 66 (2013), pp. 18–50, https://doi.org/10.1007/s00453-012-9621-y.

[4] D. BILÒ, F. GRANDONI, L. GUALÀ, S. LEUCCI, AND G. PROIETTI, *Improved purely additive fault-tolerant spanners*, in Proc. Algorithms—23rd Annual European Symposium (ESA 2015), Patras, Greece, 2015, pp. 167–178.

[5] D. BILÒ, L. GUALÀ, S. LEUCCI, AND G. PROIETTI, *Fault-tolerant approximate shortest-path trees*, in Proc. Algorithms—22nd Annual European Symposium (ESA 2014), Wroclaw, Poland, 2014, pp. 137–148.

[6] D. BILÒ, L. GUALÀ, S. LEUCCI, AND G. PROIETTI, *Multiple-edge-fault-tolerant approximate shortest-path trees*, in Proc. 33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016), Orléans, France, 2016, pp. 18:1–18:14.

[7] G. BODWIN, *Linear size distance preservers*, in Proc. Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017), Barcelona, Spain, SIAM, 2017, pp. 600–615, https://doi.org/10.1137/1.9781611974782.39.

[8] G. BODWIN AND V. V. WILLIAMS, *Better distance preservers and additive spanners*, in Proc. Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016), Arlington, VA, 2016, pp. 855–872.

[9] 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 (2008), pp. 1533–1573, https://doi.org/10.1137/070693217.

[10] S. CHECHIK, *Fault-tolerant compact routing schemes for general graphs*, Inf. Comput., 222 (2013), pp. 36–44, https://doi.org/10.1016/j.ic.2012.10.009.

[11] S. CHECHIK, M. LANGBERG, D. PELEG, AND L. RODITTY, *Fault tolerant spanners for general graphs*, SIAM J. Comput., 39 (2010), pp. 3403–3423, https://doi.org/10.1137/090758039.

[12] S. CHECHIK, M. LANGBERG, D. PELEG, AND L. RODITTY, *f-sensitivity distance oracles and routing schemes*, Algorithmica, 63 (2012), pp. 861–882, https://doi.org/10.1007/

s00453-011-9543-0.

[13] C. Demetrescu, M. Thorup, R. A. Chowdhury, and V. Ramachandran, *Oracles for distances avoiding a failed node or link*, SIAM J. Comput., 37 (2008), pp. 1299–1318, https://doi.org/10.1137/S0097539705429847.

[14] M. Dinitz and R. Krauthgamer, *Fault-tolerant spanners: Better and simpler*, in Proc. 30th Annual ACM Symposium on Principles of Distributed Computing (PODC 2011), San Jose, CA, C. Gavoille and P. Fraigniaud, eds., ACM, 2011, pp. 169–178, https://doi.org/10.1145/1993806.1993830.

[15] R. Duan and S. Pettie, *Dual-failure distance and connectivity oracles*, in Proc. 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009), SIAM, Philadelphia, 2009, pp. 506–515, https://doi.org/10.1137/1.9781611973068.56.

[16] L. R. Ford, Jr., and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962; reprinted in 2011 by Princeton University Press, http://press.princeton.edu/titles/9233.html.

[17] L. Georgiadis, G. F. Italiano, and N. Parotsidis, *Strong connectivity in directed graphs under failures, with applications*, in Proc. Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017), Barcelona, Spain, SIAM, 2017, pp. 1880–1899, https://doi.org/10.1137/1.9781611974782.123.

[18] T. Kavitha, *New pairwise spanners*, in Proc. 32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015), Garching, Germany, 2015, pp. 513–526.

[19] T. Kavitha and N. M. Varma, *Small stretch pairwise spanners and approximate D-preservers*, SIAM J. Discrete Math., 29 (2015), pp. 2239–2254, https://doi.org/10.1137/140953927.

[20] T. Lengauer and R. E. Tarjan, *A fast algorithm for finding dominators in a flowgraph*, ACM Trans. Program. Lang. Syst., 1 (1979), pp. 121–141, https://doi.org/10.1145/357062.357071.

[21] E. Nardelli, G. Proietti, and P. Widmayer, *Swapping a failing edge of a single source shortest paths tree is good and fast*, Algorithmica, 35 (2003), pp. 56–74.

[22] M. Parter, *Vertex fault tolerant additive spanners*, in Proc. Distributed Computing—28th International Symposium (DISC 2014), Austin, TX, 2014, pp. 167–181.

[23] M. Parter, *Dual failure resilient BFS structure*, in Proc. 2015 ACM Symposium on Principles of Distributed Computing (PODC 2015), Donostia-San Sebastián, Spain, C. Georgiou and P. G. Spirakis, eds., ACM, 2015, pp. 481–490, https://doi.org/10.1145/2767386.2767408.

[24] M. Parter, *Fault-tolerant logical network structures*, Bull. EATCS, 118 (2016), http://eatcs.org/beatcs/index.php/beatcs/article/view/403.

[25] M. Parter and D. Peleg, *Sparse fault-tolerant BFS trees*, in Proc. Algorithms—21st Annual European Symposium (ESA 2013), Sophia Antipolis, France, 2013, pp. 779–790.

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

[27] S. J. van Schaik, *Answering Reachability Queries on Large Directed Graphs*, Ph.D. thesis, Utrecht University and University of Oxford, 2010, http://www.sj-vs.net/wp-content/uploads/2011/10/INF-SCR-10-10.pdf.