Efficient Oracles and Routing Schemes for Replacement Paths

Davide Bilò¹, Keerti Choudhary², Luciano Gualà³, Stefano Leucci⁴, Merav Parter⁵, and Guido Proietti⁶,⁷

¹ Università di Sassari, Italy. davidebilo@uniss.it
² Department of CSE, I.I.T. Kanpur, India. keerti@cse.iitk.ac.in
³ Università di Roma “Tor Vergata”, Italy. guala@mat.uniroma2.it
⁴ ETH Zürich, Switzerland. stefano.leucci@inf.ethz.ch
⁵ CSAIL, MIT. meravparter@gmail.com
⁶ DISIM, Università dell’Aquila, Italy. guido.proietti@univaq.it
⁷ Istituto di Analisi dei Sistemi ed Informatica, CNR, Roma, Italy.

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 a graph G, the replacement path \( \pi_{G-e}(s,t) \) is a shortest s–t path that avoids e. In this paper we focus on the general case in which \( (s,t) \in S \times T \), where \( S, T \subseteq V(G) \), and we present several efficient constructions maintaining the collection of all \( \pi_{G-e}(s,t) \) for every \( e \in E(G) \), 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 \( S, T \subseteq V \), we construct a DSO for maintaining \( S \times T \) distances under single edge (or vertex) faults. This DSO has size \( O(n^{\sqrt{\log n}}) \) and query time of \( O(\sqrt{n^{\log n}}) \). At the expense of having quasi-polynomial query time, the size of the oracle can be improved to \( O(|T| \sqrt{|S|n}) \), which is optimal for \( |T| = \Omega(\sqrt{|S|}) \). When \( |T| = \Omega(n^{\frac{3}{2}}|S|)^{\frac{1}{2}} \), 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. These lower bound results establish a separation between the oracle and the subgraph settings.

2. **FTP:** We show the construction of a path-reporting DSO of size \( \tilde{O}(n^{3/2}|S||T|) \) reporting \( \pi_{G-e}(s,t) \) in \( O(|\pi_{G-e}(s,t)| + |S||T|)^2 \) 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 both in space and query time for the special case in which \( T = V(G) \). Our FPT improves over previous constructions when \( |T| = O(\sqrt{|S|n}) \).

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 provide optimal labeling scheme for the setting.

1998 ACM Subject Classification G.2.2 Graph Algorithms

Keywords and phrases Fault tolerant, Shortest path, Oracle, Routing

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 [21] 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 \( G = (V(G), E(G)) \), [21] 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 a construction of FT-BFS for each \( s \in S \) with \( O(\sqrt{|S|}n^{3/2}) \) edges. This subgraph setting has not yet been studied for the more general \( S \times T \) setting. 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 [10, 19, 1], one can show that the lower bound of \( \Omega(\sqrt{|S|}n^{3/2}) \) edges for FT-BFS structures for \( S \times V \) distances applies against any kind of distance sensitivity oracles, 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 \( \Omega(\sqrt{|S|}n^{3/2}) \) bits.*

Unlike the sourcewise setting 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 \( V \). We fill some of that gap 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

---

\(^1\) We focus on single edge failures and undirected graphs, although most of the results extend to single vertex failures and directed graphs as well.
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 \( \tilde{O}(\sqrt{|S||T|}) \); such an oracle is also able to report a corresponding path in time \( O(n|S||T|^{1/3}) \).
- A polynomial time constructable \( S \times T \) DSO of size \( \tilde{O}(n^{3/2}) \), constant query time, and additive distortion \( O(\sqrt{n}) \). **MP: Does the space depend on** \(|T|\)?

Let \( \epsilon \in (0, 1] \) be any fixed constant; we show that conditioned on the Set-Intersection Conjecture [23], any \( \{s\} \times V \) DSO with constant query time and additive distortion \( d = O(n^{1-\epsilon}) \) must use \( \tilde{\Omega}(n^{2\epsilon}) \) 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 \( O(\sqrt{|S||T|^{3/2}}) \), using the FT-BFS construction of [21] for multiple sources \( S \).

Concerning the lower bound for the single source setting, it compares favorably with the non-conditional lower-bounds given in [22]. 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 \( d = O(n^{3-\epsilon}) \), for any constant \( \epsilon > 0 \). Moreover, it shows that for any \( d = O(1) \), \( \tilde{\Omega}(n^{2}) \) 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 \( O(n^{2}) \) given in [22] 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 \( O(n^{2}) \) 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 is moved to Appendix B.

**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. As a corollary of our single source DSO, we also obtain space-optimal labeling scheme. It computes a label with \( \tilde{O}(\sqrt{n}) \) bits for each node, which allows to compute \( \text{dist}(s, t, G - e) \), namely the distance in \( G - e \) between \( s \) and \( t \), by simply looking at the label of \( t \) and of the endvertices 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 \( P \) can be decomposed into two (fault-free) tree paths connected by
an edge. By combining the routing schemes for trees of Thorup and Zwick [26] 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 data structure 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 [6] and the distance oracle setting [7]. *Multiple faults* have been studied in [8, 22] and structures with additive stretch have been studied in [20, 5].

**Distance sensitivity oracles (for all pairs).** In a seminal work, Demetrescu et al. [15] showed that given a directed weighted graph $G$ of size $n$, it is possible to construct in time $\tilde{O}(mn^2)$ a distance sensitivity oracle 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 [4]. Grandoni and Williams [17] presented the first distance sensitivity oracle for that achieves simultaneously subcubic preprocessing time and sublinear query time for weighted graphs with bounded integer weights. A dual failure fault tolerant distance sensitivity oracle of size $O(n^2 \log^3 n)$ and $O(\log n)$ query time was presented in [16]. The $f$ faults case was studied in [27, 12].

**FT distance labels and compact routing schemes.** Label-based fault-tolerant routing schemes for graphs of bounded clique-width were presented in [14]. 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 [13], for up to two edge failures. Further work considered multiple failures [11] 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 [25, 23, 24]. 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 $G$ ($H$ can be same as $G$). 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 $\mathcal{T}_{w,R}$ 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 \( d_H(u, v) \) its length, i.e., the distance between \( u \) and \( v \) in \( H \). Moreover, whenever the graph is clear from the context, we might also omit the subscripts. 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 \( \pi \), we denote by \(|\pi|\) its size, i.e., the number of its edges. Moreover, if \( \pi \) traverses the vertices \( u \) and \( v \) in this order, we denote by \( \pi(u, v) \) the subpath of \( \pi \) between \( u \) and \( v \) (endpoints included). For any non-negative integer \( i \), we define \( \pi[-i] \) to be path containing the last \( \min\{|\pi|, i\} \) edges of \( \pi \). 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 \( T \) and any two vertices \( a, b \in T \), we use the notation \( T(a, b) \) to denote the path from \( a \) to \( b \) in tree \( T \). 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. \( \Omega(f(x)/\text{polylog} f(x)) \)). Also \( \Theta(f(x)) \) is used to denote a function \( f(x) \) which is both \( O(f(x)) \) and \( \Omega(f(x)) \). Below we state a lemma that will be crucially used in our fault tolerant data structures (for the proof see Appendix A).

\textbf{Lemma 1.} Let \( L \in [5, n/\log n] \) and \( \mathcal{P} = \{\pi_G(u, v) \mid u, v \in V(G), d_G(u, v) \geq L \log n\} \) 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 helps us to uniquely define \(|\pi(x,y)|\). 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 \( w_P \) we have \(|P| = |w_P|\).

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 [15]. 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 \( T^w \) and \( T^w_R \);
   - The pre-order and post-order numbering, and depth of each \( v \in V \) in \( T^w \) and \( T^w_R \);
   - A level ancestor data structure for trees \( T^w \) and \( T^w_R \);
   - 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 index \( i \geq 0 \):
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 \( |\pi(s, u)| = 2^i \) or \( |\pi(v, t)| = 2^i \);

B2(s, t, i) stores the distance \( d_{G-\pi(s, u)}(s, t) \), where \( u, v \) are vertices on \( \pi(s, t) \) and satisfying (i) \( |\pi(u, v)| = 2^i \); and (ii) \( |\pi(s, u)| = 2^i \) or \( |\pi(v, t)| = 2^i \) (or both).

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 \( T_w \) and \( T_w^R \), \( w \in W \)). 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 \( T_w \) and \( T_w^R \). 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 \( d_{G-e}(s, t) = d_{G-e}(s, t') + d_G(t', t) \), and if \( P \) passes through \( s' \), then \( d_{G-e}(s, t) = d_G(s, s') + d_{G-e}(s', t) \). So in this case we can use B1 to report the distance between \( s \) and \( t \) in \( G - e \).

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., \( \pi_{G-e}(s, t) = \pi_{G-G-\pi(u', v')}(s, t) \). 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

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 \( L \) 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 \( \ell \) 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 \( e_1, e_2, \ldots, e_{\min\{\ell, |\pi(s, t)|\}} \) are the edges on \( \pi(s, t)[-\ell] \) listed in reverse order (i.e., from \( t \) towards \( s \)), then store in \( d_{G-e_s}(s, t) \) the distance \( d_{G-e_s}(s, t) \).

**Lemma 3.** Let \( e = (u, v) \) be an edge lying on \( \pi_G(s, t) \) for some vertices \( s, t \in V(G) \). Also let \( x \in V(G) \) be such that \( d_G(x, t) \leq d_G(v, t) \). Then \( e \not\in \pi_G(x, t) \), and so \( d_{G-e}(x, t) = d_G(x, t) \).

![Figure 1 Depiction of vertices \( s', u', v', t' \) when the failing edge \( e = (u, v) \) lies on path \( \pi_G(s, t) \).]
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 $d_{(i)}^{-1} = d_{G-e}(s, t)$
2. else return $\min_{x \in R, d_{G}(x, t) \leq \ell} (d_{G-e}(s, x) + d_{G}(x, t))$

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

$$d_{G}(x, t) = d_{G}(x, u') + 1 + d_{G}(v', t) \geq 1 + d_{G}(v, t) > d_{G}(v, t).$$

However, by our hypothesis $d_{G}(x, t) \leq d_{G}(v, t)$. Hence, we get a contradiction.

\[ \blacklozenge \]

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

Proof. Let $P$ be the shortest path from $s$ to $t$ in $G - e$. Since $|P| \geq d_{G}(s, t) > \ell$, 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 $d_{G}(r, t) \leq |\pi_{G-e}(r, t)| \leq \ell \leq d_{G}(r, t)$, Lemma 3 implies that $d_{G-e}(r, t)$ is equal to $d_{G}(r, t)$. Therefore we have:

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

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

\[ \blacklozenge \]

Query algorithm. Consider a pair $(s, t)$, let $e = (u, v)$ be a failing edge lying on $\pi_G(s, t)$. (As before whether $e$ belongs to $\pi(s, t)$ can be verified in constant time). If $\pi_G(s, t)[-\ell]$ contains $e$, then we can output new distance in $O(1)$ time. If $\pi_G(s, t)[-\ell]$ does not contain $e$, then by Lemma 4, $d_{G-e}(s, t) = \min_{r \in R, d_{G}(r, t) \leq \ell} (d_{G-e}(s, r) + d_{G}(r, t))$. 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 previous subsection. Since the values $d_{G}(r, t)$ 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 at most $O(|S||T|L \log n)$. We hence obtain the following result:

\[ \blacklozenge \]

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 if we choose $L = \Theta(n/\sqrt{|S||T|})$, then this results in an oracle of size $O(n \sqrt{|S||T|} \log n)$ and query time $O(\sqrt{|S||T|})$. Observe that for $|T| = \Theta(n)$, this matches the optimal-size multi-source preserver provided in [21].

### 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 $R_1, R_2, \ldots, R_\alpha$ for an appropriate $\alpha$. 

We now explain our construction. Let $\alpha$ be integer to be fixed later on, and for $i \in [0, \alpha]$, let $L_i$ be $(n/|S|)^{2\alpha + i}$ and $R_i$ be random set of size $O(n/L_i) = O(|S|^{2\alpha + i} n^{-\alpha})$ computed using Lemma 1. For each $i \geq 0$, we will compute an oracle for $S \times R_i$, say $O_{S \times R_i}$. Also we use $O_{S \times T}$ to compute the oracles 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 $O_{S \times R_i}$ uses $O_{S \times R_{i-1}}$, and $O_{S \times T}$ uses $O_{S \times R_0}$.

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

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

So, we have $\text{size}(O_{S \times R_{i+1}}) = \text{size}(O_{S \times R_i}) + (|R_i||R_{i+1}| + |S||R_{i+1}|) |L_i| \log n$.

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 $\pi_G(s, y)[-L_i, \log n]$ contains $e$ we return the stored distance. Otherwise we compute $d_{G-e}(s, y)$ as $\min_{x \in R_i, d_G(x, y) \leq L_i, \log n} (d_{G-e}(s, x) + d_G(x, y))$, where $d_{G-e}(s, x)$ is obtained by querying $O_{S \times R_i}$. Since at most $|R_i|$ queries to $O_{S \times R_i}$ are performed, we have $

\text{time}(O_{S \times R_{i+1}}) = \text{time}(O_{S \times R_i}) \times (|R_i|)$. Summing the first equation from $i = 0$ to $\alpha$, and substituting $\alpha = \log n - 1$, we obtain the size of oracle $O_{S \times T}$:

$$\text{size}(O_{S \times T}) = \text{size}(O_{S \times R_0}) + \sum_{i=0}^{\alpha} (|R_i||R_{i+1}| + |S||R_{i+1}|) |L_i| \log n$$

$$\leq n|S| \log n + \alpha|R_i|^2 + |R_{\alpha+1}| |T| + \sum_{i=0}^{\alpha-1} (|S||R_{i+1}|) |L_i| \log n + |S||T| L_\alpha \log n$$

$$= (\alpha + \log n)n|S| + |T| \sqrt{n|S|} + \sum_{i=0}^{\alpha-1} (n|S|(n/|S|)^{\frac{\alpha}{2}} \log n) + |T| \sqrt{n|S|} \log n$$

$$= O(n|S| \log^2 n + |T| \sqrt{n|S|} \log n).$$

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

The following theorem follows from above discussion.

\textbf{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 subgraph lower bound of $\Omega(n \sqrt{n|S|})$ provided in [21] holds also for the oracles setting (by using standard information theoretic arguments). Moreover, the lower bound graph of [21] can be easily adopted to yield an $\Omega(|T| \sqrt{n|S|})$ lower bound for the $S \times T$ setting. Moreover, we have that 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 $|T| = \Omega(n^{\frac{1}{3}}|S|^{\frac{2}{3}})$, we can obtain an even better space-optimal oracle (up to logarithmic factors) having polynomial query time.

\textbf{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 $|T| = \Omega(n^{\frac{1}{3}}|S|^{\frac{2}{3}})$ can be preprocessed in polynomial time to compute a DSO of size $O(T \sqrt{n|S|} \log n)$ and query time $O(n^{\frac{2}{3}}|S|^{\frac{2}{3}} |T|^{-1}) = O(n^{\frac{2}{3}}|S|^{\frac{2}{3}}).$
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 \( O_{S \times R} \) 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) \( d_G(x,t) \) for \( x \in R, t \in T \), and (ii) the distance \( d_{G-\epsilon}(s,t) \) for \( s \in S, t \in T, e \in \pi(s,t)[-\ell] \) (recall that \( \ell = \lceil \log n \rceil \)). The overall size is \( O(|S||R|L_0 \log n + |R||T| + |S||T|L \log n) \).

To report \( d_{G-\epsilon}(s,t) \) we proceed in a similar way discussed in Algorithm 1. If \( e \in \pi_G(s,t)[-\ell] \) we return the stored distance. Otherwise we compute \( d_{G-\epsilon}(s,t) \) as \( \min_{x \in R} \left( d_G(s,x) + d_G(x,t) \right) \), where \( d_{G-\epsilon}(s,x) \) is obtained by querying \( O_{S \times R} \). At most \( O(|R|) \) queries to \( O_{S \times R} \) are performed, the total query time is \( O(|R|/L_0) \).

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, whose proof is postponed to Appendix A, provides a family of oracles whose additive stretch decreases as soon as the size increases, regardless of the number of destinations.

\( \triangleright \) 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, which is extensively discussed in Appendix B:

\( \triangleright \) 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 \( \Omega \left( n^{2+\epsilon} \right) \) bits of memory.

The above result improves several (unconditional) lower bounds on fault-tolerant additive-distortion single-source structures, as discussed in the appendix. However, the problem of extending it to the case of multiple sources, in order to make a coherent comparison with the construction provided in Theorem 8, is left open.

### 4 Path-Reporting \( S \times T \) Oracle and Preservers

The following lemma (with proof in Appendix A) 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.)

\( \triangleright \) Lemma 10 (also proved in [9]). 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 \( \pi_G(s,y)::(y,z)::\pi_G(z,t) \) is a shortest path in \( G - e \). We will refer to \( (x,y) \) by \( \text{LINK}(s,t,e) \).

Moreover, the following holds (the proof is postponed to Appendix A):

\( \triangleright \) Lemma 11. Let \( G \) be an undirected weighted graph, \( s,t \in V(G) \) and let \( \pi(u,v) \) be a subpath of \( \pi(s,t) \). There exists an edge \( (y,z) \in G - \pi(u,v) \) that satisfies the following property: \( \forall e \in \pi(x,y) \) if \( \pi_{G-\epsilon}(s,t) \) is vertex-disjoint from \( \pi(u,v) \), then \( \pi_G(s,y)::(y,z)::\pi_G(z,t) \) forms a shortest path in \( G - e \). We will refer to the edge \( (y,z) \) by \( \text{LINK}(s,t,x,y) \).
For $W \times W$, here we only state our result, and its proof is presented in Appendix A).

\textbf{Theorem 12.} 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 $e \in \pi(s,t)\lceil -\ell\rceil$, we store the following: (i) edge $(y,z) = \text{Link}(s,t,e)$, (ii) the distance $d_G(z,t)$, and (iii) the suffix $\pi_{G-e}(s,t)\lceil -\ell\rceil$.

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:

\textbf{Path-reporting Query Algorithm}

1. If $e \notin \pi(s,t)$, we return $\pi_G(s,t)$ stored in the shortest path tree $T_s$ (recall $s \in W$).
2. If $e \in \pi(s,t)\lceil -\ell\rceil$ then we retrieve the pre-stored edge $(y,z) = \text{Link}(s,t,e)$, the distance $d_G(z,t)$, and the path $P = \pi_{G-e}(s,t)\lceil -\ell\rceil$.
   - If $d_G(z,t) \leq \ell$, then $z$ must lie on $P$ and $\pi_{G-e}(s,t)$ will be equal to $\pi_G(s,y)\lceil (y,z);P(z,t)\rceil$.
   - If $d_G(z,t) > \ell$, then we compute a vertex $r \in R$ lying on $P = \pi_{G-e}(s,t)\lceil -\ell\rceil$ in $O(|P|) = O(l)$ time. Such an $r$ exists by Lemma 1. We output $\pi_{G-e}(s,t) = \pi_{G-e}(s,r)\lceil \pi_G(r,t)\rceil$. Notice that in this case $\pi_{G-e}(s,t)$ can be outputted in $O(|\pi_{G-e}(s,t)|)$ time.
3. If $e \notin \pi(s,t)\lceil -\ell\rceil$, then it follows from Lemma 4 that $\pi_{G-e}(s,t) = \pi_{G-e}(s,r)\lceil \pi_G(r,t)\rceil$, where $r = \arg\min\{d_{G-e}(s,x) + d_G(x,t) \mid x \in R, d_G(x,t) \leq \ell\}$. 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:

\textbf{Theorem 13.} For any undirected unweighted graph $G$ there exists a polynomial-time constructible DSO for a source set $S$ and destination set $T$ of size $O(n^{1/3}(|S||T|)^{1/3}\log^2 n)$ 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 $O(n^{1/3}(|S|^{1/3}|T|^{1/3})\log^2 n)$ 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 $e \in \pi(s,t)\lceil -\ell\rceil$, 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 $\pi_G(s,t)$ stored in the shortest path tree $T_s$ (recall $s \in W$); if $e$ appears on $\pi(s,t)\lceil -\ell\rceil$, 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 $r = \arg\min\{d_{G-e}(x) + d_G(z,t) \mid x \in R, d_G(x,t) \leq \ell\}$. We know that the concatenation...
\[ \pi_{G-e}(s, r) \] \text{ is a shortest path from } s \text{ to } t \text{ in } G - e. \text{ Notice that using Theorem 12, } \\
\pi_{G-e}(s, r) \text{ can be reported in } O(|\pi_{G-e}(s, r)|) \text{ time, also the path } \pi_{G}(r, t) \text{ is stored in the tree } T_r \text{ (recall } r \in W). \text{ Thus the time for reporting path } \pi_{G-e}(s, t) \text{ is } O(|R| + |\pi_{G-e}(s, t)|). \n\text{ Notice that } |R| = \sqrt{n|S|} \text{ and } |\pi_{G-e}(s, t)| \geq d_G(s, t) \geq \ell = (n/|R|) \log n = \sqrt{n/S} \log n. \text{ 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). \text{ To summarize, we have:} \\

\textbf{Theorem 14.} For any undirected unweighted graph } G \text{ there exists a polynomial-time constructible DSO for a source set } S \text{ of size } O(n\sqrt{n/S} \log n) \text{ that for any } s \in S, t \in V(G), \text{ and any failing edge } e \in E(G), \text{ can report } \pi_{G-e}(s, t) \text{ in } O(|\pi_{G-e}(s, t)|) \text{ time.}

Remarkably, the above multi-source oracle is the natural counterpart of the multi-source fault-tolerant preserver given in [21], and so it is optimal in space (up to poly-logarithmic factors) and in query time.

\section{Distributed Routing Scheme for Single Source Distances}

In this section, we present the main crux of our distributed algorithms for the single source and all destinations case. Other details for the routing and labeling schemes are deferred to Appendix C. We use } s \text{ to denote the designated source vertex. For any two vertices } a, b, \text{ we use the notation } \text{Treepath}(a, b) \text{ to denote the path from } a \text{ to } b \text{ in tree } T_s.

It turns out that the single source labeling scheme is easy, but designing compact routing scheme is not that straightforward. To overcome this we show how to represent } H (\text{FT-BFS subgraph of } O(n^{3/2}) \text{ size}) \text{ as a union of trees and link-edges (see Definition 10) so that each replacement path can be represented as a combination of two tree paths connected by link edge. We can then use routing scheme over trees by Thorup and Zwick [26]. For ensuring small routing table, we will need that each vertex must be present in only at most } O(\sqrt{n}) \text{ number of trees in the family of trees considered by us.}

\subsection{Tree Representations}

For a parameter } L = \sqrt{n} \text{ we take } R \subseteq V(G) \text{ to be a set of vertices as obtained from Lemma 1. Also } \ell \text{ is taken to be } \lfloor L \log n \rfloor. \text{ We define two family of trees } T_{\text{long}} \text{ and } T_{\text{short}} \text{ as follows: (i) The family } T_{\text{long}} \text{ consists of shortest path tree } T_s, \text{ and the shortest path trees } T_r \text{ for each } r \in R; \text{(ii) The family } T_{\text{short}} \text{ consists of all the possible trees } T_e, z \text{ given by Definition 15 below and have depth at most } \ell.

\textbf{Definition 15.} Let } e = (u, v) \in T_s, \text{ and } (y, z) \text{ be an edge that becomes tree edge in } T_{s, G-e}. \text{ Also assume } d_G(u, z) \leq 2\ell. \text{ We define } T_{e, z} \text{ to be a subtree of } T_{s, G-e} \text{ which is (i) rooted at vertex } z, \text{ and (ii) truncated to depth } \ell, \text{ that is, it contains only those vertices whose depth differ from depth of } z \text{ in } T_{s, G-e} \text{ by at most } \ell.

In the following table, we show how the families } T_{\text{long}} \text{ and } T_{\text{short}} \text{ can be directly used to obtain a shortest path from } s \text{ to any arbitrary vertex } t \text{ after an edge failure on } \text{Treepath}(s, t).
### Efficient Oracles and Routing Schemes for Replacement Paths

<table>
<thead>
<tr>
<th>If</th>
<th>Then</th>
<th>New shortest path from $s$ to $t$ in $G - e$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t = r_0 \in R$</td>
<td></td>
<td>TREEPATH($s, y) :: (y, z) :: \mathcal{T}_{r_0}(z, r_0)$ where $(y, z) = \text{LINK}(s, r_0, e)$</td>
</tr>
<tr>
<td>$e \in \text{TreePath}(s, t)[-\ell]$</td>
<td>$t \in \mathcal{T}<em>{e,z} \in \mathcal{T}</em>{\text{short}}$</td>
<td>TREEPATH($s, y) :: (y, z) :: \mathcal{T}_{e,z}(z, t)$</td>
</tr>
<tr>
<td>$(y, z) = \text{LINK}(s, t, e), d_G(z, t) \leq \ell$</td>
<td></td>
<td>TREEPATH($s, y) :: (y, z) :: \mathcal{T}_{r}(z, r), t$</td>
</tr>
<tr>
<td>Remaining cases</td>
<td>$\pi_{G-e}(s, t) [-\ell]$ must contain a vertex from $R$, say $r$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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, $\pi_G(s, y) :: (y, z) :: \pi_G(z, r_0) = \text{TREEPATH}(s, y) :: (y, z) :: \mathcal{T}_{r_0}(z, r_0)$ is a shortest path from $s$ to $r_0$ in $G - e$. Notice that the trees $\mathcal{T}_e$ and $\mathcal{T}_{r_0}$ are present in the family $\mathcal{T}_{\text{long}}$.

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

Since $d_G(t, z) \leq \ell$, we will have $d_G(u, z) \leq 2\ell$. This along with the fact that $(y, z)$ becomes a tree edge in $\mathcal{T}_{u,G-e}$ shows that tree $\mathcal{T}_{e,z}$ is present in the family $\mathcal{T}_{\text{short}}$. Also from Lemma 10 we know that $e$ cannot lie on $\pi_G(z, t)$, so $d_{G-e}(z, t) = d_G(z, t) \leq \ell$. Since tree $\mathcal{T}_{e,z}$ contains shortest paths up to depth $\ell$, vertex $t$ must lie in $\mathcal{T}_{e,z}$. This shows that $\mathcal{T}_{e,z}(z, t) = \pi_{G-e}(z, t) = \pi_G(z, t)$. 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).** $e \in \text{TreePath}(s, t)[-\ell], (y, z) = \text{LINK}(s, t, e), d_G(z, t) > \ell$

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

**Case 3(ii).** $e \notin \text{TreePath}(s, t)[-\ell]$

Let $P = \pi_{G-e}(s, t)$. By Lemma 1, $P[-\ell]$ 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 at most $\tilde{O}(\sqrt{n})$ trees in the families $\mathcal{T}_{\text{short}}$ and $\mathcal{T}_{\text{long}}$. For the family $\mathcal{T}_{\text{long}}$, proof is trivial because $|\mathcal{T}_{\text{long}}| = O(|R|) = O(\sqrt{n})$. In the following lemma, we provide an upper bound on the number of trees in which a vertex can appear in the family $\mathcal{T}_{\text{short}}$.

**Lemma 16.** Each vertex $t \in V(G)$ lies in at most $O(\sqrt{n} \log n)$ trees in the family $\mathcal{T}_{\text{short}}$.

**Proof.** Consider any tree $\mathcal{T}_{e,z} \in \mathcal{T}_{\text{short}}$ that contains vertex $t$. Notice that $e = (u, v)$ must lie on $\mathcal{T}_{e,z}(s, t)$, and there must exist a vertex $y$ for which $(y, z) = \text{LINK}(s, t, e)$. Also $d_G(u, z) \leq 2\ell$ and $d_{G-e}(z, t) \leq \ell$. Since by Lemma 10, $d_G(z, t) = d_{G-e}(z, t)$, we have $d_G(u, t) \leq 3\ell$.

As $t$ belongs to at most one tree $\mathcal{T}_{e,z}$ for each failing edge $e = (u, v)$, and such an edge $e$ must lie on $\text{TREEPATH}(s, t)$ and satisfy the criteria that $d_G(u, t) \leq 3\ell$, we can conclude that $t$ lies in only $O(\ell) = O(\sqrt{n} \log n)$ trees in the family $\mathcal{T}_{\text{short}}$.$\blacksquare$
References

A Missing Proofs

Reminder of Lemma 1 Let $L \in [5, n / \log n]$ and $\mathcal{P} = \{ \pi_G(u, v) \mid u, v \in V(G), d_G(u, v) \geq L \log n \}$ 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 $O(\frac{2}{L} \log n)$ vertices.

Proof. Let $W$ be a uniformly random set in $V(G)$ containing $c = (4n/L)$ vertices. Note that there are a total of $\binom{n}{c}$ possibilities for set $W$. For any path $P \in \mathcal{P}$, the probability that $W \cap P$ is empty is

$$\frac{(n - |P|)}{\binom{n}{c}} \leq \left( \frac{n - |P|}{n} \right)^c \leq \left( 1 - \frac{L \log n}{n} \right)^{4n/L} \leq \frac{1}{n^4}.$$  

(The above equation holds whenever $c = 4n/L < (n - L \log n)$ and $L \log n \leq n$, that is, $5 \leq L \leq n/\log n$).

By union bound we get that the probability set $W$ does not intersect a path in $\mathcal{P}$ is at most $1/n^2$. Now we can repeat the process of randomly choosing set $W$ and verifying if it actually intersects all paths in $\mathcal{P}$, until we are able to find one. This set (which intersects all paths in $\mathcal{P}$) will be the required set $R$. The time for verification is polynomial in $n$, and the expected number of repetitions is less than 2. Hence the expected time for computing set $R$ is also polynomial in $n$.

In order to compute $R$ deterministically, consider an instance of the hitting set problem [18] where the universe is set $V(G)$ and the family of subsets of $V(G)$ is $\mathcal{P}' = \{ V(P) \mid P \in \mathcal{P} \}$. From above discussion it follows that the optimal hitting set contains at most $4n/L$ vertices. To obtain the set $R$ deterministically, it suffices to compute (in polynomial time) an $O(\log n)$-approximation of the hitting-set instance. This gives us a deterministic polynomial time construction for set $R$ with at most $\log n \cdot 4n/L = O(\frac{n}{L} \log n)$ vertices.

Reminder of Theorem 8 Let $L \in [5, n/|S|]$ and $G$ be an undirected graph with $S$ as source set and $T$ as destination set. Then, there exists a polynomial-time constructible approximate DSO of $O(n^2/L) \cdot \log n$ size that for any query $(s, t, e)$ reports in constant time a distance $\hat{d}_{G-e}(s, t)$, satisfying the relation $d_{G-e}(s, t) \leq \hat{d}_{G-e}(s, t) \leq d_{G-e}(s, t) + 2L \log n$.

Proof. We first prove the following useful lemma:

**Lemma 17.** Let $G$ be an undirected graph and $s, t, x \in V(G), e \in E(G)$ be such that $e \in \pi(s, t)$ and $e \notin \pi(x, t)$, then $d_{G-e}(s, t) \leq d_{G-e}(s, x) + d_G(x, t) \leq d_{G-e}(s, t) + 2d_G(x, t)$.

Proof. Since $e \notin \pi_G(x, t)$, the concatenation $\pi_{G-e}(s, t) : \pi_G(x, t)$ is a path from $s$ to $t$ in $G - e$, thus $d_{G-e}(s, t) \leq d_{G-e}(s, x) + d_G(x, t)$. Next, since $G$ is undirected, we have $d_{G-e}(s, x) \leq d_{G-e}(s, t) + d_{G-e}(t, x) = d_{G-e}(s, t) + d_G(x, t)$.

The result follows on adding $d_G(x, t)$ on both sides of above inequality.

Then, we take a random set $R$ of $O(n/L)$ vertices and compute the exact oracle for $W \times W$, where $W = S \cup R$. In addition, for each $t \in T$, we store an arbitrary vertex $r_t \in \{ x \in R \mid d_G(x, t) \leq \ell \} \subseteq R$. Also for each $s \in S$ and each $v \in V(G)$, if $e = (u, v)$ is an edge in $E(G)$ lying on $\pi(s, v)$, then we store the distance $d_{G-e}(s, v)$. Notice that the total space required is still $O(n(|S| + \frac{n}{L}) \log n) = O(\frac{n^2}{L} \log n)$, due to the range of $L$. 


Let $e = (u, v)$ be a failing edge. As before, we can detect in constant time whether or not $e$ belongs to $\pi(s, t)$. If $e \in \pi(s, t)[-\ell]$, we output $d_{G-e}(s, t) = d_{G-e}(s, v) + d_G(v, t)$. Since $\pi(v, t)$ does not contain $e$ and $d(v, t) \leq \ell$, by Lemma 17 the reported distance has a stretch of at most $2\ell$. If $e$ lies on $\pi(s, t)$ but not on $\pi(s, t)[-\ell]$, we output $d_{G-e}(s, t) = d_{G-e}(s, r_i) + d_G(r_i, t)$.

From Lemma 3, $e \notin \pi_G(r_i, t)$, and by definition $d_G(r_i, t) \leq \ell$, thus the additive stretch in this case as well is at most $2\ell$.

**Reminder of Lemma 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 $\pi_G(s, y) : (y, z) : \pi_G(z, t)$ is a shortest path in $G - e$. We will refer to $(x, y)$ by $\text{LINK}(s, t, e)$.

**Proof.** Let $(y, z)$ be the first edge traversed by $\pi = \pi_{G-e}(s, t)$ such that $y$ (resp $z$) is in the same connected component as $s$ (resp. $t$) in $T_s - e$. Clearly, by definition of $\pi$, $\pi_G(s, y) = \pi(s, y)$ can not contain $e$. Consider now $\pi' = \pi(z, t)$ and assume, towards a contradiction, that $\pi_G(z, t)$ traverses $e = (u, v)$. Let $u'$ (resp. $v'$) be the first (resp. last) vertex in $\{u, v\}$ it encounters, and let $w_e$ be the weight $e$. Since $z, t$ lie in $\pi_G(s, t)$, we have

$$d_G(z, t) = d_G(z, u') + w_e + d_G(v', t) \geq d_G(z, v) + w_e + d_G(v, t)$$

However, the concatenation $T_s(z, v) : T_s(v, t)$ is a path from $z$ to $t$ in $G$ whose length is $d_G(z, v) + d_G(v, t)$ which is strictly less than $d_G(z, v) + d_G(v, t) + w_e$. Hence, we get a contradiction.

**Reminder of Lemma 11** Let $G$ be an undirected weighted graph, $s, t \in V(G)$ and let $\pi(u, v)$ be a subpath of $\pi(s, t)$. There exists an edge $(y, z) \in G - \pi(u, v)$ that satisfies the following property: $\forall e \in \pi(x, y)$ if $\pi_{G-e}(s, t)$ is vertex-disjoint from $\pi(u, v)$, then $\pi_G(s, y) : (y, z) : \pi_G(z, t)$ forms a shortest path in $G - e$. We will refer to the edge $(y, z)$ by $\text{LINK}(s, t, x, y)$.

**Proof.** Let $e' \in \pi(x, y)$ be an edge such that $\pi_{G-e'}(s, t)$ is vertex-disjoint from $\pi(x, y)$ (if no such edge exists then the claim is immediately satisfied). We choose $(z, y)$ as $\text{LINK}(s, t, e')$, as discussed in the proof of Lemma 10.

Let $e \in T^R_s$ be a failing edge that is also vertex-disjoint from $\pi(x, y)$. Due to the vertex-disjointness property, both $\pi_{G-e'}(s, t)$ and $\pi_{G-e}(s, t)$ coincide with $\pi_{G-\pi(u, v)}(s, t)$, meaning that $\pi_{G-e}(s, t) = \pi_{G-e'}(s, t)$. Moreover, by our choice of $(u, b)$ we also have:

$$\pi_{G-e}(s, t) = \pi_{G-e'}(s, t) = \pi_G(s, y) : (y, z) : \pi_G(z, t).$$

**Reminder of Theorem 12** 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 $e$.

**Proof.** We compute our path-reporting $W \times W$ oracle by augmenting the $W \times W$ DSO of Section 3.1, as follows:

- In addition to $d_{G-e}(s, t)$, $B1(s, t, i)$ also stores $\text{LINK}(s, t, e)$ (where $e = (u, v)$ is an edge lying on $\pi(s, t)$ and satisfying $|\pi(s, u)| = 2^i$ or $|\pi(v, t)| = 2^i$).
- In addition to $d_{G-e}(u, v)(s, t)$, $B2(s, t, i)$ also stores $\text{LINK}(s, t, u, v)$ (where $u, v$ are vertices on $\pi(s, t)$ and satisfying (i) $|\pi(u, v)| = 2^i$, and (ii) $|\pi(s, u)| = 2^i$ or $|\pi(v, t)| = 2^i$).
To answer a path query for the pair \( s, t \in W \times W \) when edge \( e \in \pi(s, t) \) is failing we proceed as follows. Let \( P \) be an \( s-t \) shortest path in \( G - e \). 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 \( |\pi(v, t')| = 2^{j_0} \). We have the following two cases.

1. \( P \) passes through either \( s' \) or \( t' \):
   - If \( P \) passes through \( t' \), then let \( (y, z) \) be \( \text{LINK}(s, t', e) \) stored in \( B_1(s, t', j_0) \). By our hypothesis together with Lemma 10 we can rewrite \( \pi_{G-e}(s, t) \) as:
     
     \[
     \pi_{G-e}(s, t) = \pi_{G-e}(s, t') \triangleq \pi_G(t', t) = \pi_G(s, y) :: (y, z) :: (z, t') :: \pi_G(t', t) = \pi_G(s, y) :: (y, z) :: \pi_G(z, t).
     \]

     Here, the path \( \pi_G(s, y) \) is stored in \( T_s \), while \( \pi_G(z, t) \) is stored in \( T_t^R \).
   - If \( P \) passes through \( s' \), then let \( (y, z) \) be \( \text{LINK}(t, s', e) \) stored in \( B_1(t, s', i_0) \). We can similarly rewrite \( \pi_{G-e}(s, t) \) as:
     
     \[
     \pi_{G-e}(s, t) = \pi_{G-e}(s, s') \triangleq \pi_{G-e}(s', t) = \pi_{G}(s, y) :: (y, z) :: \pi_{G}(s', z) :: (z, y) :: \pi_{G}(y, t) = \pi_{G}(s, y) :: (y, z) :: \pi_{G}(y, t)
     \]

     Here, \( \pi_{G}(s, z) \) is stored in \( T_s \), while \( \pi_{G}(y, t) \) is stored in \( T_t^R \).

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., \( \pi_{G-e}(s, t) = \pi_{G-e}(u', v')(s, t) \). By Lemma 11 \( \pi_{G-e}(s, t) = \pi_{G}(s, y) :: (y, z) :: \pi_{G}(z, t) \) where \( (y, z) = \text{LINK}(s, t, u', v') \). So in this case we can use \( \text{LINK}(s, t, u', v') \) stored in \( B_2(s, t, i_0) \), together with \( T_s \) and \( T_t^R \), to report \( \pi_{G-e}(s, t) \).

To obtain a a subgraph \( H \) with \( O(nW \log n) \) such that \( d_{G-e}(s, t) = d_H(s, t) \) for \( (s, t) \in W \times W, e \in E(G) \), it suffices to union all the trees \( T_w, T_w^R \) for \( w \in W \), with all the edges \( \text{LINK}(s, t, e) \) and \( \text{LINK}(s, t', u, v) \) stored in \( B_1 \) and \( B_2 \).

**B. A lower bound for the single-source DSO**

Our lower bound relates the problem of building a single-source DSO to the problem of designing an oracle for the following set-intersection problem: let \( U = \{1, 2, \ldots, X\} \) be a universe of \( X \) elements, we are given two collections of subsets of \( U \), namely \( \mathcal{A}, \mathcal{B} \subseteq 2^U \). A set-intersection oracle for \( \langle \mathcal{A}, \mathcal{B} \rangle \) is a data structure that is able to answer queries of the form “Does \( A_i \in \mathcal{A} \) intersect \( B_j \in \mathcal{B} \)?”.

**Conjecture 1 (Similar to Conjecture 3 in [23]).** Let \( |\mathcal{A}| = |\mathcal{B}| = N \). If \( |U| = \Omega(\log N) \) for some constant \( c \), then any set-intersection oracle for \( \langle \mathcal{A}, \mathcal{B} \rangle \) having constant query time must use \( \Omega(N^c) \) bits of memory.

**Lemma 18.** Let \( \epsilon \in (0, 1] \) be any fixed constant, \( |\mathcal{A}| = \sqrt{N} \), \( |\mathcal{B}| = N \) and \( |U| = O(\log^c N) \) for some constant \( c \). If, given any \( n \)-vertex graph, there exists a single-source DSO having additive distortion \( d = O(n^{1-\epsilon}) \), size \( f(n, d) \), and constant query time, then there exists a set-intersection oracle for \( \langle \mathcal{A}, \mathcal{B} \rangle \) having size \( f(O(N^{1/2}), d) \) and constant query time.

**Proof.** Let \( \mathcal{A} = \{A_1, \ldots, A_\sqrt{N}\} \) and \( \mathcal{B} = \{B_1, \ldots, B_N\} \) be \( U = \{u_1, \ldots, u_k\} \) be the input for the Set-Intersection problem. We set \( d' = \left\lceil N^{1/2-1/\log N} \right\rceil \) for a large enough constant.
α and we construct a graph G containing: (i) A path of length \( \sqrt{N} + 1 \) whose vertices are \( s, u_1, u_2, \ldots, u_{\sqrt{N} + 1} \) (we think of vertex \( u_i \) as being associated to the set \( A_i \)); (ii) A vertex \( x_k \) for each \( k \in U \); (iii) A vertex \( v_j \) for each set \( B_j \in \mathcal{B} \); (iv) For every \( B_j \in \mathcal{B} \) and every \( k \in B_j \), a path of length \( k + 1 \) between \( x_k \) and \( v_j \); (v) For every \( A_i \in \mathcal{A} \) and every \( k \in A_i \), a path of length \((\sqrt{N} - i + 1)(2 + d')\) between \( u_i \) and \( x_k \).

Notice that the number \( n \) of vertices of \( G \) can be upper bounded by:

\[
n = O(d'N|U|) + |U| \sum_{i=1}^{\sqrt{N}} (\sqrt{N} - i + 1)(2 + d') \leq O(d'N|U|) + (2 + d')|U| \sum_{i=1}^{\sqrt{N}} i
\]

and since \( n = O(N^{\frac{1}{2}} \ polylog N) = O(N^{\frac{1}{2}} \ polylog n) \), we have \( N = \Omega\left(\frac{n^\epsilon}{polylog n}\right) = \tilde{\Omega}(n^\epsilon)\).

Let \( e_i = (u_i, u_{i+1}) \) and let \( L = 3 + 2\sqrt{N} + d' (\sqrt{N} - i + 2) - i \). We will now show that if \( A_i \cap B_j \neq \emptyset \) then \( d_{G-e}(s, v_j) \leq L \), while \( d_{G-e}(s, v_j) \geq L + d' + 1 \) otherwise.

If \( A_i \cap B_j \neq \emptyset \) let \( k \in A_i \cap B_j \). We have:

\[
d_{G-e}(s, v_j) \leq d_{G-e}(s, u_i) + d_{G-e}(u_i, x_k) + d_{G-e}(x_k, v_j)
\]

\[
= i + (\sqrt{N} - i + 1)(2 + d') + 1 + d' = 3 + 2\sqrt{N} + d' (\sqrt{N} - i + 2) - i + 2 + 2d' = L + 2d' + 2
\]

If \( z < i \) (notice that the case \( z > i \) is impossible), then:

\[
|\pi| \geq d_{G-e}(s, u_z) + d_{G-e}(u_z, x_k) + d_{G-e}(x_k, v_j)
\]

\[
\geq z + (\sqrt{N} - z + 1)(2 + d') + 1 + d' \geq i - 1 + (\sqrt{N} - i + 2)(2 + d') + 1 + d' = 3 + 2\sqrt{N} + d' (\sqrt{N} - i + 2) - i + 1 + d' = L + d' + 1.
\]

Remember that \( N \geq n^\epsilon / \log^\beta n \) for some constant \( \beta \), therefore (for sufficiently large values of \( N \) and \( \alpha \)):

\[
d' \geq N^{\frac{1}{\beta} - 1} \log^\alpha N \geq \frac{n^{1-\frac{1}{\alpha}}}{\log^{\beta(1-\frac{1}{\alpha})} n} \log^\alpha n \geq \frac{n^{1-\frac{1}{\alpha}}}{\log^\beta n} (\epsilon \log n - \beta \log \log n)^\alpha
\]

\[
\geq \frac{n^{1-\frac{1}{\alpha}}}{\log^\beta n} \left( \frac{\epsilon}{2} \right)^\alpha \log^\alpha n = \left( \frac{\epsilon}{2} \right)^\alpha \cdot n^{1-\frac{1}{\alpha}} \log^{\frac{\alpha-1}{\beta}} n \geq n^{1-\frac{1}{\alpha} - \frac{1}{\alpha} \beta} = d.
\]

We now construct a single-source DSO \( O \) with additive distortion \( d \) of \( G \) using \( s \) as the source vertex. The resulting oracle has size \( f(n, d) = f(O(N^{\frac{1}{2}}), d) \) and allows us to answer set-intersection queries as follows: to check whether \( A_i \) intersects \( B_j \) we query \( O \) for the distance between \( s \) and \( v_j \) in \( G - e_i \), where \( e_i = (u_i, u_{i+1}) \). If the returned distance is at most \( L + d \) the sets intersect, otherwise they do not.
Then, Theorem 9 provided in the main body of the paper can now be rephrased as follows:

\[ \textbf{Theorem 19.} \] Let \( \epsilon \in (0, 1] \) be any fixed constant. If Conjecture 1 holds, then any single-source DSO with constant query time and additive distortion \( d = O(n^{1-\epsilon}) \) must use \( \tilde{\Omega}(n^{2\epsilon}) \) bits of memory.

\[ \textbf{Proof.} \] Let \( f(n,d) \) denote the size of a single-source DSO \( O \) with additive distortion \( d \) and constant query time. Consider a set \( U \) and let \( A = \{A_1, \ldots, A_N\} \), \( B = \{B_1, \ldots, B_N\} \) be two collection of \( N \) subsets of \( U \), with \( |U| = O(\log^c n) \) for some constant \( c \). For the sake of simplicity we suppose that \( N \) is a perfect square.

We partition the elements of \( A \) into \( \sqrt{N} \) groups \( S_1, \ldots, S_{\sqrt{N}} \) of \( \sqrt{N} \) elements each (i.e., \( A_i \in S_k \) where \( (k-1)\sqrt{N} < i \leq k\sqrt{N} \)). Then, for each group \( S_k \), we construct an oracle \( O_k \) of size \( |O_k| = f(\tilde{O}(N^{\frac{1}{2}}),d) \) for \( (U, S_k, B) \), as shown by Lemma 18. The resulting collection of oracles allows us to answer in constant time to queries for the intersection of any two sets \( A_i \) and \( B_j \): it suffices to query \( O_k \) where \( k \) is such that \( A_i \in S_k \). As a consequence, if Conjecture 1 holds, we must have \( \sum_{j=1}^{\sqrt{N}} |O_j| \geq \tilde{\Omega}(N^2) \), and hence, for at least one \( j \), we have \( f(\tilde{O}(N^{\frac{1}{2}}),d) = |O_j| = \tilde{\Omega}(N^{\frac{3}{2}}) \). By setting \( n = N^{\frac{4}{3}} \text{polylog}(N) \) it follows that: \( f(n,d) = \tilde{\Omega}(n^{\frac{2}{3}}) \), as claimed.

Notice that if we consider constant additive distortions \( d \), then we strengthen the (unconditional) lower bounds on \( d \)-additive FT-BFS of [22], ranging from \( \Omega(n^{\frac{2}{3}}) \) (for \( d = 2 \)) to \( \Omega(n^{1+\frac{1}{2d-2}}) \) (for \( d > 9 \)) to \( \tilde{\Omega}(n^{\frac{2}{3}}) \). This is in sharp contrast with the existence of a corresponding 4-additive FT-BFS structure of size \( O(n^{\frac{2}{3}}) \) [22], hence the problem of designing a single-source DSO having constant query time is harder than the corresponding single-source spanner problem. Finally, we also extends the range for which superlinear lower bounds
on the size of any $d$-additive FT-BFS are known from $d = O(\log n)$, as shown in [22], to $d = O(n^{\frac{3}{2}})$ and $\Omega(n^{1+\frac{3}{2}})$ bits are still needed.

C A detailed description of the labeling and the routing scheme

In this part of the appendix, we exploit the results provided in Section 5.

C.1 Labeling Scheme

Our labeling scheme $\mathcal{L}$ consists of a label $\lambda(v)$ for each vertex $v$ in $G$, and a virtual edge label $\lambda(e)$ for each tree-edge $e$ in $T_v$. In fact, each of these $n - 1$ virtual labels can be stored in the end vertex farthest from $s$ of the corresponding tree edge. We also assign an empty label $\varnothing$ to each non-tree edge of $G$. Formally, edge and vertex labels are bit strings chosen so that given only the labels of a destination vertex $t$ and of a failing edge $e$, we can compute a query $Q(\lambda(e), \lambda(t)) = d_{G-e}(s, t)$.

We chose an arbitrary DFS traversal of $T_v$ from $s$, and denote $\sigma_u$ (resp. $\tau_u$) to be the time at which the vertex $u \in V(G)$ was encountered for the first (resp. last) time in this traversal. Below we explain the construction of our vertex and edge labels.

= For each vertex $t \in V(G)$, $\lambda(t)$ stores:
  (L1) Value $\sigma_t$, and the distance $d_G(s, t)$.
  (L2) Set $R$, and all the distances $d_G(r, t)$, for each $r \in R$.
  (L3) If $e_i$ is the $i$th edge on the unique path between $t$ and $s$ in $T_v$, then the label $\lambda(t)$ stores the all distances $d_{G-e}(s, t) = \{d_{G-e}(s, t)\}$ for $i = 1, \ldots, \min(\ell, d_G(s, t))$.

= For each edge $e = (u, v) \in E(T_v)$, $\lambda(e)$ stores:
  (L4) Values $\sigma_u$ and $\tau_u$ of vertex $v$, and the distance $d_G(s, u)$.
  (L5) The distances $d_{G-e}(s, r)$, for each $r \in R$.

Lemma 20. The maximum label size is $O(L \log^2 n + |R| \log n)$-bits.

Proof. Let us first consider vertex labels: each label requires $O(\log n)$ bits for (L1), $O(|R| \log n)$ bits to store to all the distances of (L2), and $O(\ell \log n)$ bits for (L3). As for edge labels, each label requires $O(\log n)$ bits for (L4) and $O(|R| \log n)$ bits for (L5). Thus, the maximum label size is $O(L \log^2 n + |R| \log n)$ bits which is $O(L \log^2 n + |R| \log n)$ bits.

C.1.1 Answering a query using the labels $\lambda$

Our query process remains similar to that of Algorithm 1. Following steps in the given order can be used to compute query $Q(\lambda(e), \lambda(t)) = d_{G-e}(s, t)$.

1. If $e \notin \text{TREEPATH}(s, t)$, then we return $d_{G-e}(s, t) = d_G(s, t)$. Notice that $e \notin \text{TREEPATH}(s, t)$ if and only if either $\lambda(e) = \varnothing$, or $\sigma_t < \sigma_v$, or $\sigma_t > \tau_v$.
2. If $e \in \text{TREEPATH}(s, t)[-\ell]$, then $d_{G-e}(s, t) = d_{G-e}^{i}(s, t)$, where $i = d_G(s, t) - d_G(s, u)$. This value was explicitly stored in $\lambda(t)$ (see (L3)). Also the value $i$ can be easily calculated since $d_G(s, t)$ and $d_G(s, u)$ are resp. stored in $\lambda(t)$ and $\lambda(e)$.
3. The only remaining case is $e \in \text{TREEPATH}(s, t)$ and $e \notin \text{TREEPATH}(s, t)[-\ell]$. By Lemma 4, in this case $d_{G-e}(s, t) = \min_{x \in R, \ d_G(x, t) < \ell} (d_{G-e}(x) + d_G(x, t))$. Here the distances $d_{G-e}(s, x), \forall x \in R$ are stored in $\lambda(e)$ while all the distances $d_G(x, t), \forall x \in R$ are stored in $\lambda(t)$.
Thus we are able to show construction of a 1-EFT labeling scheme with labels of size $O(L \log^2 n + |R| \log n)$ bits. Since $L$ is chosen to be $\sqrt{n}$ we get (in the deterministic case) $|R| = O(\sqrt{n} \log n)$, and so we have the following:

\textbf{Theorem 21.} An $n$-vertex unweighted graph $G$ can be preprocessed in polynomial time to compute a 1-EFT SSLLabels $L$ with maximum label size $O(\sqrt{n} \log n)$ bits.

### C.2 Routing Scheme

In order to have efficient routing from $s$ under one edge failure, it suffices to have a routing scheme that can efficiently route packets over any tree in the families $T_{\text{long}}$ and $T_{\text{short}}$. In [26], Thorup and Zwick showed that it is indeed possible to have efficient routing scheme over trees.

\textbf{Lemma 22 ([26]).} For any tree $T_0$ with at most $n$ vertices, there is a labeling scheme that assigns each $v \in T_0$, a label $\sigma(v, T_0)$ of $O(\log^2 n)$-bits such that given the labels of any two nodes $w$ and $y$, one can compute the port number of the neighbour of $w$ lying on $T_0(w, y)$.

![Figure 3](image.png)

- Figure 3: The bold edges belong to tree $T_{G-e}$. Edge $(y, z)$ is the first edge in $\pi_{G-e}(s, t)$ that was not present in $T$. Alternatively, $(y, z) = \text{LINK}(s, t, e)$.

We now explain construction of vertex/edge labels of 1-EFT routing scheme. The routing scheme consists of all the labels of the labeling scheme (L1)–(L5), and in addition the following labels:

- \textbf{(L6)} For each edge $e \in E(T_s)$ and each $r \in R$, $\lambda(e)$ stores the edge $(y, z) = \text{LINK}(s, r, e)$, $O(\log^2 n)$-bit information $\sigma(y, T_s)$ and $\sigma(r, T_e)$, and also the port number of the edge $(y, z)$ going to $z$ from vertex $y$.

- \textbf{(L7)} For each vertex $t$ and each $e \in \text{Treepath}(s, t)[-\ell]$, the label $\lambda(t)$ stores the edge $(y, z) = \text{LINK}(s, t, e)$, the information $\sigma(y, T_s)$ and $\sigma(r, T_e)$, and the port number of edge $(y, z)$ going to $z$ from vertex $y$. It also stores the value $d_G(z, t)$. If $d_G(z, t) > \ell$, then it stores in $\lambda(t)$ an arbitrary vertex of $R$, say $r$, lying on $\pi_G(z, t)$. (Such an $r$ must exist from Lemma 1).

- \textbf{(L8)} For each tree $T_0 \in T_{\text{short}} \cup T_{\text{long}}$, each vertex $v$ of $T_0$ stores in $\lambda(v)$ the $O(\log^2 n)$-bits information $\sigma(v, T_0)$ for efficient routing over $T_0$. (See Lemma 22).
C.2.1 Routing Algorithm

We now explain the steps to route a packet from $s$ to $t$ after failure of an edge $e = (u, v)$.
(Refer to Algorithm 2 for the pseudocode).

1. Check if $e$ lies on $\text{Treepath}(s, t)$, if it doesn’t then we simply send the packet along $s$ to $t$ path in $T_s$. The header of packet stores the information $\sigma(t, T_s)$.
2. Next we check if $t = r \in R$, if it is so, then we extract out the information of the edge $(y, z) = \text{Link}(s, r, e)$ from the label $\lambda(e)$. We route packets along $\text{Treepath}(s, y)$, then the edge $(y, z)$, and finally the tree-path $T_r(z, r)$. The header of the packet stores the information $\sigma(y, T)$, the port number of edge $(y, z)$ at vertex $y$, and the information $\sigma(r, T_r)$ (these can be retrieved from $\lambda(e)$).
3. If $t \notin R$, then we first compute $i = d_G(s, t) - d_G(s, u)$. If $i \leq \ell$ (i.e. $e \in \text{Treepath}(s, t) [-\ell]$), then we extract out $(y, z) = \text{Link}(s, t, e)$ and the value $d_G(z, t)$ from the label $\lambda(t)$.
   - If $d_G(z, t) \leq \ell$, then the route takes the path $\text{Treepath}(s, y); (y, z):\text{Treepath}(z, r)$. In this case the header of the packets stores the information $\sigma(y, T)$, $\sigma(t, T_r)$ and the port number of edge $(y, z)$ at vertex $y$.
   - If $d_G(z, t) > \ell$, then it sets $r$ to be the vertex of $\pi_G(z, t)$ which was pre-stored in $\lambda(t)$. If $i > \ell$, then we compute $r$ as per equation in line 12.
4. Finally, we first route packet to node $r$ (by using the steps 3 to 5) and then along the path $T_r(r, t)$. The header stores the information for routing packet to node $r$ (which can be retrieved from $\lambda(e)$) and the value $\sigma(t, T_r)$. The correctness of this step can be easily verified from Lemma 4.

<table>
<thead>
<tr>
<th>Algorithm 2: Route($t, e$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>if $(\lambda(e) = \emptyset$, or $\sigma_t &lt; \sigma_v$, or $\sigma_t &gt; \tau_v)$ then /* $e \notin \text{Treepath}(s, t)$ */</td>
</tr>
<tr>
<td>Traverse path $\text{Treepath}(s, t)$;</td>
</tr>
<tr>
<td>if $(t = r \in R)$ then</td>
</tr>
<tr>
<td>$(y, z) \leftarrow \text{Link}(s, r, e)$ pre-stored in $\lambda(e)$;</td>
</tr>
<tr>
<td>Traverse path $\text{Treepath}(s, y); (y, z):\text{Treepath}(z, r)$;</td>
</tr>
<tr>
<td>$i \leftarrow (d_G(s, t) - d_G(s, u))$;</td>
</tr>
<tr>
<td>if $(i \leq \ell)$ then</td>
</tr>
<tr>
<td>$(y, z) \leftarrow \text{Link}(s, t, e)$;</td>
</tr>
<tr>
<td>if $d_G(z, t) \leq \ell$ then Traverse path $\text{Treepath}(s, y); (y, z):\text{Treepath}(z, r)$;</td>
</tr>
<tr>
<td>else $r \leftarrow$ an arbitrary vertex of $\pi_G(z, t)$ lying in $R$, pre-stored in $\lambda(t)$;</td>
</tr>
<tr>
<td>$r \leftarrow \arg\min{d_G(x) + d_G(x, t) \mid x \in R, d_G(x, t) \leq \ell}$;</td>
</tr>
<tr>
<td>Route($r, e$) and then Traverse path $T_r(r, t)$;</td>
</tr>
</tbody>
</table>

C.2.2 Analysis of the Size of the Routing tables

Lemma 23. The maximum label size in the 1–EFT routing is $O(L \log^3 n + |R| \log^2 n)$-bits.

Proof. Recall that labels (L1)–(L5) in total required $O(L \log^2 n + |R| \log n)$-bits. In (L6) each edge label requires $O(|R| \log^2 n)$ bits. In (L7) each vertex requires $O(\ell \log^2 n)$ bits. Finally in (L8), each vertex requires $O(\ell \log^2 n + |R| \log^2 n)$ bits as a vertex appears in at most $O(\ell)$ trees in $T_{short}$ and in $O(|R|)$ trees in $T_{long}$. □
Recall that in our routing algorithm, the size of header is at always at most $O(\log^2 n)$ bits. Recall that $L$ was chosen as $\sqrt{n}$, thus we get the following:

\textbf{Theorem 24.} For an undirected graph $G$, there exists a 1-EFT SSRouting $R$ that can route packets from a fixed source vertex $s$ along shortest paths with the following properties.

1. The label of each tree edge and vertex comprise of $O(\sqrt{n} \log^3 n)$ bits.
2. While routing, each packet has to have extra $O(\log^2 n)$ bits as a header.
3. To route packets from $s$ to a destination $t$ under failure of an edge $e$, $s$ should know labels (identity) of both $t$ and $e$. 