

Efficient Oracles and Routing Schemes for Replacement Paths

Davide Bilò

Università di Sassari, Italy
davidebilo@uniss.it

Keerti Choudhary

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

Luciano Gualà

Università di Roma “Tor Vergata”, Italy
guala@mat.uniroma2.it

Stefano Leucci

ETH Zürich, Switzerland
stefano.leucci@inf.ethz.ch

Merav Parter

CSAIL, MIT
meravparter@gmail.com

Guido Proietti

DISIM, Università dell’Aquila, Italy and
Istituto di Analisi dei Sistemi ed Informatica, CNR, Roma, Italy
guido.proietti@univaq.it

Abstract

Real life graphs and networks are prone to failure of nodes (vertices) and links (edges). In particular, for a pair of nodes s and t and a failing edge e in an n -vertex unweighted graph $G = (V(G), E(G))$, the *replacement path* $\pi_{G-e}(s, t)$ is a shortest $s-t$ path that avoids e . In this paper we present several efficient constructions that, for every $(s, t) \in S \times T$, where $S, T \subseteq V(G)$, and every $e \in E(G)$, maintain the collection of all $\pi_{G-e}(s, t)$, either *implicitly* (i.e., through compact data structures a.k.a. *distance sensitivity oracles* (DSO)), or *explicitly* (i.e., through sparse subgraphs a.k.a. *fault-tolerant preservers* (FTP)). More precisely, we provide the following results:

- (1) *DSO*: For every $S, T \subseteq V(G)$, we construct a DSO for maintaining $S \times T$ distances under single edge (or vertex) faults. This DSO has size $\tilde{O}(n\sqrt{|S||T|})$ and query time of $O(\sqrt{|S||T|})$. At the expense of having *quasi-polynomial* query time, the size of the oracle can be improved to $\tilde{O}(n|S| + |T|\sqrt{|S|n})$, which is optimal for $|T| = \Omega(\sqrt{n|S|})$. When $|T| = \Omega(n^{\frac{3}{4}}|S|^{\frac{1}{4}})$, the construction can be further refined in order to get a polynomial query time. We also consider the *approximate additive* setting, and show a family of DSOs that exhibits a tradeoff between the additive stretch and the size of the oracle. Finally, for the meaningful single-source case, the above result is complemented by a lower bound conditioned on the Set-Intersection conjecture. This lower bound establishes a *separation* between the oracle and the subgraph settings.
- (2) *FTP*: We show the construction of a *path-reporting* DSO of size $\tilde{O}(n^{4/3}(|S||T|)^{1/3})$ reporting $\pi_{G-e}(s, t)$ in $O(|\pi_{G-e}(s, t)| + (n|S||T|)^{1/3})$ time. Such a DSO can be transformed into a FTP having the same size, and moreover it can be elaborated in order to make it optimal (up to a poly-logarithmic factor) both in space and query time for the special case in which $T = V(G)$. Our FTP improves over previous constructions when $|T| = O(\sqrt{|S|n})$ (up to inverse poly-logarithmic factors).



© Davide Bilò, Keerti Choudhary, Luciano Gualà, Stefano Leucci, Merav Parter, and Guido Proietti;
licensed under Creative Commons License CC-BY

35th Symposium on Theoretical Aspects of Computer Science (STACS 2018).
Editors: Rolf Niedermeier and Brigitte Vallée; Article No. 13; pp. 13:1–13:15



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



SYMPOSIUM
ON THEORETICAL
ASPECTS
OF COMPUTER
SCIENCE

- (3) *Routing and Labeling Schemes*: For the well-studied single-source setting, we present a novel *routing scheme*, that allows to route messages on $\pi_{G-e}(s, t)$ by using edge labels and routing tables of size $\tilde{O}(\sqrt{n})$, and a header message of poly-logarithmic size. We also present a labeling scheme for the setting which is optimal in space up to constant factors.

2012 ACM Subject Classification Theory of computation \rightarrow Sparsification and spanners

Keywords and phrases Fault Tolerant, Shortest Path, Oracle, Routing

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.13

1 Introduction

1.1 Motivation

Shortest path in graphs is perhaps one of the most classical concepts in network algorithms. As real life networks are prone to failures, much attention has been devoted, recently, for studying *replacement paths*, namely, shortest paths that avoid failed edges or vertices.

A traditional objective in shortest path research is to reduce the size of the distance representation. One common way to do so is to use sparse graph *spanners*, that is a spanning subgraph of the original graph using possibly few edges while preserving some distance information. In the context of fault tolerance, Peleg and Parter [24] introduced the notion of *FT-BFS trees*, namely sparse subgraphs that contain a collection of all replacement paths from a given source s that avoids a single edge or vertex in the graph. For an n -vertex unweighted graph $G = (V(G), E(G))$, [24] showed a simple construction of FT-BFS subgraphs with $O(n^{3/2})$ edges. For the case of multiple sources $S \subseteq V$, they showed the construction of an FT-BFS for each $s \in S$ with $O(n^{3/2}\sqrt{|S|})$ edges. Albeit being optimal in space, FT-BFS structures $H \subseteq G$ are lacking some useful properties such as fast reporting of $s - t$ distances in $G - e = (V(G), E(G) \setminus \{e\})$ or being able to route messages along the replacement paths. For instance, to return the distance between the source vertex s and any other vertex t of the graph, following a failure of e , the best one can do with FT-BFS structure is to run a Dijkstra's algorithm in $H - e$ rather than $G - e$.

Our goal in this paper is to devise more structured representations of replacement paths that have useful applications in communication networks.¹ We present efficient constructions of data structures that enjoy not only optimal space (like FT-BFS subgraphs) but also have additional desired attributes, e.g., allowing fast extraction of distances; balanced information spreading in the network; and routing on replacement paths using small routing tables.

In principle, storing the replacement paths in data structures might be more space efficient than using a subgraph of the original network. Unfortunately, here this is not the case; by using standard tools [11, 22, 1], one can show that the lower bound of $\Omega(n^{3/2}\sqrt{|S|})$ edges for FT-BFS structures for $S \times V$ distances applies against *any* kind of replacement paths representation, and not just subgraphs. Our starting point is:

There are bad n -vertex graph families, for which any representation allowing for the return of all the $S \times V$ post-failure distances must have size $\Omega(n^{3/2}\sqrt{|S|})$ bits.

¹ We focus on single edge failures and undirected graphs, although most of the results extend to single vertex failures and directed graphs as well.

Unlike the sourcewise scenario that has been studied thoroughly in the subgraph setting, almost nothing is known for the more general $S \times T$ case, i.e., where T is not necessarily V , but for the fault-free framework [20, 15]. We fill some of that gap here and provide tools that go beyond the sourcewise setting.

1.2 Contribution

We provide a comprehensive study of several space aspects for replacement paths. We consider three fundamental data structures for maintaining shortest paths: distance sensitivity oracles, labeling schemes and compact routing schemes. Roughly speaking, *distance sensitivity oracle* is a compact data structure that can also report distances fast; *labeling scheme* is a more structured type of distance sensitivity oracles in which (hopefully) the same amount of distance information is now spread evenly in the network and hence the memory load per vertex is bounded; Finally a *compact routing scheme* is a distributed algorithm that sends messages from s to t along some short path. The next hop is computed by using the information at the message headers as well as the routing table stored at the current vertex.

Distance Sensitivity Oracles (DSO) and Labeling Schemes

For an n -vertex unweighted graph $G = (V(G), E(G))$, subsets $S, T \subseteq V$, an $S \times T$ DSO is a compact data structure that answers efficiently queries of the form (s, t, e) : *Return the distance between $s \in S$ and $t \in T$ when the edge e fails.* Our main results are the following:

- A polynomial time constructable $S \times T$ DSO of size $\tilde{O}(n\sqrt{|S||T|})$ and query time $O(\sqrt{|S||T|})$. If *quasi-polynomial* query time is allowed, then the size of such oracle can be improved to $\tilde{O}(n|S| + |T|\sqrt{|S|n})$, which we will show to be optimal for $|T| = \Omega(\sqrt{n|S|})$. Moreover, when $|T| = \Omega(n^{\frac{3}{4}}|S|^{\frac{1}{4}})$, the construction can be further refined in order to get a polynomial query time.
- A polynomial time constructable family of *approximate* $S \times T$ DSOs, returning in constant time a distance stretched by an *additive* term which decreases as soon as the size of the oracle increases. In particular, for $|S| = O(\sqrt{n})$, we can obtain an oracle of size $\tilde{O}(n^{3/2})$ and additive distortion $\tilde{O}(\sqrt{n})$.
- A *path-reporting* DSO of size $\tilde{O}(n^{4/3}(|S||T|)^{1/3})$ returning $\pi_{G-e}(s, t)$ in $O(|\pi_{G-e}(s, t)| + (n|S||T|)^{1/3})$ time. Such a DSO can be transformed into an $S \times T$ *fault-tolerant preserver* (FTP) (i.e., a subgraph of G maintaining all the $S \times T$ shortest paths after any edge failure) having the same size. Thus, our FTP improves over the multi-source preserver provided in [24] as soon as $|T| = O(\sqrt{|S|n})$ (up to inverse poly-logarithmic factors). Finally, for the remarkable case in which $T = V(G)$, it can be elaborated in order to get a DSO having size $\tilde{O}(n\sqrt{n|S|})$ and reporting $\pi_{G-e}(s, t)$ in $O(|\pi_{G-e}(s, t)|)$ time; this construction represents the oracle counterpart of the multi-source preserver provided in [24], and thus it has not only optimal size (up to a poly-logarithmic factor), but it also allows to retrieve a shortest path in optimal time.
- Let $\epsilon \in (0, 1]$ be any fixed constant; we show that conditioned on the Set-Intersection Conjecture [26], any $\{s\} \times V$ DSO with constant query time and additive distortion $d = O(n^{1-\epsilon})$ must use $\tilde{\Omega}(n^{\frac{3}{2}\epsilon})$ bits of memory.

Concerning the first result, our construction in fact gives a tradeoff between the query time and the size of the oracle. Note that prior to our construction, for the single-source setting, a trivial query time was $O(n^{3/2})$ by running Dijkstra on the FT-BFS structure with $O(n^{3/2})$ edges. For the $S \times T$ setting, the trivial query time was $O(\sqrt{|S|}n^{3/2})$, using the FT-BFS construction of [24] for multiple sources S .

Concerning the lower bound for the single-source setting, it compares favorably with the non-conditional lower-bounds given in [25]. Indeed, it improves the range of additive distortions for which no linear-size $\{s\} \times V$ DSO can exist from $d = O(\log n)$ to $d = O(n^{\frac{1}{3}-\epsilon})$, for any constant $\epsilon > 0$. Moreover, it shows that for any $d = O(1)$, $\tilde{\Omega}(n^{\frac{3}{2}})$ bits are needed by any $\{s\} \times V$ DSO with constant query time. This is in contrast with the 4-additive FT-BFS structure of size $O(n^{\frac{4}{3}})$ given in [25] thus establishing that designing a corresponding oracle is harder than its FT-BFS counterpart. Notice also that for exact distances (i.e., $d = 0$) this lower bound still allows for the existence of a (single-source) DSO having size $O(n^{\frac{3}{2}})$ and constant query time. We regard the problem of finding the best query time for an optimal-size DSO as an interesting remaining open problem. Due to space limitations the discussion of our lower bound, as well as the proof of several statements, will be provided in the full version of the paper.

Single-Source Labeling Schemes. Labeling schemes are special type of a “balanced” distance oracle with the benefit of having the distance information evenly distributed between all the nodes in the network. Here we obtain a space-optimal (up to constant factors) labeling scheme for the meaningful *single-source to all-destinations* case. It consists of a label with $\tilde{O}(\sqrt{n})$ bits for each node, which allows to compute $|\pi_{G-e}(s, t)|$, by simply looking at the label of t and of the end-vertices of the failing edge e .

Single-Source Routing Scheme

A routing scheme for a given source s is a distributed mechanism that, for the failure of any edge $e \in E$, can deliver packets of information from s to any other node t of the network along the corresponding replacement path. This is done by storing compact *routing tables* at each node, by assigning labels to edges, and finally by adding a short header to the message containing information about the target t and the failing edge e . Our key observation is that every replacement path can be decomposed into two (fault-free) *tree* paths connected by an edge, as shown in [21]. By combining the routing schemes for trees of Thorup and Zwick [29] along with our labeling scheme, we can provide the following:

- A scheme for routing packets from a source s along shortest paths with poly-logarithmic headers and $\tilde{O}(\sqrt{n})$ -size routing tables and edge labels.

1.3 Additional Related Work

In this work, we mainly consider exact distances under faults. In the literature, many related settings have been studied thoroughly as discussed next.

Single source approximate shortest paths avoiding any failed vertex. Baswana and Khanna [3] showed that for the undirected unweighted graph $G = (V, E)$, one can construct a subgraph H with $O(n \log n / \epsilon^3)$ edges satisfying that $\text{dist}(s, t, H - e) \leq (1 + \epsilon) \text{dist}(s, t, G - e)$ for every $t \in V(G), e \in E(G)$. They also provide a DSO of the same size that can report these distances or even the paths in optimal time. This was later extended to the weighted case, for both the subgraph [7] and the oracle setting [8]. *Multiple faults* have been studied in [9, 25] and structures with *additive* stretch have been studied in [23, 6].

Distance sensitivity oracles (for all pairs). In a seminal work, Demetrescu et al. [17] showed that given a directed weighted graph G of size n , it is possible to construct in time $\tilde{O}(mn^2)$ a DSO of size $O(n^2 \log n)$ capable of answering distance queries in $O(1)$ time in

the presence of a single failed edge or vertex. The preprocessing time was then improved to $\tilde{O}(mn)$, with unchanged size and query time [5]. Grandoni and Williams [19] presented the first DSO that achieves simultaneously subcubic preprocessing time and sublinear query time for directed graphs with bounded integer edge weights. A dual failure fault tolerant DSO of size $O(n^2 \log^3 n)$ and $O(\log n)$ query time was presented in [18]. The f faults case was studied in [30, 13].

FT distance labels and compact routing schemes. Label-based fault-tolerant routing schemes for graphs of bounded clique-width are presented in [16]. To route from s to t , the source needs to specify the labels $\lambda(s)$ and $\lambda(t)$ and the set of failures F , and the scheme efficiently calculates the shortest path between s and t that avoids F . For an n -vertex graph of tree-width or clique-width k , the constructed labels are of size $O(k^2 \log^2 n)$. Turning to general graphs, FT compact routing schemes were first considered in [14], for up to two edge failures. Further work considered multiple failures [12] and $(1 + \epsilon)$ approximation [2].

Set intersection and distance oracles. The set intersection problem has several related variants and has been widely used to provide conditional lower bounds on the space and query time of distance oracles. The folklore conjecture for set intersection states that, given n sets of cardinality polylogarithmic in n , answering a set intersection query in constant time requires $\Omega(n^2)$ space. For the connection between distance oracles and various variants of the set intersection problem, see [28, 26, 27]. In this paper we provide the first connection between distance *sensitivity* oracles and the set intersection problem.

2 Preliminaries and Notations

Let $G = (V(G), E(G))$ be a directed or undirected graph on n vertices with $S \subseteq V(G)$ as the source set and $T \subseteq V(G)$ as the destination set. Let H be a subgraph of G . We use H^R to denote the graph obtained by reversing all edge directions of H (if H is undirected then H^R is same as H). For any vertex w , let $\mathcal{T}_{w,H}$ be the shortest path tree of H rooted at w , and $\mathcal{T}_{w,H}^R$ be the shortest path tree of H^R rooted at w . When H is same as G , we can as well use the notions \mathcal{T}_w and \mathcal{T}_w^R . We will denote by $\pi_H(u, v)$ the shortest path between the two vertices u and v in H , and by $d_H(u, v)$ its length, i.e., the distance between u and v in H . Moreover, whenever $H = G$, we will omit the subscript. Given a set $F \subseteq E(G)$ of edges, we will denote by $G - F$ the subgraph of G obtained by removing the edges in F from $E(G)$. For the sake of simplicity we might slightly abuse the notation and write $G - e$ instead of $G - \{e\}$ when $F = \{e\}$. Given a simple path P , we denote by $|P|$ its *size*, i.e., the number of its edges. Moreover, if P traverses the vertices u and v in this order, we denote by $P(u, v)$ the subpath of P between u and v (endpoints included). For any non-negative integer i , we define $P[-i]$ to be the path containing the last $\min\{|P|, i\}$ edges of P . Given any two paths P and Q with last vertex of P same as the first vertex of Q , we use $P::Q$ to denote the path formed by concatenating paths P and Q .

Given a tree \mathcal{T} and any two vertices $a, b \in \mathcal{T}$, we use the notation $\mathcal{T}(a, b)$ to denote the path from a to b in tree \mathcal{T} . Throughout the paper we use $\tilde{O}(f(x))$ (resp. $\tilde{\Omega}(f(x))$) as a shorthand for $O(f(x) \text{ polylog} f(x))$ (resp. $\Omega(f(x)/\text{polylog} f(x))$). Below we state a lemma that will be crucially used in our fault tolerant data structures.

► **Lemma 1.** *Let G be an undirected unweighted graph, and let $L \in [5, n/\log n]$ and $\mathcal{P} = \{\pi(u, v) \mid u, v \in V(G), d(u, v) \geq L \log n\}$ be the family of shortest paths in G having length at least $L \log n$. Then (i) In expected polynomial time we can compute a subset R of $V(G)$*

with $O(n/L)$ vertices such that $R \cap V(P) \neq \emptyset$ for each path $P \in \mathcal{P}$; (ii) We can also have a deterministic polynomial time construction for set R that intersects each path in \mathcal{P} , such an R contains $O(\frac{n}{L} \log n)$ vertices.

Although Lemma 1 allows for both randomized and deterministic constructions, in the rest of the paper, we will only focus on the randomized case, as however this will only differ up to logarithmic factors in the query time and the size of our solutions.

We assume edge weights are slightly perturbed by adding a small noise so that edge-weights are always positive and between any two vertices x, y there is exactly one shortest path. This will help us to uniquely define $\pi_H(x, y)$ for any subgraph H of G . When we focus on undirected graphs, we assume perturbation is small enough so that for any simple path P between x and y of weighted length λ , we have $|P| = \lfloor \lambda \rfloor$.

3 Distance Sensitivity Oracle

The basic building block in our construction is an $W \times W$ DSO that reports, in $O(1)$ time, the distance between any pair of vertices in $W \subseteq V(G)$. This will be used to obtain our $S \times T$ oracle.

3.1 Distance Sensitivity Oracle for $W \times W$

As an input we are given a set W of vertices in a directed or undirected *weighted* graph G . We will use ideas similar to the ones of the edge/vertex fault tolerant $V \times V$ oracle of [17]. For the sake of simplicity we only discuss the edge-failure case, but our results naturally extend to the vertex failures as well. Our data structure stores the following information:

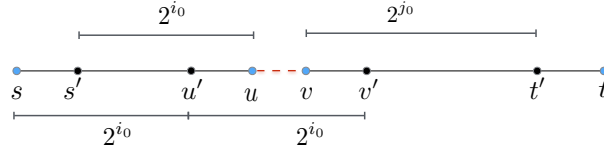
1. For each $w \in W$, it stores:
 - An incoming and an outgoing shortest path tree rooted at w , i.e. trees \mathcal{T}_w and \mathcal{T}_w^R ;
 - The pre-order and post-order numbering, and depth of each $v \in V$ in \mathcal{T}_w and \mathcal{T}_w^R ;
 - A level ancestor data structure for trees \mathcal{T}_w and \mathcal{T}_w^R , namely a data structure able to return in $O(1)$ time the k -th ancestor of a node, for any $k \geq 1$ [4];
 - The distances $d(w, v)$ and $d(v, w)$, where $v \in V$.
2. For every vertex pair $(s, t) \in (W \times V) \cup (V \times W)$ and every integer $0 \leq i \leq \log |\pi(s, t)|$:
 - B1(s, t, i) stores the distance $d_{G-e}(s, t)$, where $e = (u, v)$ is an edge lying on $\pi(s, t)$ and satisfying $|\pi(s, u)| = 2^i$;
 - B2(s, t, i) stores the distance $d_{G-\pi(u, v)}(s, t)$, where u, v are vertices on $\pi(s, t)$ and satisfying (i) $|\pi(u, v)| = 2^i$, and (ii) $|\pi(s, u)| = 2^i$.

We now explain the query process. Let $(s, t) \in W \times W$ be a query pair and $e = (u, v)$ be a failing edge lying on $\pi(s, t)$. (Whether e lies on $\pi(s, t)$ or not can be verified in constant time using pre-order and post-order numbering, and depth of vertices in \mathcal{T}_w and \mathcal{T}_w^R , $w \in W$). Let i_0 and j_0 be greatest integers satisfying $2^{i_0} \leq |\pi(s, u)|$ and $2^{j_0} \leq |\pi(u, t)|$. Let s', t' be vertices on $\pi(s, t)$ such that $|\pi(s', u)| = 2^{i_0}$ and $|\pi(v, t')| = 2^{j_0}$. (See Figure 1). These vertices can be computed in constant time by using the level ancestor data structure on shortest path trees \mathcal{T}_w and \mathcal{T}_w^R .

Let P be an $s - t$ shortest path in $G - e$. We have the following two cases.

- 1 P passes through either s' or t' :

If P passes through t' , then $d_{G-e}(s, t) = d_{G-e}(s, t') + d(t', t)$, and if P passes through s' , then $d_{G-e}(s, t) = d(s, s') + d_{G-e}(s', t)$. So in this case we can use B1 to report the distance between s and t in $G - e$.



■ **Figure 1** Depiction of vertices s', u', v', t' when the failing edge $e = (u, v)$ lies on path $\pi(s, t)$.

2 P does not pass through s' and t' :

Let us assume that $i_0 \leq j_0$. (If $j_0 < i_0$ then a similar analysis will follow). Let u', v' be vertices on $\pi(s, t)$ such that $|\pi(s, u')| = |\pi(u', v')| = 2^{i_0}$. Since $2^{i_0} \leq 2^{j_0}$, we have $u' \in \pi(s', u)$ and $v' \in \pi(v, t')$. Thus P does not pass through segment $\pi(u', v')$, i.e., $\pi_{G-e}(s, t) = \pi_{G-\pi(u', v')}(s, t)$. So in this case, we can use B2 to report $d_{G-e}(s, t)$.

The space and the query time of our data structure are summarized by the following theorem:

► **Theorem 2.** *An n -vertex directed or undirected weighted graph G for a given set $W \subseteq V(G)$ can be preprocessed in polynomial time to compute a data structure of $O(n|W| \log n)$ size that given any two vertices $s, t \in W$ and any failing edge e can report $d_{G-e}(s, t)$ in constant time. Our result also holds for single vertex failure.*

3.2 Distance Sensitivity $S \times T$ Oracle

In the following we assume that G is a directed or undirected unweighted graph. Also we assume $|S| \leq |T|$, as otherwise we could consider G^R instead and swap the roles of S and T . Let $L \geq 5$ be a parameter in $[n/\sqrt{|S||T|}, n/\log n]$ and let $R \subseteq V(G)$ be a set of size $O(n/L)$ as obtained from Lemma 1. Also let ℓ be $\lceil L \log n \rceil$. Our construction is a simple two step process:

1. Set $W = S \cup R$, and compute the $W \times W$ oracle of Section 3.1 over set W .
2. For each pair $(s, t) \in S \times T$, if $e_1, e_2, \dots, e_{\min\{\ell, |\pi(s, t)|\}}$ are the edges on $\pi(s, t)[- \ell]$ listed in reverse order (i.e., from t towards s), then store in $d_{(s, t)}^{-i}$ the distance $d_{G-e_i}(s, t)$.

► **Lemma 3.** *Let $e = (u, v)$ be an edge lying on $\pi(s, t)$ for some vertices $s, t \in V(G)$. Also let $x \in V(G)$ be such that $d(x, t) \leq d(v, t)$. Then $e \notin \pi(x, t)$, and so $d_{G-e}(x, t) = d(x, t)$.*

Proof. Let us assume on the contrary that $\pi(x, t)$ traverses e , and let u' (resp. v') be the first (resp. last) vertex in $\{u, v\}$ it encounters. Then

$$d(x, t) = d(x, u') + 1 + d(v', t) \geq 1 + d(v, t) > d(v, t).$$

However, by our hypothesis $d(x, t) \leq d(v, t)$. Hence, we get a contradiction. ◀

► **Lemma 4.** *Let $e = (u, v)$ be an edge lying on $\pi(s, t)$ for some vertices $s, t \in V(G)$. Also assume $e \notin \pi(s, t)[- \ell]$, then $d_{G-e}(s, t) = \min_{x \in R, d(x, t) \leq \ell} (d_{G-e}(s, x) + d(x, t))$.*

Proof. Let P be the shortest path from s to t in $G-e$. Since $|P| \geq d(s, t) > \ell$, by Lemma 1 and sub-optimality of shortest paths, the path $P[- \ell]$ must contain at least one vertex from set R , let this be r . Consider the path $P(r, t) = \pi_{G-e}(r, t)$. Since $d(r, t) \leq |\pi_{G-e}(r, t)| \leq \ell \leq d(v, t)$, Lemma 3 implies that $d_{G-e}(r, t)$ is equal to $d(r, t)$. Therefore we have:

$$d_{G-e}(s, t) = d_{G-e}(s, r) + d_{G-e}(r, t) = d_{G-e}(s, r) + d(r, t).$$

Algorithm 1: Compute $d_{G-e}(s, t)$ where $e = (u, v)$ is a failing edge on $\pi(s, t)$.

1 **if** $(i \leq \ell)$ **then return** $d_{(s,t)}^{-i} = d_{G-e}(s, t)$
2 **else return** $\min_{x \in R, d(x,t) \leq \ell} (d_{G-e}(s, x) + d(x, t))$

Also notice that for any $r_0 \in R$, if $d(r_0, t) \leq \ell$, then by Lemma 3, $d(r_0, t) = d_{G-e}(r_0, t)$, as $d(v, t) \geq \ell$. Thus $d_{G-e}(s, r_0) + d(r_0, t) = d_{G-e}(s, r_0) + d_{G-e}(r_0, t) \geq d_{G-e}(s, t)$. So from above discussion it follows that $d_{G-e}(s, t) = \min_{x \in R, d(x,t) \leq \ell} (d_{G-e}(s, x) + d(x, t))$. ◀

Query algorithm. Consider a pair (s, t) , let $e = (u, v)$ be a failing edge lying on $\pi(s, t)$. (As before whether e belongs to $\pi(s, t)$ can be verified in constant time). If $\pi(s, t)[- \ell]$ contains e , then we can output new distance in $O(1)$ time. If $\pi(s, t)[- \ell]$ does not contain e , then by Lemma 4, $d_{G-e}(s, t) = \min\{d_{G-e}(s, r) + d(r, t) | r \in R, d(r, t) \leq \ell\}$. In this equation the distance $d_{G-e}(s, r)$ for any $s \in S$ and $r \in R$ can be computed in constant time using the data structure of the previous subsection. Since the values $d(r, t)$ are pre-stored, the query time is $O(|R|) = O(n/L)$. Algorithm 1 presents the pseudocode of our implementation. Notice that the space used is $O(n|W| \log n + |S||T|\ell) = O((n^2/L + |S||T|L) \log n)$ which, due to our choice of L , is $O(|S||T|L \log n)$. We hence obtain the following result:

► **Theorem 5.** *An n -vertex (directed or undirected) unweighted graph G for a given source set $S \subseteq V(G)$ and destination set $T \subseteq V(G)$ can be preprocessed in polynomial time to compute a DSO of size $O(|S||T|L \log n)$ and query time $O(n/L)$, where $L \in [n/\sqrt{|S||T|}, n/\log n]$.*

Notice that the subgraph lower bound of $\Omega(n\sqrt{n|S|})$ provided in [24] holds also for the oracles setting (by using standard information theoretic arguments). Therefore, by choosing $L = \Theta(n/\sqrt{|S||T|})$ in Theorem 5, we obtain an oracle of size $O(n\sqrt{|S||T|} \log n)$ and query time $O(\sqrt{|S||T|})$ which, for $|T| = \Theta(n)$, has optimal size (up to the poly-logarithmic factors).

3.3 Space-Improved $S \times T$ Oracle

Recall that in last subsection we computed an $S \times T$ oracle with $O(n\sqrt{|S||T|} \log n)$ space and $O(\sqrt{|S||T|})$ query time using a random sample of vertices R . In this section, we obtain an oracle with an improved size at the expense of higher (quasi-polynomial) query time. In particular, this oracle has the optimal size for $|T| = \Omega(\sqrt{|S|n})$. Our main idea is to use a hierarchy of random sets $R_1, R_2, \dots, R_\alpha$ for an appropriate α .

We now explain our construction. Let α be integer to be fixed later on, and for $i \in [0, \alpha]$, let L_i be $(n/|S|)^{\frac{2\alpha-i}{2\alpha}}$ and R_i be random set of size $O(n/L_i) = O(|S|^{\frac{2\alpha-i}{2\alpha}} n^{\frac{i}{2\alpha}})$ computed using Lemma 1. For each $i \geq 0$, we will compute an oracle for $S \times R_i$, say $\mathcal{O}_{S \times R_i}$. Also we use $\mathcal{O}_{S \times T}$ to denote our oracle for the product $S \times T$. Since $|R_0| = O(|S|)$, we use Theorem 2 to compute an oracle for $S \times R_0$ with $O(n|S| \log n)$ space and $O(1)$ query time. It turns out that for any $i > 1$, our oracle $\mathcal{O}_{S \times R_i}$ uses $\mathcal{O}_{S \times R_{i-1}}$, and $\mathcal{O}_{S \times T}$ uses $\mathcal{O}_{S \times R_\alpha}$.

For sake of convenience let $R_{\alpha+1} = T$. For an oracle \mathcal{O} , let $\mathbf{size}(\mathcal{O})$ be the size of the oracle and let $\mathbf{time}(\mathcal{O})$ be its query time. For any $i \in [0, \alpha]$, we compute the $\mathcal{O}_{S \times R_{i+1}}$ oracle from $\mathcal{O}_{S \times R_i}$ as follows. We first compute oracle $\mathcal{O}_{S \times R_i}$, this is augmented by storing

1. $d(x, y)$ for $(x, y) \in R_i \times R_{i+1}$, and
2. $d_{G-e}(s, y)$ for $(s, y) \in S \times R_{i+1}$, $e \in \pi(s, y)[-L_i \log n]$.

So, we have: $\mathbf{size}(\mathcal{O}_{S \times R_{i+1}}) = \mathbf{size}(\mathcal{O}_{S \times R_i}) + (|R_i||R_{i+1}| + |S||R_{i+1}|L_i \log n)$.

For any $y \in R_{i+1}$, to report the distance $d_{G-e}(s, y)$ we proceed in a similar way as in Algorithm 1. If $\pi(s, y)[-L_i \log n]$ contains e we return the stored distance. Otherwise we compute $d_{G-e}(s, y)$ as $\min_{x \in R_i, d(x, y) \leq L_i \log n} (d_{G-e}(s, x) + d(x, y))$, where $d_{G-e}(s, x)$ is obtained by querying $\mathcal{O}_{S \times R_i}$. Since at most $|R_i|$ queries to $\mathcal{O}_{S \times R_i}$ are performed, we have $\mathbf{time}(\mathcal{O}_{S \times R_{i+1}}) = \mathbf{time}(\mathcal{O}_{S \times R_i}) \times |R_i|$. Summing the first equation from $i = 0$ to α , and substituting $\alpha = \log n - 1$, we obtain the size of oracle $\mathcal{O}_{S \times T}$:

$$\begin{aligned} \mathbf{size}(\mathcal{O}_{S \times T}) &= \mathbf{size}(\mathcal{O}_{S \times R_0}) + \sum_{i=0}^{\alpha} (|R_i| |R_{i+1}| + |S| |R_{i+1}| L_i \log n) \\ &\leq n |S| \log n + \alpha |R_\alpha|^2 + |R_\alpha| |T| + \sum_{i=0}^{\alpha-1} (|S| |R_{i+1}| L_i \log n) + |S| |T| L_\alpha \log n \\ &= (\alpha + \log n) n |S| + |T| \sqrt{n |S|} + \sum_{i=0}^{\alpha-1} (n |S| (n/|S|)^{\frac{1}{2^i}} \log n) + |T| \sqrt{n |S|} \log n \\ &= O(n |S| \log^2 n + |T| \sqrt{n |S|} \log n). \end{aligned}$$

Turning to query time, we get that $\mathbf{time}(\mathcal{O}_{S \times T}) = O(1) \cdot \prod_{i=0}^{\alpha} |R_i| = (n |S|)^{\frac{\alpha+1}{2}} = O((n |S|)^{\frac{\log n}{2}})$.

The following theorem follows from above discussion.

► **Theorem 6.** *An n -vertex (directed or undirected) unweighted graph G for a given source set $S \subseteq V(G)$ and destination set $T \subseteq V(G)$ can be preprocessed in polynomial time to compute a DSO with $O(n |S| \log^2 n + |T| \sqrt{n |S|} \log n)$ size and $O((n |S|)^{\frac{\log n}{2}})$ query time.*

Notice that the subgraph lower bound of $\Omega(n \sqrt{n |S|})$ provided in [24] can be easily adapted to yield an $\Omega(|T| \sqrt{n |S|})$ lower bound for the $S \times T$ case (oracles setting included), and the details will be given in the full version of the paper. Therefore, for $|T| = \Omega(\sqrt{n |S|})$, the oracle of Theorem 6 has optimal size (up to poly-logarithmic factors).

We next show that for the special case of $|T| = \Omega(n^{\frac{3}{4}} |S|^{\frac{1}{4}})$, we can obtain an even better space-optimal oracle (up to logarithmic factors) having *polynomial* query time.

► **Theorem 7.** *An n -vertex (directed or undirected) unweighted graph G for a given source set $S \subseteq V(G)$ and destination set $T \subseteq V(G)$ satisfying the condition $|T| = \Omega(n^{\frac{3}{4}} |S|^{\frac{1}{4}})$ can be preprocessed in polynomial time to compute a DSO of size $O(T \sqrt{n |S|} \log n)$ and query time $O(n^{\frac{3}{2}} |S|^{\frac{3}{2}} |T|^{-1}) = O(n^{\frac{3}{4}} |S|^{\frac{5}{4}})$.*

Proof. Let L be a parameter and R be a random set of size $O(n/L)$ computed by Lemma 1. Our oracle consists of an $S \times R$ oracle $\mathcal{O}_{S \times R}$ of size $O(|S| |R| L_0 \log n)$ and query time $O(n/L_0)$, for some parameter $L_0 \in [n/\sqrt{|S| |R|}, n/\log n]$ (see Theorem 5). This is augmented by storing (i) $d(x, t)$ for $x \in R, t \in T$, and (ii) the distance $d_{G-e}(s, t)$ for $s \in S, t \in T, e \in \pi(s, t)[-l]$ (recall that $l = \lceil L \log n \rceil$). The overall size is $O(|S| |R| L_0 \log n + |R| |T| + |S| |T| L \log n)$.

To report $d_{G-e}(s, t)$ we proceed in a similar way discussed in Algorithm 1. If $e \in \pi(s, t)[-l]$ we return the stored distance. Otherwise we compute $d_{G-e}(s, t)$ as $\min_{x \in R, d(x, t) \leq l} (d_{G-e}(s, x) + d(x, t))$, where $d_{G-e}(s, x)$ is obtained by querying $\mathcal{O}_{S \times R}$. Since $O(|R|)$ queries to $\mathcal{O}_{S \times R}$ are performed, the total query time is $O(|R| (n/L_0))$.

Now we get our result by substituting $|R|$ as $\Theta(n/L)$, and picking $L = \sqrt{n/|S|}$ and $L_0 = |T|/|S|$ to obtain an oracle of size $O(T \sqrt{n |S|} \log n)$, and query time $O((n |S|)^{\frac{3}{2}} / |T|)$. ◀

3.4 $S \times T$ Oracle with Additive Distortion

We conclude this section by focusing on the case in which an additive distortion to the reported distance is allowed. The following theorem provides a family of oracles whose additive stretch

decreases as soon as the size increases, regardless of the number of destinations.

► **Theorem 8.** *Let $L \in [5, n/|S|]$ and G be an undirected unweighted graph with S as source set and T as destination set, and assume w.l.o.g. that $|S| \leq |T|$. Then, there exists a polynomial-time constructible $(2L \log n)$ -additive DSO of $O((n^2/L) \cdot \log n)$ size and $O(1)$ query time.*

For the prominent single-source case, the above result is complemented by the following conditional lower bound:

► **Theorem 9.** *Let $\epsilon \in (0, 1]$ be any fixed constant. If the Set-Intersection Conjecture holds, then any single-source DSO with constant query time and additive distortion $d = O(n^{1-\epsilon})$ must use $\tilde{\Omega}(n^{\frac{3}{2}\epsilon})$ bits of memory.*

The above result improves several (unconditional) lower bounds on fault-tolerant additive-distortion single-source structures. Moreover, we have recently extended the above lower bound to the case of multiple sources, and we will provide it in the full version of the paper.

4 Path-Reporting $S \times T$ Oracle and Fault-Tolerant Preservers

The following lemma shows that the shortest paths have a very nice structure in the graph $G - e$. (Since all our results in this section will crucially use this lemma, the results in this section will hold for undirected graphs and edge failures only.)

► **Lemma 10** (also proved in [21, 10]). *Let G be an undirected weighted graph, $s, t \in V(G)$ and $e \in \pi(s, t)$ such that s and t are connected in $G - e$. There exists an edge $(y, z) \in G - e$ such that $\pi(s, y) \cup \pi(y, z) \cup \pi(z, t)$ is a shortest path in $G - e$. We will refer to (y, z) by $\text{LINK}(s, t, e)$.*

For $W \times W$, the above lemma implies the following:

► **Theorem 11.** *An n -vertex undirected weighted graph G for a given set $W \subseteq V(G)$ can be preprocessed in polynomial time to compute a data structure of $O(n|W| \log n)$ size that given any two vertices $s, t \in W$ and any failing edge $e \in E(G)$, can report $\pi_{G-e}(s, t)$ in $O(|\pi_{G-e}(s, t)|)$ time.*

Moreover, as a by-product we get a sparse subgraph with $O(n|W| \log n)$ edges that preserves distance between any vertex pair $(s, t) \in W \times W$ after single edge failure e .

The above construction can be used to design a path-reporting oracle for $S \times T$, for the unweighted case only though. As before, for a parameter L we take a random set R with $O(n/L)$ vertices. We pre-compute the path reporting oracle for $W \times W$, where $W = S \cup R$, and also the distance oracle for $S \times T$. Recall that this will take $O((n^2/L + |S||T|L) \log n)$ space. Next, for each $(s, t) \in S \times T$ and $e \in \pi(s, t)[-l]$, we store the following: (i) edge $(y, z) = \text{LINK}(s, t, e)$, (ii) the distance $d(z, t)$, and (iii) the suffix $\pi_{G-e}(s, t)[-l]$.

Notice that the total space used by us is $O((n^2/L) \log n + |S||T|L^2 \log^2 n)$. On choosing L as $n^{2/3}(|S||T|)^{-1/3}$, we get a bound of $O(n^{4/3}(|S||T|)^{1/3} \log^2 n)$. Then, we use the following:

Path-reporting Query Algorithm

1. If $e \notin \pi(s, t)$, we return $\pi(s, t)$ stored in the shortest path tree \mathcal{T}_s (recall $s \in W$).
2. If $e \in \pi(s, t)[-l]$ then we retrieve the pre-stored edge $(y, z) = \text{LINK}(s, t, e)$, the distance $d(z, t)$, and the path $P = \pi_{G-e}(s, t)[-l]$.
 - If $d(z, t) \leq l$, then z must lie on P and $\pi_{G-e}(s, t)$ will be equal to $\pi(s, y) \cup \pi(y, z) \cup P(z, t)$.

- If $d(z, t) > \ell$, then we compute a vertex $r \in R$ lying on $P = \pi_{G-e}(s, t)[- \ell]$ in $O(|P|) = O(\ell)$ time. Such an r exists by Lemma 1. We output $\pi_{G-e}(s, t) = \pi_{G-e}(s, r) :: \pi(r, t)$. Notice that in this case $\pi_{G-e}(s, t)$ can be outputted in $O(|\pi_{G-e}(s, t)|)$ time.
- 3. If $e \notin \pi(s, t)[- \ell]$, then it follows from Lemma 4 that $\pi_{G-e}(s, t) = \pi_{G-e}(s, r) :: \pi(r, t)$, where $r = \arg \min (d_{G-e}(s, x) + d(x, t) \mid x \in R, d(x, t) \leq \ell)$. Such an r is computable in $O(|R|) = O(n^{1/3}|S|^{1/3}|T|^{1/3})$ time. Thus in this case in $O(|\pi_{G-e}(s, t)| + (n|S||T|)^{1/3})$ time we can report $\pi_{G-e}(s, t)$.

The above analysis thus implies the following result:

► **Theorem 12.** *For any undirected unweighted graph G there exists a polynomial-time constructible DSO for a source set S and destination set T of size $O(n^{4/3}(|S||T|)^{1/3} \log^2 n)$ that for any $(s, t) \in S \times T$, and any failing edge $e \in E(G)$, can report $\pi_{G-e}(s, t)$ in $O(|\pi_{G-e}(s, t)| + (n|S||T|)^{1/3})$ time.*

Moreover, as a by-product we get a sparse subgraph with $O(n^{4/3}|S|^{1/3}|T|^{1/3} \log^2 n)$ edges that preserves distance between any vertex pair $(s, t) \in S \times T$ after single edge failure e .

Finally, we conclude this section by providing an even better oracle for the meaningful scenario in which $T = V(G)$. As before we take a set R with $\sqrt{|S||T|}$ vertices and compute the path reporting oracle for $W \times W$, where $W = S \cup R$. We also pre-compute a distance oracle for $S \times T$. For each $t \in V(G)$, and each $e \in \pi(s, t)[- \ell]$, we store the last edge of $\pi_{G-e}(s, t)$. Notice that the overall size of the oracle remains same, i.e. $O(n\sqrt{|S||T|} \log n)$.

A path query is performed as follows: if $e \notin \pi(s, t)$, we return $\pi(s, t)$ stored in the shortest path tree \mathcal{T}_s (recall $s \in W$); if e appears on $\pi(s, t)[- \ell]$, we can access the last edge, say (w, t) , of $P = \pi_{G-e}(s, t)$ in constant time and obtain path $P[s, w] = \pi_{G-e}(s, w)$ by recursively querying the oracle. Finally, in the remaining case, we compute in $O(|R|)$ time the vertex $r = \arg \min \{d_{G-e}(s, x) + d(x, t) \mid x \in R, d(x, t) \leq \ell\}$. We know that the concatenation $\pi_{G-e}(s, r) :: \pi(r, t)$ is a shortest path from s to t in $G - e$. Notice that using Theorem 11, $\pi_{G-e}(s, r)$ can be reported in $O(|\pi_{G-e}(s, r)|)$ time, also the path $\pi(r, t)$ is stored in the tree \mathcal{T}_r (recall $r \in W$). Thus the time for reporting path $\pi_{G-e}(s, t)$ is $O(|R| + |\pi_{G-e}(s, t)|)$. Notice that $|R| = \sqrt{n|S|}$ and $|\pi_{G-e}(s, t)| \geq d(s, t) \geq \ell = (n/|R|) \log n = \sqrt{n/|S|} \log n$. Therefore in this case as well, the total time spent is of order of the number of edges on the shortest path $\pi_{G-e}(s, t)$. To summarize, we have:

► **Theorem 13.** *For any undirected unweighted graph G there exists a polynomial-time constructible DSO for a source set S of size $O(n\sqrt{n|S|} \log n)$ that for any $s \in S, t \in V(G)$, and any failing edge $e \in E(G)$, can report $\pi_{G-e}(s, t)$ in $O(|\pi_{G-e}(s, t)|)$ time.*

Remarkably, the above multi-source oracle is the natural counterpart of the multi-source fault-tolerant preserver given in [24], and so it is optimal in space (up to poly-logarithmic factors) and in query time.

5 Distributed Routing Scheme for Single-Source Distances

In this section, we present the main crux of our edge-fault-tolerant routing distributed schemes for the *single-source to all-destinations* case. Details about our labeling scheme and the remaining details of the routing scheme can be found in the full version of the paper. We use s to denote the designated source vertex. For any two vertices a, b , we denote by $\text{TREEPATH}(a, b)$ the path from a to b in tree \mathcal{T}_s .

It turns out that finding our labeling scheme of $\tilde{O}(n^{\frac{3}{2}})$ bitsize is quite easy, while designing our routing scheme is not that straightforward. Our approach for it works as follows. First

we show how to represent the FT-BFS (w.r.t. s) subgraph as a union of trees and link-edges (see Lemma 10) of total size $\tilde{O}(n^{3/2})$, so that each replacement path can be represented as a combination of two tree paths connected by a link edge. We can then use the routing scheme over trees by Thorup and Zwick [29]. For ensuring small routing tables, we will need that each vertex must be present in only $\tilde{O}(\sqrt{n})$ number of trees in the family of trees considered by us. In this way, we will obtain a scheme for routing packets from s along replacement shortest paths with poly-logarithmic headers and $\tilde{O}(\sqrt{n})$ bitsize node routing tables and edge labels.

5.1 Tree Representations

For a parameter $L = \sqrt{n}$ we take $R \subseteq V(G)$ to be a set of vertices as obtained from Lemma 1. Also ℓ is taken to be $\lceil L \log n \rceil$. We define two family of trees \mathcal{T}_{long} and \mathcal{T}_{short} as follows: (i) The family \mathcal{T}_{long} consists of shortest path tree \mathcal{T}_s , and the shortest path trees \mathcal{T}_r for each $r \in R$; (ii) The family \mathcal{T}_{short} consists of all the possible trees $\mathcal{T}_{e,z}$ given by Definition 14 below and have depth at most ℓ .

► **Definition 14.** Let $e = (u, v) \in \mathcal{T}_s$, and (y, z) be an edge that becomes tree edge in $\mathcal{T}_{s, G-e}$. Also assume $d(u, z) \leq 2\ell$. We define $\mathcal{T}_{e,z}$ to be a subtree of $\mathcal{T}_{s, G-e}$ which is (i) rooted at vertex z , and (ii) truncated to depth ℓ , that is, it contains only those vertices whose depth differ from depth of z in $\mathcal{T}_{s, G-e}$ by at most ℓ .

In the following table, we show how the families \mathcal{T}_{long} and \mathcal{T}_{short} can be directly used to obtain a shortest path from s to any arbitrary vertex t after an edge failure on $\text{TREEPATH}(s, t)$.

If	Then	New $s - t$ shortest path in $G - e$
$t = r_0 \in R$		$\text{TREEPATH}(s, y) :: (y, z) :: \mathcal{T}_{r_0}(z, r_0)$ where $(y, z) = \text{LINK}(s, r_0, e)$
$e \in \text{TREEPATH}(s, t)[- \ell]$ $(y, z) = \text{LINK}(s, t, e)$, $d(z, t) \leq \ell$	$t \in \mathcal{T}_{e,z} \in \mathcal{T}_{short}$	$\text{TREEPATH}(s, y) :: (y, z) :: \mathcal{T}_{e,z}(z, t)$
Remaining cases	$\pi_{G-e}(s, t)[- \ell]$ must contain some $r \in R$	$\text{TREEPATH}(s, y_r) :: (y_r, z_r) :: \mathcal{T}_r(z_r, t)$ where $(y_r, z_r) = \text{LINK}(s, r, e)$

We now show correctness of above table case by case.

Case 1. $t = r_0 \in R$

Let $(y, z) = \text{LINK}(s, r_0, e)$. Recall that we showed in Lemma 10, $\pi(s, y) :: (y, z) :: \pi(z, r_0) = \text{TREEPATH}(s, y) :: (y, z) :: \mathcal{T}_{r_0}(z, r_0)$ is a shortest path from s to r_0 in $G - e$. Notice that the trees \mathcal{T}_s and \mathcal{T}_{r_0} are present in the family \mathcal{T}_{long} .

Case 2. $e \in \text{TREEPATH}(s, t)[- \ell]$, $(y, z) = \text{LINK}(s, t, e)$, $d(z, t) \leq \ell$

Since $d(t, z) \leq \ell$, we will have $d(u, z) \leq 2\ell$. This along with the fact that (y, z) becomes a tree edge in $\mathcal{T}_{s, G-e}$ shows that tree $\mathcal{T}_{e,z}$ is present in the family \mathcal{T}_{short} . Also from Lemma 10 we know that e cannot lie on $\pi(z, t)$, so $d_{G-e}(z, t) = d(z, t) \leq \ell$. Since tree $\mathcal{T}_{e,z}$ contains shortest paths up to depth ℓ , vertex t must lie in $\mathcal{T}_{e,z}$. This shows that $\mathcal{T}_{e,z}(z, t) = \pi_{G-e}(z, t) = \pi(z, t)$. So by applying Lemma 10, we get that $\text{TREEPATH}(s, y) :: (y, z) :: \mathcal{T}_{e,z}(z, t)$ is a shortest path from s to t in $G - e$.

Case 3(i). $e \in \text{TREEPATH}(s, t)[- \ell]$, $(y, z) = \text{LINK}(s, t, e)$, $d(z, t) > \ell$

Let $P = \pi(s, y) :: (y, z) :: \pi(z, t)$ be a shortest path from s to t in $G \setminus e$ (see Lemma 10). As $|P(z, t)| \geq \ell$, by Lemma 1, $P[- \ell]$ must contain a vertex from set R , say r . Let $(y_r, z_r) = \text{LINK}(s, r, e)$. Since $P[s, r] = \pi_{G-e}(s, r)$, edge (y_r, z_r) must be identical to the

edge (y, z) . Notice that $\pi(z, t) = \mathcal{T}_r(z, r) :: \mathcal{T}_r(r, t) = \mathcal{T}_r(z, t) = \mathcal{T}_r(z_r, t)$. Thus in this case $\text{TREEPATH}(s, y_r) :: (y_r, z_r) :: \mathcal{T}_r(z_r, t)$ is a shortest path from s to t in the graph $G - e$.

Case 3(ii). $e \notin \text{TREEPATH}(s, t)[-l]$

Let $P = \pi_{G-e}(s, t)$. By Lemma 1, $P[-l]$ must contain a vertex from set R , say r . We know by Lemma 3 that $P[r, t]$ is a shortest path in G , thus $P[r, t] = \mathcal{T}_r(r, t)$. Let $(y_r, z_r) = \text{LINK}(s, r, e)$, then $P[s, r] = \pi_{G-e}(s, r) = \text{TREEPATH}(s, y_r) :: (y_r, z_r) :: \mathcal{T}_r(z_r, r)$. Thus in this case also $\pi_{G-e}(s, t) = \text{TREEPATH}(s, y_r) :: (y_r, z_r) :: \mathcal{T}_r(z_r, t)$. (Notice that if $\text{TREEPATH}(s, r)$ is intact in graph $G - e$, then we can define $\text{LINK}(s, r, e)$ to be any arbitrary edge on $\text{TREEPATH}(s, r)$).

All that remains is to show that each vertex in G appears in $\tilde{O}(\sqrt{n})$ trees in the families \mathcal{T}_{short} and \mathcal{T}_{long} . For the family \mathcal{T}_{long} , proof is trivial because $|\mathcal{T}_{long}| = O(|R|) = O(\sqrt{n})$, and it turns out the same holds for the family \mathcal{T}_{short} . From this, the bound on the size of our routing scheme will follow.

References

- 1 A. Abboud and G. Bodwin. The $4/3$ additive spanner exponent is tight. In *STOC*, pages 351–361, 2016.
- 2 Ittai Abraham, Shiri Chechik, Cyril Gavoille, and David Peleg. Forbidden-set distance labels for graphs of bounded doubling dimension. *ACM Transactions on Algorithms (TALG)*, 12(2):22, 2016.
- 3 Surender Baswana and Neelesh Khanna. Approximate shortest paths avoiding a failed vertex: Near optimal data structures for undirected unweighted graphs. *Algorithmica*, 66(1):18–50, 2013.
- 4 Michael A. Bender and Martin Farach-Colton. The level ancestor problem simplified. *Theor. Comput. Sci.*, 321(1):5–12, 2004. doi:10.1016/j.tcs.2003.05.002.
- 5 Aaron Bernstein and David R. Karger. A nearly optimal oracle for avoiding failed vertices and edges. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 101–110, 2009.
- 6 Davide Bilò, Fabrizio Grandoni, Luciano Gualà, Stefano Leucci, and Guido Proietti. Improved purely additive fault-tolerant spanners. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 167–178, 2015.
- 7 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Fault-tolerant approximate shortest-path trees. In *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 137–148, 2014.
- 8 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Compact and fast sensitivity oracles for single-source distances. In *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, pages 13:1–13:14, 2016.
- 9 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido 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.
- 10 Greg Bodwin, Fabrizio Grandoni, Merav Parter, and Virginia Vassilevska Williams. Preserving Distances in Very Faulty Graphs. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 73:1–73:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

- 11 Jean Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985.
- 12 Shiri Chechik. Fault-tolerant compact routing schemes for general graphs. *Inf. Comput.*, 222:36–44, 2013.
- 13 Shiri Chechik, Sarel Cohen, Amos Fiat, and Haim Kaplan. $(1 + \epsilon)$ -approximate f -sensitive distance oracles. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1479–1496, 2017.
- 14 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. f -sensitivity distance oracles and routing schemes. *Algorithmica*, 63(4):861–882, 2012.
- 15 Don Coppersmith and Michael Elkin. Sparse source-wise and pair-wise distance preservers. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 660–669. SIAM, 2005. URL: <http://dl.acm.org/citation.cfm?id=1070432.1070524>.
- 16 Bruno Courcelle and Andrew Twigg. Compact forbidden-set routing. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 37–48. Springer, 2007.
- 17 Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, 2008.
- 18 Ran Duan and Seth 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.
- 19 Fabrizio Grandoni and Virginia Vassilevska Williams. Improved distance sensitivity oracles via fast single-source replacement paths. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 748–757, 2012.
- 20 Telikepalli Kavitha. New pairwise spanners. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPICs*, pages 513–526. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPICs.STACS.2015.513.
- 21 Kavindra Malik, A.K. Mittal, and Sumit K. Gupta. The k most vital arcs in the shortest path problem. *Oper. Res. Lett.*, 8:223–227, 1989.
- 22 Jiří Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93(1):333–344, 1996.
- 23 Merav Parter. Vertex fault tolerant additive spanners. In *International Symposium on Distributed Computing*, pages 167–181. Springer, 2014.
- 24 Merav Parter and David 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.
- 25 Merav Parter and David Peleg. Fault tolerant approximate BFS structures. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1073–1092, 2014.
- 26 Mihai Patrascu and Liam Roditty. Distance oracles beyond the thorup-zwick bound. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 815–823. IEEE, 2010.
- 27 Mihai Patrascu, Liam Roditty, and Mikkel Thorup. A new infinity of distance oracles for sparse graphs. In *Foundations of Computer Science (focs), 2012 Ieee 53rd Annual Symposium on*, pages 738–747. IEEE, 2012.

- 28 Christian Sommer, Elad Verbin, and Wei Yu. Distance oracles for sparse graphs. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 703–712. IEEE, 2009.
- 29 Mikkel Thorup and Uri Zwick. Compact routing schemes. In *SPAA*, pages 1–10, 2001.
- 30 Oren Weimann and Raphael Yuster. Replacement paths via fast matrix multiplication. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 655–662, 2010.