Algorithmica (2026) 88:56
https://doi.org/10.1007/s00453-026-01399-z

Fault-Tolerant ST-Diameter Oracles

Davide Bilò[1] · Keerti Choudhary[2] · Sarel Cohen[3] · Tobias Friedrich[4] · Simon Krogmann[4] · Martin Schirneck[5]

Received: 13 April 2026 / Accepted: 25 May 2026

Abstract
Given two vertex sets S and T in a graph, the ST-diameter is the maximum s-t-distance between vertices $s \in S$ and $t \in T$. We study the problem of estimating the ST-diameter of graphs that are subject to a small number of transient edge failures. An *f-edge fault-tolerant ST-diameter oracle* (f-FDO-ST) is a data structure that preprocesses a graph G, sets S, T, and a positive integer f. When queried with a set F of at most f failing edges, the oracle returns an estimate $\widehat{D}$ of the ST-diameter in $G - F$. The oracle is said to have stretch $\sigma \geqslant 1$ if $\operatorname{diam}(G-F, S, T) \leqslant \widehat{D} \leqslant \sigma \cdot \operatorname{diam}(G-F, S, T)$. We design new f-FDO-STs by reducing their construction to that of all-pairs and single-source *distance sensitivity oracles* (f-DSOs). These are data structures that estimate the pairwise graph distances, or respectively the distances from a distinguished source, under up to f failures. We obtain several new trade-offs between the size of the ST-diameter oracles, their stretch guarantees, query and preprocessing times by combining our black-box reductions with f-DSO results from the literature. We further provide a lower bound on the space requirement of approximate ST-diameter oracles. We prove that there exists a family of graphs for which any f-FDO-ST with sensitivity $f \geqslant 2$ and stretch better than $5/3$ requires $\Omega(n^{3/2})$ bits of space, regardless of the query time.

Keywords Diameter oracles · Distance sensitivity oracles · Space lower bounds · Fault-tolerant data structures

1 Introduction

The diameter of a graph is the largest distance between any two of its vertices. It is one of the most fundamental graph parameters as it measures how fast information can spread in a network or how quickly any other node can be visited. The problem of computing, or at least approximating, the diameter in a time-efficient manner has been extensively studied, see e.g. [2–10]. We continue this investigation through the lens of fault tolerance, assuming that the graph undergoes a small number of transient edge failures. The interest in this setting stems from the fact that most real-world

An extended abstract of this work appeared at ICALP 2023 [1].

Extended author information available on the last page of the article

networks are prone to errors. These failures, though unpredictable, are temporary due to some simultaneous repair process that is undertaken in these applications. This has motivated research on the design of *f-edge fault-tolerant oracles*, which are a compact data structures that can quickly report the desired solution or graph property after up to f links failed in the network. The *sensitivity* parameter f describes the oracle's robustness against failures. In the last two decades, a lot of work went into constructing fault-tolerant oracles for graph problems with a special focus on connectivity [11–13] and finding shortest paths [14–24].

The landscape for extremal distances like the diameter is far less explored. We are only aware of three articles in that direction. The problem of designing fault-tolerant diameter oracles was originally raised by Henzinger, Lincoln, Neumann, and Vassilevska Williams [25]. It has recently been studied by Bilò, Cohen, Friedrich, and Schirneck [26] and the same authors together with Choudhary [27]. For a more detailed discussion, see Section 1.1.

We broaden the scope to the notion of ST-diameter that was introduced and studied in recent years. For a given graph $G = (V, E)$ and two sets $S, T \subseteq V$ of vertices, the *ST-diameter* $\operatorname{diam}(G, S, T) = \max_{s\in S, t\in T} d_G(s, t)$ is the maximum distance between vertices in S and T. Clearly, when choosing $S = T = V$, we recover the (ordinary) graph diameter. All previous works on the ST-diameter focus on static graphs. For example, Backurs, Roditty, Segal, Vassilevska Williams, and Wein [4] proved that for any undirected graph one can deterministically compute a 3-approximation of the ST-diameter in time $O(m)$. They also provided a randomized 2-approximate algorithm in time $\widetilde{O}(m\sqrt{n})$ time.[1] Dalirrooyfard, Vassilevska Williams, Vyas, and Wein [28] investigated the problem of computing the bi-chromatic ST-diameter, the special case where the sets S and T partition V.

When applying the fault-tolerant model to the ST-diameter, we get the following notions. An *f-edge fault-tolerant ST-diameter oracle* (f-FDO-ST) is a data structure that stores information about the input graph G in a preprocessing step. When queried with a set $F \subseteq E$ of at most f edges, the oracle returns an estimate $\widehat{D}$ of the ST-diameter of the graph $G - F$. This is the maximum s-t-distances, for $s \in S$ and $t \in T$, under the condition that a shortest path realizing this distance cannot use a failing edge from F. For the (ordinary) diameter, that is, when $S = T = V$, the data structure is called an *f-edge fault-tolerant diameter oracle* (f-FDO). Figure 1 illustrates this problem, we use this graph as a running example. We say an oracle has *stretch* $\sigma \geqslant 1$ if the value $\widehat{D}$ returned upon query F satisfies $\operatorname{diam}(G-F, S, T) \leqslant \widehat{D} \leqslant \sigma \cdot \operatorname{diam}(G-F, S, T)$. Other important performance parameters of such a data structure are its *query time* as well as its *space* requirement.[2] We also consider the time needed to *preprocess* the oracle as a secondary measure.

Since there is a plethora of work on fault-tolerant oracles for shortest paths and the ST-diameter is somewhat related to that, it is a natural question whether we can convert the results on distance computation under failures into oracles for the ST-diameter without sacrificing too much performance.

[1] For a non-negative function $g(n, m, f)$, we write $\widetilde{O}(g)$ for $O(g \cdot \mathsf{polylog}(n))$.

[2] Unless otherwise stated, the space of a data structure is given in the $O(\log n)$-bit word RAM model by the number of stored machine words.

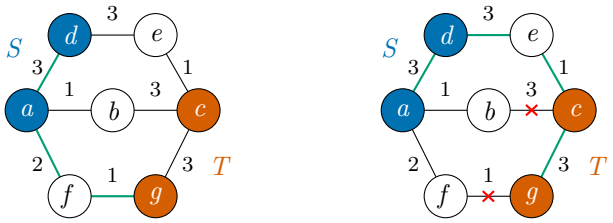

Fig. 1 The edge-weighted graph we use as a running example. It has unique shortest paths and distinguished vertex sets $S = \{a, d\}$ and $T = \{c, g\}$. The ST-diameter is 6, realized by the shortest path from vertex d to g (in green). If the edges $\{b, c\}$ and $\{f, g\}$ fail (red crosses), the ST-diameter lengthens to 10. When queried with the two edges, an f-edge fault-tolerant ST-diameter oracle with stretch $\sigma = 2$ returns a value between 10 and 20

Question *Are there black-box reductions from fault-tolerant ST-diameter oracles to distance oracles without considerable overhead in stretch, query time, and space?*

The epitome of fault-tolerant data structures for shortest-path distances are *f-edge fault-tolerant distance sensitivity oracles* (f-DSOs). The more common *all-pairs* variety can be queried with a triplet (s, t, F) where s, t are any two vertices and F is a set of at most f edges. The oracle then returns an estimate of the *replacement distance* $d_{G-F}(s, t)$ between s and t. A *single-source* f-DSO has a designated source vertex s and the queries only specify the target t and failures F. If the variant is not explicitly stated, we mean all pairs.

When using distance sensitivity oracles to build data structures for the ST-diameter it is important to insist on efficiency. Consider access to some f-DSO $\mathcal{D}$. Naively, whenever the ST-diameter oracle receives a query set F, it could just invoke $\mathcal{D}$ for all pairs of vertices from $S \times T$ and report their maximum replacement distance. The resulting $|S||T|\mathtt{Q}$ query time, where $\mathtt{Q}$ denotes the query time of $\mathcal{D}$, is prohibitive if S and T are large. Instead, all our query times are bounded by a function of the sensitivity f and query time $\mathtt{Q}$ only.

We show how to employ both kinds of DSOs in the construction of fault-tolerant ST-diameter oracles. All our reductions are oblivious to (positive) edge weights, but most results assume the input graph G to be undirected.

Theorem 1 *Let $G = (V, E)$ be an undirected positively edge-weighted graph with n vertices and m edges. Let $S, T \subseteq V$ be two non-empty sets. Assume access to an f-DSO for G with stretch σ, space $\mathtt{S}$, query time $\mathtt{Q}$, and preprocessing time $\mathtt{P}$.*

(i) There exists an f-FDO-ST for G with stretch $1 + 3\sigma$, space $\mathtt{S} + O(n^2)$, query time $O(f^2(\log(f) + \mathtt{Q}))$, and preprocessing time $\mathtt{P} + \widetilde{O}(mn + n|S||T|)$.

(ii) If $f \leqslant \log_2(n)$, the space can be improved to $\mathtt{S} + O(2^{f/2} n^{3/2} \sqrt{\log n})$ at the expense of an $O(f^2(2^f + \mathtt{Q}))$ query time. The stretch and preprocessing time remain the same.

If additionally $M = \max(|S|, |T|) \leqslant 2^{f-1}$ holds, the space of the second variant further decreases to $\mathtt{S} + O(M^{1/2} n^{3/2} \sqrt{\log n})$, with a query time of $O(f^2(M + \mathtt{Q}))$.

The result offers two constructions that mainly differ in the space overhead (over the size $\mathtt{S}$ of the underlying f-DSO) and the query time. Some remark on the preprocessing

Table 1 Properties of the fault-tolerant ST-diameter oracles for undirected graphs obtained via Theorem 1 using all-pairs distance sensitivity oracles from the literature

Sensitivity	Stretch	Space	Query time	Ref.
1	4	$\widetilde{O}(n^2)$	$O(1)$	[29, 30]
1	4	$\widetilde{O}(n^2)$	$O(1)$	[31]
1	$(6k-3)(1+\varepsilon)+1$	$\widetilde{O}(n^{3/2} + k^5 n^{1+1/k}/\varepsilon^4)$	$O(1)$	[32]
2	4	$\widetilde{O}(n^2)$	$\widetilde{O}(1)$	[12]
$f = O(1)$	$10+\varepsilon$	$\widetilde{O}(n^{2-\frac{\alpha}{2(f+1)}}(\log(n)/\varepsilon)^{O(f)})$	$O(n^{\frac{\alpha}{2}}/\varepsilon^2)$	[33]
$f = o(\frac{\log n}{\log\log n})$	4	$\widetilde{O}(n^{3-\alpha})$	$\widetilde{O}(f^2 n^{2-(1-\alpha)/f})$	[24]
$f = o(\frac{\log n}{\log\log n})$	$4+\varepsilon$	$O(fn^{2+o(1)}\log(W)/\varepsilon^f)$	$\widetilde{O}(f^7 \log\log W)$	[34]
$f \geqslant 1$	4	$O(fn^4)$	$f^{O(f)}$	[23]
$f \geqslant 1$	4	$O(f^4 n^2 \log(Mn))$	$O((cf\log(Mn))^{O(f^2)})$	[35]
$f \geqslant 1$	4	$O(Mn^{2+\alpha})$	$\widetilde{O}(f^4 Mn^{2-\alpha} + f^{\omega+2}Mn)$	[36]
$f \geqslant 1$	$(24k-6)(f+1)+1$	$O(n^{3/2+o(1)} + fkn^{1+1/k}\log(Wn))$	$\widetilde{O}(2^f f^2)$	[20]
$f \geqslant 1$	$(24k-6)(f+1)+1$	$O(n^2 + fkn^{1+1/k}\log(Wn))$	$\widetilde{O}(f^3)$	[20]

The maximum edge weight for graphs with arbitrary positive weights is denoted W, for integer-weighted graphs it is M. The parameter $k \geqslant 1$ is a positive integer, $\varepsilon > 0$ a positive real, and $\alpha \in [0, 1]$ is a real number in the unit interval. We use $c > 1$ for a constant, and $\omega < 2.371339$ for the matrix multiplication exponent

time may be in order. The reduction itself takes time $\mathtt{P} + O(mn + n^2 \log n + n|S||T|)$ to compute, but for technical reasons requires that the shortest paths in G are unique.[3] It is always possible to ensure this by either randomly perturbing the edge weights with sufficiently small values, see [37], or by using a more complex deterministic method, known as *lexicographic perturbation* [38–40]. The first method increases the preprocessing only by an additive $O(m)$ term, but makes the preprocessing algorithm randomized. Lexicographic perturbation, in turn, increases the time by an additive $\widetilde{O}(mn)$ [38]. In both cases, this conditioning step increases the preprocessing time in Theorem 1 by at most a logarithmic factor. (A more detailed description can be found at the beginning of Section 3.)

We obtain several new ST-diameter oracles at the same time by applying the reduction from Theorem 1 to existing all-pairs f-DSOs. We give an overview in Table 1. The preprocessing times are omitted for brevity. (The times for the underlying f-DSOs are reported in Table 6.) We discuss the literature on distance sensitivity oracles extensively in Section 1.2.

[3] This means, for any two vertices $s, t \in V$ in the same connected component, we distinguish one s-t-path of minimum weight $d_G(s, t)$.

Table 2 Properties of the fault-tolerant ST-diameter oracles for undirected graphs obtained via Theorem 2 using single-source distance sensitivity oracles from the literature

Sensitivity	Stretch	Space	Query time	References
1	7	$\widetilde{O}(n^{3/2})$	$\widetilde{O}(1)$	[41, 42]
1	7	$\widetilde{O}(M^{1/2}n^{3/2})$	$\widetilde{O}(1)$	[41]
1	$7+\varepsilon$	$\widetilde{O}(n\log(W)/\varepsilon)$	$O(\log\log_{1+\varepsilon}(Wn))$	[17, 32, 43]
1	12	$O(n)$	$O(1)$	[43]
$f \geqslant 1$	$10f+7$	$\widetilde{O}(fn)$	$\widetilde{O}(f^3)$	[19]

The parameters are the same as in Table 1

The previous result requires access to a distance sensitivity oracle that can be queried with any pair of vertices. In contrast, single-source f-DSO are usually easier to compute. We show next how to use them in building ST-diameter oracles. In general, all parameters of the reduction are improved compared to Theorem 1, except for the stretch. The latter increases as a result of the lack of accuracy when the source of the underlying f-DSO is fixed. The literature on single-source f-DSOs is discussed in Section 1.3 and Table 2 lists the oracles obtained from previous works via Theorem 2.

Theorem 2 *Let $G = (V, E)$ be an undirected positively edge-weighted graph with n vertices and m edges. Let $S, T \subseteq V$ be two non-empty sets. Assume access to a single-source f-DSO for G with stretch σ, space $\mathtt{S}$, query time $\mathtt{Q}$, and preprocessing time $\mathtt{P}$. There exists an f-FDO-ST for G with stretch $2+5\sigma$, space $O(\mathtt{S}+n)$, query time $O(f\,(\log(f)+\mathtt{Q}))$, and preprocessing time $\widetilde{O}(\mathtt{P}+m)$.*

We now turn to the special case of the ST-diameter with a single source or target, that is, for $|S| = 1$ or $|T| = 1$, respectively. For the sake of clarity, whenever $S = \{s\}$ is a singleton, we use the term "sT-diameter" instead of "ST-diameter" or "$\{s\}T$-diameter"; similarly for $T = \{t\}$. In this setting, the underlying single-source f-DSOs really shine in that they allow to bring the stretch back down again, along with all other parameters of the reduction. Theorem 3 is phrased for the sT-diameter but holds verbatim also for the St-diameter case. Table 3 shows the resulting oracles.

Theorem 3 *Let $G = (V, E)$ be an undirected positively edge-weighted graph with n vertices and m edges. Let $s \in V$ be a vertex and $T \subseteq V$ a non-empty set. Assume access to a single-source f-DSO for G with stretch σ, space $\mathtt{S}$, query time $\mathtt{Q}$, and preprocessing time $\mathtt{P}$. There exists an f-FDO-sT for G with stretch $1+2\sigma$, space $\mathtt{S}+O(n)$, query time $O(f\,(\log(f)+\mathtt{Q}))$, and preprocessing time $\mathtt{P}+\widetilde{O}(m)$.*

The other special case we discuss is that of the regular diameter, that is, $S = T = V$. We provide two theorems showing how both all-pairs and single-source f-DSOs can be used to construct f-FDOs. Note that both results hold for directed graphs, provided that the underlying distance sensitivity oracle supports those. The corresponding oracles can be found in Tables 4 and 5.

Theorem 4 *Let G be an (undirected or directed) positively edge-weighted graph with n vertices and m edges. Assume access to an f-DSO for G with stretch σ, space $\mathtt{S}$,*

Table 3 Properties of the fault-tolerant sT-diameter oracle (resp. St-diameter oracle) for undirected graphs obtained via Theorem 3 using single-source distance sensitivity oracles from the literature

Sensitivity	Stretch	Space	Query time	References
1	3	$\widetilde{O}(n^{3/2})$	$\widetilde{O}(1)$	[41, 42]
1	3	$\widetilde{O}(M^{1/2}n^{3/2})$	$\widetilde{O}(1)$	[41]
1	$3+\varepsilon$	$\widetilde{O}(n\log(W)/\varepsilon)$	$O(\log\log_{1+\varepsilon}(Wn))$	[17, 32, 43]
1	5	$O(n)$	$O(1)$	[43]
$f\geqslant 1$	$4f+3$	$\widetilde{O}(fn)$	$\widetilde{O}(f^3)$	[19]

The parameters are the same as in Table 1

Table 4 Properties of the fault-tolerant diameter oracles obtained via Theorem 4 using all-pairs distance sensitivity oracles from the literature

Sensitivity	Stretch	Space	Query time	Ref.
1	2	$\widetilde{O}(n^2)$	$O(1)$	[29, 30]
1	2	$\widetilde{O}(n^2)$	$O(1)$	[31]
1	$(2k-1)(1+\varepsilon)+1$	$\widetilde{O}(k^5n^{1+1/k}/\varepsilon^4)$	$O(k)$	[32]
2	2	$\widetilde{O}(n^2)$	$\widetilde{O}(1)$	[12]
$f=O(1)$	$4+\varepsilon$	$\widetilde{O}(n^{2-\frac{\alpha}{2(f+1)}}(\log(n)/\varepsilon)^{O(f)})$	$O(n^{\frac{\alpha}{2}}/\varepsilon^2)$	[33]
$f=o(\frac{\log n}{\log\log n})$	2	$\widetilde{O}(n^{3-\alpha})$	$\widetilde{O}(f^2n^{2-(1-\alpha)/f})$	[24]
$f=o(\frac{\log n}{\log\log n})$	$2+\varepsilon$	$O(fn^{2+o(1)}\log(W)/\varepsilon^f)$	$\widetilde{O}(f^7\log\log W)$	[34]
$f\geqslant 1$	2	$O(fn^4)$	$f^{O(f)}$	[23]
$f\geqslant 1$	2	$O(n^{2+\alpha}M)$	$\widetilde{O}(f^4Mn^{2-\alpha}+f^{\omega+2}Mn)$	[36]
$f\geqslant 1$	$(8k-2)(f+1)+1$	$O\left(fkn^{1+1/k}\log(Wn)\right)$	$\widetilde{O}(f^3)$	[20]

The applicable graph class (un-/directed, un-/weighted) is determined by the f-DSO. The parameters are the same as in Table 1

query time $\mathtt{Q}$*, and preprocessing time* $\mathtt{P}$*. There exists an* f*-FDO for* G *with stretch* $1+\sigma$*, space* $\mathtt{S}+O(1)$*, query time* $O(f^2\mathtt{Q})$*, and preprocessing time* $\mathtt{P}+\widetilde{O}(mn)$*.*

Theorem 5 *Let* G *be an (undirected or directed) positively edge-weighted graph with* n *vertices and* m *edges. Assume access to a single-source* f*-DSO for* G *with stretch* σ*, space* $\mathtt{S}$*, query time* $\mathtt{Q}$*, and preprocessing time* $\mathtt{P}$*. There exists an* f*-FDO for* G *with stretch* $2+2\sigma$*, space* $O(\mathtt{S})$*, query time* $O(f\,\mathtt{Q})$*, and preprocessing time* $\widetilde{O}(\mathtt{P}+m)$*.*

Finally, we also prove an information-theoretic lower bound on the space requirement of approximate diameter oracles that support $f\geqslant 2$ edge failures. Note that the bound is given in bits, while usually we measure the space in the number of $O(\log n)$-bit machine words.

Theorem 6 *Any* f*-FDO or* f*-FDO-ST for* n*-vertex graphs with sensitivity* $f\geqslant 2$ *and a stretch of* $\frac{5}{3}-\varepsilon$ *for any* $\varepsilon=\varepsilon(n)>0$ *requires* $\Omega(n^{3/2})$ *bits of space.*

Outline. The remainder of this work is structured as follows. The next three subsections review the literature on diameter oracles as well as all-pairs and single-source

Table 5 Properties of the fault-tolerant diameter oracles obtained via Theorem 5 using single-source distance sensitivity oracles from the literature

Sensitivity	Stretch	Space	Query time	References
1	4	$\widetilde{O}(n^{3/2})$	$\widetilde{O}(1)$	[41, 42]
1	4	$\widetilde{O}(M^{1/2}n^{3/2})$	$\widetilde{O}(1)$	[41]
1	$4+\varepsilon$	$\widetilde{O}(n\log(W)/\varepsilon)$	$O(\log\log_{1+\varepsilon}(Wn))$	[17, 32, 43]
1	6	$O(n)$	$O(1)$	[43]
$f \geqslant 1$	$4f+4$	$\widetilde{O}(fn)$	$\widetilde{O}(f^3)$	[19]

The applicable graph class (un-/directed, un-/weighted) is determined by the single-source f-DSO. The parameters are the same as in Table 1

distance sensitivity oracles. We fix our notation in Section 2. Section 3 presents our constructions of f-FDO-ST for general sets $S, T \subseteq V$. In Section 4 and 5, we consider the respective special cases of a single source and of the unrestricted diameter. Section 6 proves the space lower bound.

1.1 Related Work on Fault-Tolerant Diameter Oracles

Fault-tolerant diameter oracles were introduced by Henzinger, Lincoln, Neumann, and Vassilevska Williams [25]. They showed that for a single failure in unweighted directed graphs, one can compute in time $\widetilde{O}(mn + n^{1.5}\sqrt{Dm/\varepsilon})$, where $\varepsilon \in (0, 1]$ and D is the diameter of the graph, a 1-FDO with $1+\varepsilon$ stretch that has $O(m)$ space, constant query time. Bilò, Cohen, Friedrich, and Schirneck [26] improved the preprocessing time to $\widetilde{O}(mn + n^2/\varepsilon)$, which is near-optimal assuming the combinatorial Boolean matrix multiplication conjecture (for details, see [25]). Using fast matrix multiplication instead, their preprocessing time for dense graphs decreases to $\widetilde{O}(n^{2.5794} + n^2/\varepsilon)$.

Bilò, Choudhary, Cohen, Friedrich, and Schirneck [27] addressed the problem of constructing 1-FDOs with $o(m)$ space. They showed that for unweighted directed graphs with sufficiently large diameter $D = \omega(n^{5/6})$, there is a 1-FDO taking $\widetilde{O}(n)$ space, with $1 + \frac{n^{5/6}}{D} = 1 + o(1)$ stretch, and $O(1)$ query time. It has a preprocessing time of $O(mn)$. In the same work, it was also shown that for any $\varepsilon > 0$ and graphs with diameter $D = \omega((n^{4/3}\log n)/(\varepsilon\sqrt{m}))$, there is a $(1+\varepsilon)$-stretch 1-FDO, with preprocessing time $O(mn)$, space $o(m)$, and constant query time.

For *undirected* graphs the space requirement can be reduced. There is a folklore construction that combines the 1-DSO by Bernstein and Karger [30] with the observation that in undirected graphs the eccentricity of an arbitrary vertex is a 2-approximation of the diameter. This results in an 1-FDO with stretch 2 and constant query time that takes only $O(n)$ space, see [25, 26].

For $f > 1$ edge failures in undirected graphs with non-negative edge weights, Bilò et al. [26] also presented an f-FDO with stretch $f+2$, an $O(f^2\log^2 n)$ query time, $\widetilde{O}(fn)$ space, and $\widetilde{O}(fm)$ preprocessing time.

We are not aware of any $O(n)$-sized, constant-stretch FDOs for *directed* graphs with arbitrary diameter in the literature prior to this work, not even for sensitivity

Table 6 Existing all-pairs distance sensitive oracles.

Sensitivity	Stretch	Space	Query time	Preprocessing Time	Ref.
1	1	$\widetilde{O}(n^2)$	$O(1)$	$\widetilde{O}(mn)$	[29, 30]
1	1	$\widetilde{O}(n^2)$	$O(1)$	$O(n^{2.529})$	[31]
1	$(2k-1)$ $(1+\varepsilon)$	$\widetilde{O}(k^5 n^{1+1/k}/\varepsilon^4)$	$O(k)$	$O(kmn^{1+1/k})$	[32]
2	1	$\widetilde{O}(n^2)$	$\widetilde{O}(1)$	poly(n)	[12]
$f = O(1)$	$3+\varepsilon$	$\widetilde{O}(n^{2-\frac{\alpha}{2(f+1)}}$ $(\log(n)/\varepsilon)^{O(f)})$	$O(n^{\frac{\alpha}{2}}/\varepsilon^2)$	$\widetilde{O}(mn^{2-\frac{\alpha}{2(f+1)}}$ $(\log(n)/\varepsilon)^{O(f)})$	[33]
$f = o(\frac{\log n}{\log\log n})$	1	$\widetilde{O}(n^{3-\alpha})$	$\widetilde{O}(n^{2-(1-\alpha)/f})$	$O(Mn^{\omega+1-\alpha})$	[24]
$f = o(\frac{\log n}{\log\log n})$	$1+\varepsilon$	$O(fn^{2+o(1)}$ $\log(W)/\varepsilon^f)$	$\widetilde{O}(f^5 \log\log W)$	$O(fn^{5+o(1)}$ $\log(W)/\varepsilon^f)$	[34]
$f \geqslant 1$	1	$O(fn^4)$	$f^{O(f)}$	$\Omega(n^f)$	[23]
$f \geqslant 1$	1	$O(f^4 n^2 \log(Mn))$	$O((cf$ $\log(Mn))^{O(f^2)})$	$\Omega(n^f)$	[35]
$f \geqslant 1$	1	$O(Mn^{2+\alpha})$	$\widetilde{O}(f^2 Mn^{2-\alpha}+$ $f^\omega Mn)$	$\widetilde{O}(Mn^{\omega+(3-\omega)\alpha})$	[36]
' $f \geqslant 1$	$(8k-2)$ $(f+1)$	$O\Big(fkn^{1+1/k}$ $\log(Wn)\Big)$	$\widetilde{O}(f)$	poly(n)	[20]

The reported features are for undirected graphs, including for those f-DSOs that also support directed graphs. The maximum edge weight for graphs with arbitrary positive weights is denoted W, for integer-weighted graphs it is M. The parameter $k \geqslant 1$ is a positive integer, $\varepsilon > 0$ a positive real, and $\alpha \in [0, 1]$ is a real number in the unit interval. We use $c > 1$ for a constant, and $\omega < 2.371339$ for the matrix multiplication exponent

$f = 1$. Also, no non-trivial f-FDOs with $o(f)$-stretch were known. To the best of our knowledge, we are the first to study the problem of general f-FDO-STs with $S, T \neq V$.

We now discuss the known space lower bounds for FDOs. Bilò, Cohen, Friedrich, and Schirneck [26] proved that f-FDOs with finite stretch must take $\Omega(fn)$ bits of space, which nearly matches their construction (see above). They also gave an $\Omega(m)$-bit bound for 1-FDOs with stretch $\sigma < 3/2$ for undirected *unweighted* graphs, or *edge-weighted* graphs when $\sigma < 2$. In a follow-up work, Bilò, Choudhary, Cohen, Friedrich, and Schirneck [27] generalized this to directed graphs. In particular, they showed that for directed unweighted graphs with diameter $D = O(n/\sqrt{m})$, any 1-FDO with stretch better than $\left(\frac{3}{2} - \frac{1}{D}\right)$ requires $\Omega(m)$ bits of space. They further proved that f-FDOs for digraphs require $\Omega(2^{f/2}n)$ bits of space, as long as $2^{f/2} = O(n)$.

1.2 All-Pairs Distance Sensitivity Oracles

We now discuss previous works on distance sensitivity oracles. They are summarized in Table 6. For simplicity, the features for undirected graphs are reported, even if the oracle also works on digraphs. This serves as the basis for Tables 1 and 4.

The first distance-sensitive oracle was given by Demetrescu and Thorup for directed graphs [44]. It maintains exact distances for a single edge failure ($\sigma = 1$ and $f = 1$). The space requirement of the oracle is $O(n^2 \log n)$ and its query time is $O(\log n)$. This was later generalized to also handle a single vertex failures by Demetrescu, Thorup, Chowdhury, and Ramachandran [21]. They presented an exact 1-DSO of size $O(n^2 \log n)$, with $O(1)$ query time and $\widetilde{O}(mn^2)$ preprocessing time. In two consecutive papers, Bernstein and Karger improved the preprocessing time first to $O(n^2\sqrt{m})$ in [29] and then to $\widetilde{O}(mn)$ in [30] (keeping the space and query time unchanged). Baswana and Khanna [32] considered approximate 1-DSOs for unweighted graphs. They devised a data structure of size $O(k^5 n^{1+1/k} \log^3(n)/\varepsilon^4)$, with stretch $(2k-1)(1+\varepsilon)$, and a $O(k)$ query time for any positive integer parameter k. Duan and Pettie [12] considered the case of two failures (vertices and edges) with exact distances. The size of their oracle is $O(n^2 \log^3 n)$, the query time is $O(\log n)$.

Relying on fast matrix multiplication, the 1-DSO by Chechik and Cohen [45] has a subcubic $\widetilde{O}(Mn^{2.873})$ preprocessing time for integer edge weights in the range $[-M, M]$ (that is, including negative values). Their oracle has an $\widetilde{O}(1)$ query time. This was later improved by Ren [46] as well as by Gu and Ren [47], who obtained an $\widetilde{O}(Mn^{2.5794})$ preprocessing time and constant query time. For unweighted directed graphs, Karczmarz and Sankowski [31] showed that one can construct in time $O(n^{2+\rho}) = O(n^{2.529})$ a 1-DSO with $O(1)$ query time and $O(n^2)$ space. Here, $\rho \approx 0.529$ is the solution of the equation $\omega(1, \rho, 1) = 1 + 2\rho$, where $\omega(a, b, c)$ is the smallest exponent such that one can multiply $n^a \times n^b$ and $n^b \times n^c$ matrices in $O(n^{\omega(a,b,c)})$ time. The data structure can also handle single vertex failures. The preprocessing time matches the best time known for computing APSP [48].

Regarding f-DSOs for larger values of f, Duan and Ren [23] gave an exact data structure for undirected weighted graphs with $O(fn^4)$ space, $f^{O(f)}$ query time. However, their construction has an $\Omega(n^f)$ preprocessing time. Later, Dey and Gupta [35] provided an alternative solution for undirected graphs with positive integer weights in $[1, M]$ with the same preprocessing time, but a better space of $O(f^4 n^2 \log(Mn))$ and a query time $O(c^{(f+1)^2} f^{8(f+1)^2} \log^{2(f+1)^2}(Mn))$ for some constant $c > 1$. The preprocessing times in [23, 35] are only polynomial for constant values of f.

Weimann and Yuster [24] presented an distance sensitivity oracle that can handle up to $f = o(\log n / \log\log n)$ edge or vertex failures with an $\widetilde{O}(n^{2-(1-\alpha)/f})$ query time and $O(Mn^{\omega+1-\alpha})$ preprocessing time for directed graphs with integer weights in the range $[-M, M]$. Here, $\alpha \in [0, 1]$ is a real number and $\omega < 2.371339$ the matrix multiplication exponent [49–51]. Grandoni and Vassilevska Williams [16] constructed a 1-DSO with subcubic $\widetilde{O}(Mn^{\omega+\frac{1}{2}} + Mn^{\omega+\alpha(4-\omega)})$ preprocessing time and sublinear $\widetilde{O}(n^{1-\alpha})$ query time. Inspired by the related topic of dynamic graphs, van den Brand and Saranurak [36] gave a distance sensitive oracle that can handle $f \leqslant \log n$ (batch) updates, where an update is an edge insertion or deletion. It has $\widetilde{O}(Mn^{\omega+(3-\omega)\alpha})$

preprocessing time, $\widetilde{O}(f^2Mn^{2-\alpha}+f^\omega Mn)$ update time, and an $\widetilde{O}(fMn^{2-\alpha}+f^2Mn)$ query time. The work of Karczmarz and Sankowski [31] also contains a data structure for unweighted directed graphs with $\widetilde{O}(n^\omega)$ preprocessing time and $O(n^2)$ space, such that for any set F of f edge or vertex failures, the data structure can be *updated* in time $O(f^{\omega-1}n)$ to then support distance queries with failures F in $O(fn)$ time.

We now discuss approximate f-DSOs. For undirected graphs with real edge weights bounded by W, Chechik, Langberg, Peleg, and Roditty [20] presented an f-DSO with stretch $(8k-2)(f+1)$, where again $k \geqslant 1$ is some integer parameter. Notably the stretch scales with the number f of supported edge failures. The oracle takes space $O(fkn^{1+1/k}\log(Wn))$ and has an $\widetilde{O}(f\log\log(Wn))$ query time. Later, Chechik, Cohen, Fiat and Kaplan [34] gave a solution whose stretch is independent of the sensitivity. Namely, for every approximation parameter $1 > \varepsilon > 1/n$ (including non-constant ones), they designed an $(1+\varepsilon)$-approximate f-DSO with a query time of $O(f^5\log(n)\log\log(W))$, space $O\big(fn^2\log(W)\cdot(c\log(n)/\varepsilon)^f\big)$, and $O\big(fn^5\log(W)\cdot(c\log(n)/\varepsilon)^f\big)$ preprocessing time. In the common case of a sensitivity $f = o(\log n/\log\log n)$, polynomial weights $W = \mathsf{poly}(n)$, and constant $1 > \varepsilon > 0$, their f-DSO has $1+\varepsilon$ stretch, takes $n^{2+o(1)}$ space, and has $\widetilde{O}(1)$ query time, and $n^{5+o(1)}$ preprocessing time.

Recently, Bilò, Chechik, Choudhary, Cohen, Friedrich, Krogmann, and Schirneck [33] designed the first f-DSO with subquadratic space, constant stretch, and truly sublinear query time for undirected unweighted graphs with unique shortest paths. More precisely, for constants $f \geqslant 2$ and $1/2 > \alpha > 0$, and for every $\varepsilon > 0$, their data structure has stretch $3+\varepsilon$, a space of $\widetilde{O}\big(n^{2-\frac{\alpha}{f+1}}\cdot(c\log n/\varepsilon)^{f+2}\big)$, query time of $O(n^\alpha/\varepsilon^2)$, and preprocessing time $\widetilde{O}\big(mn^{2-\frac{\alpha}{f+1}}\cdot(c\log n/\varepsilon)^{f+1}\big)$. In a follow-up work, Bilò, Chechik, Choudhary, Cohen, Friedrich, and Schirneck [52] improved the approximation ratio to (essentially) $1+\varepsilon$ multiplicative by introducing an *additive* stretch of 2. They kept the space subquadratic and the query sublinear by designing an f-DSO that, for any positive integer ℓ, real number $(\ell+1)/2 \geqslant \alpha > 0$, sensitivity $f = o(\log(n)/\log\log n)$, and $\varepsilon = \omega(\sqrt{\log n}/n^{\frac{\alpha}{2(\ell+1)(f+1)}})$, has stretch $((1+\frac{1}{\ell})(1+\varepsilon), 2)$, space $n^{2-\frac{\alpha}{(\ell+1)(f+1)}+o(1)}/\varepsilon^{f+2}$, query time $O(n^\alpha/\varepsilon^2)$, and preprocessing time $n^{2+\alpha+o(1)} + mn^{2-\frac{\alpha}{(\ell+1)(f+1)}+o(1)}/\varepsilon^{f+1}$.

1.3 Single-Source Distance Sensitivity Oracles

There are much fewer works on single-source DSOs. They are summarized in Table 7, which in turn underpins Tables 2, 3 and 5. We first discuss undirected graphs. Baswana and Khanna [32] showed that unweighted graphs can be preprocessed in $O(m\sqrt{n/\varepsilon})$ time to compute a $(1+\varepsilon)$-stretch single-source 1-DSO for edge and vertex failures. The oracle has size $O(n\log n + n/\varepsilon^3)$ and constant query time. For weighted graphs, they showed how an $O(n\log n)$ size oracle can report 3-approximate distances for a single failure in $O(1)$ time. Bilò, Gualà, Leucci, and Proietti [43] proved, for a single edge failure in weighted graphs, one can compute an $O(n)$-size oracle with stretch 2, maintaining a constant query time. Also, a construction was provided that has $1+\varepsilon$ stretch, with $O(n\log(1/\varepsilon)/\varepsilon)$ space and $O(\log(n)\log(1/\varepsilon)/\varepsilon)$ query time.

Table 7 Existing single-source distance sensitive oracles

Sensitivity	Stretch	Space	Query time	Preprocessing Time	Ref.
1	1	$\widetilde{O}(n^{3/2})$	$\widetilde{O}(1)$	$\widetilde{O}(mn^{1/2}+n^2)$	[41, 42]
1	1	$\widetilde{O}(M^{1/2}n^{3/2})$	$\widetilde{O}(1)$	$\widetilde{O}(Mn^{\omega})$	[41]
1	$1+\varepsilon$	$\widetilde{O}(n\log(W)/\varepsilon)$	$O(\log\log_{1+\varepsilon}(Wn))$	poly(n)	[17, 32, 43]
1	2	$O(n)$	$O(1)$	$\widetilde{O}(mn)$	[43]
$f \geqslant 1$	$2f+1$	$\widetilde{O}(fn)$	$\widetilde{O}(f^2)$	$\widetilde{O}(fm)$	[19]

The reported features are for undirected graphs, including for those f-DSOs that also support directed graphs. The parameters are the same as in Table 6

All the results stated until now were for a single edge or vertex failure. In a more recent work, Bilò, Gualà, Leucci, and Proietti [19] gave a construction for multiple failures that takes space $O(fn\log^2 n)$, and is computable in $\widetilde{O}(fm)$ time. The oracle reports $(2f+1)$-stretched distances in time $O(f^2\log^2 n)$.

Bilò, Cohen, Friedrich, and Schirneck [41] presented several additional single-source DSOs. For undirected unweighted graphs, they presented an oracle with $O(n^{3/2})$ space, query time $\widetilde{O}(1)$, and a $\widetilde{O}(m\sqrt{n}+n^2)$ preprocessing time. For the case that the graph has integer edge weights in the range $[1, M]$ and one is willing to use fast matrix multiplication, they presented an alternative construction with $O(M^{1/2}n^{3/2})$ space, query time $\widetilde{O}(1)$, and a $\widetilde{O}(Mn^{\omega})$ preprocessing time. Finally, for sufficiently sparse graphs with $m = O(M^{3/7}n^{7/4})$ edges they proved that a subquadratic-in-n preprocessing time is possible. Namely, they devised a single-source DSO with the same $O(M^{1/2}n^{3/2})$ size and $\widetilde{O}(1)$ query time, but a preprocessing time of $\widetilde{O}(M^{7/8}m^{1/2}n^{11/8})$.

For directed graphs, Baswana, Choudhary, Hussain, and Roditty [17] showed that one can preprocess graphs with edge weights in the range $[1, W]$ into an oracle with space $\widetilde{O}(n\log(W)/\varepsilon)$ that reports $(1+\varepsilon)$-approximate distances for a single edge/vertex failure in $\widetilde{O}(\log\log_{1+\varepsilon}(Wn))$ time. Gupta and Singh [42] designed an exact 1-DSO for directed unweighted graphs. The data structure has size $\widetilde{O}(n^{3/2})$ and $\widetilde{O}(1)$ query time.

2 Preliminaries

For a given graph $G = (V, E)$, possibly with positive edge weights, we denote by $d_G(u, v)$ the distance in G from vertex $u \in V$ to vertex $v \in V$. Given two non-empty subsets $S, T \subseteq V$, the *ST-diameter* of G is defined as $\mathrm{diam}(G, S, T) = \max_{s\in S, t\in T} d_G(s, t)$. With a slight abuse of notation, when $S = \{s\}$, we also use $\mathrm{diam}(G, s, T)$ as a shorthand of $\mathrm{diam}(G, \{s\}, T)$ for the sT-diameter. Likewise, we let $\mathrm{diam}(G, S, t)$ stand for $\mathrm{diam}(G, S, \{t\})$ in case $T = \{t\}$ is a singleton. If $S = T = V$, we use $\mathrm{diam}(G)$ instead of $\mathrm{diam}(G, V, V)$. The value $\mathrm{diam}(G, s, V) = \max_{v\in V} d_G(s, v)$ is the *eccentricity* of s in G.

For a set $F \subseteq E$ of edges, we denote by $G - F$ the graph obtained from G by removing all the edges in F. If H is a subgraph of G, we use $V(H)$ and $E(H)$ for the

vertices and edges of H, respectively. An *f-edge fault-tolerant distance sensitivity oracle* (f-DSO) is a data structure that receives queries (u, v, F) with $u, v \in V$ and $F \subseteq E$ with $|F| \leqslant f$. It returns an estimate $\widehat{d}_{G-F}(u, v)$ of the u-v-distance in $G - F$ such that $d_{G-F}(u, v) \leqslant \widehat{d}_{G-F}(u, v) \leqslant \sigma \cdot d_{G-F}(u, v)$, where $\sigma \geqslant 1$ is the *stretch*. The true quantity $d_{G-F}(u, v)$ is sometimes called the *replacement distance* from u to v.

An *f-edge fault-tolerant ST-diameter oracle* (f-FDO-ST) with stretch σ returns, upon query $F \subseteq E$ with $|F| \leqslant f$, an estimate $\widehat{D}$ of the ST-diameter of $G-F$ such that $\mathrm{diam}(G-F, S, T) \leqslant \widehat{D} \leqslant \sigma \cdot \mathrm{diam}(G-F, S, T)$. If $S = \{s\}$ or $T = \{t\}$ are singletons or $S = T = V$ are both the whole vertex set, we abbreviate the respective oracles as f-FDO-sT, f-FDO-St, and f-FDO. We sometimes use the term *fault-tolerant diameter* for $\mathrm{diam}(G-F)$ if the set F is clear from context.

3 *ST*-Diameter Oracles

Arguably, the technically most interesting construction is the reduction from fault-tolerant ST-diameter oracles to distance sensitivity oracles (Theorem 1 restated below). This and the proof of Theorem 3 in Section 4 contain many techniques that are also used to establish the other results later in Section 5.

Theorem 1 *Let $G = (V, E)$ be an undirected positively edge-weighted graph with n vertices and m edges. Let $S, T \subseteq V$ be two non-empty sets. Assume access to an f-DSO for G with stretch σ, space $\mathtt{S}$, query time $\mathtt{Q}$, and preprocessing time $\mathtt{P}$.*

(i) There exists an f-FDO-ST for G with stretch $1 + 3\sigma$, space $\mathtt{S} + O(n^2)$, query time $O(f^2(\log(f) + \mathtt{Q}))$, and preprocessing time $\mathtt{P} + \widetilde{O}(mn + n|S||T|)$.

(ii) If $f \leqslant \log_2(n)$, the space can be improved to $\mathtt{S} + O(2^{f/2} n^{3/2} \sqrt{\log n})$ at the expense of an $O(f^2(2^f + \mathtt{Q}))$ query time. The stretch and preprocessing time remain the same.

If additionally $M = \max(|S|, |T|) \leqslant 2^{f-1}$ holds, the space of the second variant further decreases to $\mathtt{S} + O(M^{1/2} n^{3/2} \sqrt{\log n})$, with a query time of $O(f^2(M + \mathtt{Q}))$.

In the following, we assume that the shortest paths in G are made unique. We can thus identify a shortest path with its endpoints. This allows us to save some preprocessing time and also makes the resulting data structure more space-efficient. In fact, it will turn out to be the key ingredient to keep the space overhead over the underlying f-DSO subquadratic (in n). However, the precise way how to make the paths unique influences the nature of the preprocessing. As hinted in Section 1, one can ensure a unique shortest path in a random fashion by slightly perturbing the edge weights. Alternatively, lexicographic perturbation [38–40] provides a deterministic procedure but adds an $\widetilde{O}(mn)$ term to the running time.

We elaborate on the lexicographic perturbation approach. Assume that each of the m edges is assigned a distinct integer label from the range $[1, m]$. For a path P, define the string $S_P \in \{0, 1\}^m$ as the characteristic vector of its edge set $E(P) \subseteq E$. That means, we have $S_P(i) = 1$ iff the edge with label i belongs to P. Given two different paths P_1 and P_2 of equal length between the same endpoints, we break the tie in favor

of P_1 if the string S_{P_1} is lexicographically smaller than S_{P_2}, that is, if at the first index i where $S_{P_1}(i) \neq S_{P_2}(i)$, we have $S_{P_1}(i) < S_{P_2}(i)$.

This ordering can be implemented efficiently when running Dijkstra's algorithm to compute a shortest-path tree from every vertex. Specifically, we maintain the current shortest-path tree in a top-tree data structure [53] augmented with a lowest common ancestor (LCA) structure, supporting path queries. Consider a relax(u, v) operation where $d_G(s, u) + w(u, v) = d_G(s, v)$. In this case, we break ties by comparing the candidate path obtained by extending the path from s to u with the edge (u, v), and the current path to v. Let $z = \text{LCA}(u, v)$. Using the top-tree structure, we can lexicographically compare these two paths by identifying the maximum-labeled edge along each path segment beyond z, and selecting the path with the smaller such maximum-label, and update the top-trees and LCA data structures accordingly. A description of the lexicographic perturbation technique with full details can be found for example in [38, Section 6.2]. In the following, whenever we refer to *the* shortest path between vertices s and t, we mean the respective path in the tree rooted at the source s.

Let us first give a coarse outline of the remaining section. We define two sets S_F and T_F depending on S, T and the failure sets F. We then show how they can be used to estimate the fault-tolerant ST-diameter. The main part of the section consists of constructing an auxiliary data structure that efficiently computes S_F and T_F. We then show how to reduce the space of that structure, albeit at the expense of a higher query time.

Fix a set $F \subseteq E$ of at most f edges and recall that we use $V(F)$ to denote the set of endpoints of edges in F. We distinguish two subsets of $V(F)$, S_F and T_F. Intuitively, a vertex v belongs to S_F if it is on some shortest ST-path and the portion of that path from the source to v survives even after the failures; likewise, v is in T_F if the portion from it to the target is not affected by failing edges. For the exact definition, let $\pi_{x,y}$ denote the unique shortest path from vertex x to y in G. Some $v \in V(F)$ is in S_F if there exist $s \in S$ and $t \in T$ such that v is on the shortest path $\pi_{s,t}$ and the subpath from s to v contains no failing edge. The definition of T_F is analogous with t in place of s.

We use two crucial properties of the sets S_F and T_F and their elements. First, let $v \in S_F$ be such an element as witnessed by some $s \in S$ and $t \in T$. The prefix of $\pi_{s,t}$ from s to v has length[4] $d_G(s, v)$ and, since $v \in S_F$, this prefix does not contain any edge of F. This gives $d_{G-F}(s, v) = d_G(s, v)$. Similarly, we have $d_{G-F}(v, t) = d_G(v, t)$ for $v \in T_F$ and their respective targets t. Secondly, S_F and T_F are subsets of $V(F)$ and *not* of S and T. This will help to reduce the query times tremendously. The cardinalities $|S_F|, |T_F|$ are in $O(f)$ while S and T may be much larger. The core of this section is to prove that the information in S_F, T_F is enough to estimate the fault-tolerant ST-diameter.

[4] Since the shortest paths are unique, the prefix of $\pi_{s,t}$ from s to v *is* the shortest path $\pi_{s,v}$, but this stronger property is not needed here.

3.1 Query Algorithm

Before we describe how S_F and T_F are computed and stored, we present the query algorithm of our ST-diameter oracle. Let $\mathcal{D}$ denote the f-DSO with stretch σ that is assumed in Theorem 1. Let F be the query set and suppose S_F and T_F to be given. For every pair of vertices $(u, v) \in S_F \times T_F$, the diameter oracle queries $\mathcal{D}$ with the triple (u, v, F) to obtain a σ-approximation of $d_{G-F}(u, v)$. The f-FDO-ST returns the value

$$\widehat{D} = \operatorname{diam}(G, S, T) + \max_{(u,v)\in S_F\times T_F} \mathcal{D}(u, v, F).$$

The value $\operatorname{diam}(G, S, T)$ can be precomputed. So given S_F and T_F, the time needed to obtain $\widehat{D}$ is $O(f^2\mathrm{Q})$, where Q is the query time of the f-DSO $\mathcal{D}$.

We next prove the stretch of our oracle. Using the sets S_F and T_F in place of S and T have the advantage that they make the query time independent of the sizes $|S|$ and $|T|$. However, this bears the danger that the oracle may overestimate the ST-diameter of $G-F$. After all, for arbitrary vertices $u \notin S$ and $v \notin T$, the distance $d_{G-F}(u, v)$ may not be bounded by any multiple of $\operatorname{diam}(G-F, S, T)$. The crucial part in the proof of the next lemma is to show that in the specific case $u \in S_F$ and $v \in T_F$, $d_{G-F}(u, v)$ is at most three times the fault-tolerant ST-diameter.

Lemma 7 *The f-FDO-ST has stretch $1 + 3\sigma$.*

Proof Let $s \in S$ and $t \in T$ be two vertices. We first show $d_{G-F}(s, t) \leqslant \widehat{D}$. That means, the returned value never underestimates the ST-diameter of $G-F$. We only need to prove the case in which some of the failing edges in F belong to $\pi_{s,t}$; otherwise, $d_{G-F}(s, t) = d_G(s, t) \leqslant \operatorname{diam}(G, S, T) \leqslant \widehat{D}$. Consider all failing edges on $\pi_{s,t}$ and let u^* be the endpoint of such an edge that is closest to s. Also, let v^* be the endpoint closest to t among all edges in $F \cap E(\pi_{s,t})$. We thus have $u^* \in S_F$ and $v^* \in T_F$ and $d_{G-F}(s, u^*) = d_G(s, u^*)$ as well as $d_{G-F}(v^*, t) = d_G(v^*, t)$. Since π_{s,u^*} and $\pi_{v^*,t}$ are vertex-disjoint subpaths of $\pi_{s,t}$, we get

$$d_{G-F}(s, u^*) + d_{G-F}(v^*, t) = d_G(s, u^*) + d_G(v^*, t) \leqslant d_G(s, t) \leqslant \operatorname{diam}(G, S, T).$$

Recall that the distance sensitivity oracle $\mathcal{D}$ satisfies $d_{G-F}(u, v) \leqslant \mathcal{D}(u, v, F)$ for all admissible queries (u, v, F). Since $(u^*, v^*) \in S_F \times T_F$, we have

$$d_{G-F}(u^*, v^*) \leqslant \max_{(u,v)\in S_F\times T_F} d_{G-F}(u, v) \leqslant \max_{(u,v)\in S_F\times T_F} \mathcal{D}(u, v, F).$$

Combining the two facts gives

$$\begin{aligned} d_{G-F}(s, t) &\leqslant d_{G-F}(s, u^*) + d_{G-F}(u^*, v^*) + d_{G-F}(v^*, t) \\ &\leqslant \operatorname{diam}(G, S, T) + \max_{(u,v)\in S_F\times T_F} \mathcal{D}(u, v, F) = \widehat{D}. \end{aligned}$$

We now prove the other inequality $\widehat{D} \leqslant (1+3\sigma) \cdot \operatorname{diam}(G-F, S, T)$, where σ is the stretch of $\mathcal{D}$. Let $u \in S_F$ and $v \in T_F$ be two vertices. Our claim will follow

once we have established that $d_{G-F}(u, v) \leqslant 3 \cdot \operatorname{diam}(G-F, S, T)$. By definition, the reason why u is in S_F is that there exist $s_u \in S$ and $t_u \in T$ such that u is on the shortest s_u-t_u-path π_{s_u,t_u} in G and $d_{G-F}(s_u, u) = d_G(s_u, u)$. We get

$$d_{G-F}(s_u, u) = d_G(s_u, u) \leqslant d_G(s_u, t_u) \leqslant \operatorname{diam}(G, S, T).$$

By the same argument, the fact $v \in T_F$ is witnessed by a vertex $t_v \in T$ with $d_{G-F}(v, t_v) \leqslant \operatorname{diam}(G, S, T)$. Now note that the graph $G-F$ is undirected, which means $d_{G-F}(s_u, u) = d_{G-F}(u, s_u)$. The ST-diameter of $G-F$ is also never smaller than that of G. Thus,

$$\begin{aligned} d_{G-F}(u, v) &\leqslant d_{G-F}(u, s_u) + d_{G-F}(s_u, t_v) + d_{G-F}(t_v, v) \\ &\leqslant 2 \cdot \operatorname{diam}(G, S, T) + \operatorname{diam}(G-F, S, T) \leqslant 3 \cdot \operatorname{diam}(G-F, S, T). \end{aligned}$$

The final bound on the stretch follows from the fact that the f-DSO overestimates any fault-tolerant distance by at most a factor σ,

$$\begin{aligned} \widehat{D} &= \operatorname{diam}(G, S, T) + \max_{(u,v) \in S_F \times T_F} \mathcal{D}(u, v, F) \\ &\leqslant \operatorname{diam}(G, S, T) + \max_{(u,v) \in S_F \times T_F} \sigma \cdot d_{G-F}(u, v) \\ &\leqslant \operatorname{diam}(G, S, T) + \sigma \cdot 3 \operatorname{diam}(G-F, S, T) \leqslant (1 + 3\sigma) \operatorname{diam}(G-F, S, T). \end{aligned}$$

□

3.2 Basic Data Structure for S_F and T_F

Given the failure set F, the set S_F contains all $v \in V(F)$ such that there exists $s \in S$ and $t \in T$ for which v is the closest endpoint to s of a failing edge on the shortest path $\pi_{s,t}$. The set T_F is defined analogously. We now describe the data structure that computes S_F and T_F. The exposition focuses on S_F, the structure for T_F uses the same ideas.

The first variant of our construction takes space $O(n^2)$ in addition to the space S of the underlying f-DSO. For any vertex $v \in V$, let $\widetilde{T}_v$ denote the shortest path tree of the graph G rooted in v. We use the notation $\widetilde{T}_v$ to avoid confusion with the sets T and T_F. The preprocessing algorithm of our data structure computes, for each $v \in V$, the tree $\widetilde{T}_v$ and marks certain nodes in it. A node $s \in S$ is marked in the tree $\widetilde{T}_v$ if there is some $t \in T$ such that v lies on the shortest path $\pi_{s,t}$. For any two vertices $s \in S$ and $t \in T$, $\pi_{s,t}$ contains v if and only if $d_G(s, t) = d_G(s, v) + d_G(v, t)$. We used here that the shortest paths are unique. The algorithm can thus compute all-pairs distances in G in time[5] $O(mn + n^2 \log n)$ and mark the relevant nodes in *all* trees with the obvious $O(n|S||T|)$-time algorithm.

[5] The time needed for this step reduces to $O(mn)$ in case G is unweighted, has only small integer weights, or floating points in exponent-mantissa representation using Thorup's algorithm [54].

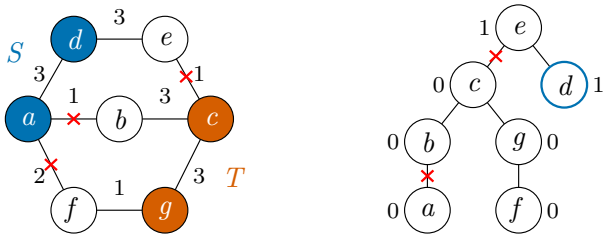

Fig. 2 Illustration of the basic data structure for the set S_F. On the left is the example graph with failing edges $\{a, b\}$, $\{a, f\}$, and $\{c, e\}$. On the right is the shortest-path tree $\widetilde{T}_e$ rooted in e. The nodes are annotated with their *count*. Only the node $d \in S$ is marked as none of the shortest paths that start in $a \in S$ and end in some vertex of T use e. The set F_0 consists of the edges $\{a, b\}$ and $\{c, e\}$, the former is dominated by the latter. Since the marked node d remains reachable from e in $\widetilde{T}_e - F_0$, the root e is included in the set S_F

Additionally, each node u of $\widetilde{T}_v$ is annotated with its pre- and post-order traversal number and the value $count_v(u)$, the latter being the number of marked nodes in the subtree $(\widetilde{T}_v)_u$ rooted in u. Each node except for the root v also stores a pointer to its parent. The annotations do not increase the preprocessing time or storage space by more than a constant factor. In particular, for a fixed tree $\widetilde{T}_v$, all values $count_v(u)$ are computable in $O(n)$ time in bottom-up fashion. The total time needed for preprocessing is thus $O(mn + n^2 \log n + n|S||T|)$ and the space is $O(n^2)$.

To answer a query F, the algorithm scans all the vertices $v \in V(F)$ and decides which of them to include in S_F. An illustration is given in Figure 2. The graph $\widetilde{T}_v - F$ is a forest of rooted trees. Possibly some of the trees degenerated into isolated vertices. We claim that $v \in S_F$ holds if and only if $\widetilde{T}_v - F$ contains a marked node that is still reachable from v. To see this, observe that the uniqueness of shortest paths implies their consistency. That means, if v is on some shortest path $\pi_{s,t}$, then the subpath from s to v is the shortest path $\pi_{s,v}$ (and the one from v to t is $\pi_{v,t}$). Since G is undirected, $\pi_{s,v}$ is contained in $\widetilde{T}_v$. Now consider the two defining conditions for $v \in S_F$. Vertex v lying on the path $\pi_{s,t}$ means $s \in S$ is marked, and $\pi_{s,v}$ not having a failing edge is equivalent to s being reachable in $\widetilde{T}_v - F$.

To check whether $v \in S_F$, the algorithm first computes the set $F_0 = F \cap E(\widetilde{T}_v)$ of those edges whose failure affect a shortest path starting in v. An edge $\{u, w\} \in F$ is in F_0 iff the parent of u in $\widetilde{T}_v$ is w or the parent of w is u. When talking about tree edges, we adopt the notation $(u, w) \in F_0$ to indicate that u is the parent of w. The edges are still meant to be undirected. Next, we define a notion of domination among edges in F_0. We say that an edge $(x, y) \in F_0$ is *dominated* by another edge $(u, w) \in F_0$ if (x, y) is in the subtree $(\widetilde{T}_v)_w$. We show how to identify all dominated edges in time $O(f \log f)$ using the tree-traversal numbers stored during preprocessing. For a node u, let $pre(u)$ denote the pre-order number of u and $post(u)$ its post-order number. We further say x is *a parent node* if there exists a y such that $(x, y) \in F_0$, that is, x is *the* parent node in a failing edge in $\widetilde{T}_v$. Likewise, we say w is *a child node* if there is some edge $(u, w) \in F_0$. These notions are not mutually exclusive. An edge in F_0 is dominated iff its parent node x has some child node as an ancestor (which may be x itself). Finally, some child node w is an ancestor of x if and only if $pre(w) \leqslant pre(x)$ and $post(w) \geqslant post(x)$.

To exploit this connection, we first sort the full set $V(F_0)$ by descending pre-order number, resulting in the list *pre-all*$^{\downarrow}$. Let further *post-child*$^{\downarrow}$ denote only the child nodes ordered by descending post-order number.[6] We scan through *pre-all*$^{\downarrow}$ consuming both lists in the process. Let x be the first entry of *pre-all*$^{\downarrow}$. If it is not a parent node (i.e., only a child node and not both), then remove x from both lists. If x is a parent node, check the current-first entry w of *post-child*$^{\downarrow}$. Since w has not been removed yet, it is a child with $pre(w) \leqslant pre(x)$. Therefore, the edge $(x, y) \in F_0$ is dominated if and only if $post(w) \geqslant post(x)$. After this check, x is again removed from both lists (if present) and we continue with the new first entry of *pre-all*$^{\downarrow}$. The bottleneck is to sort the two lists in time $O(|V(F_0)| \cdot \log |V(F_0)|) = O(f \log f)$. We denote the set of the remaining non-dominated edges by F_0'.

Recall that we include the vertex v in S_F if the number of marked node in the connected component of v in $\widetilde{T}_v - F$ is positive. Clearly, this connected component is the same in $\widetilde{T}_v - F_0$ and even in $\widetilde{T}_v - F_0'$, i.e., when ignoring dominated edges. We now use the stored *count*s. The value $count_v(w)$ is the number of marked nodes in the subtree $(\widetilde{T}_v)_w$ *before* failures. No two edges in F_0' dominate each other. Hence, the subtrees of $\widetilde{T}_v$ rooted at the child nodes of edges in F_0' span exactly the vertices that are *not* in the same connected component as v in $\widetilde{T}_v - F_0'$. By summing the *count*-values over all child nodes[7] and comparing it to $count_v(v)$, we get the desired information.

Since we have to construct the set F_0' for each root $v \in V(F)$ separately, computing S_F takes total time $O(f^2 \log f)$, same for T_F.

3.3 A More Compact Data Structure

We now design an alternative data structure that has a space overhead of $O(2^{f/2} n^{3/2} \sqrt{\log n})$, rather than $O(n^2)$, provided that $f \leqslant \log_2(n)$. This will, however, increase the query time. The idea is to store the information of the trees $\widetilde{T}_v$ in a more compact way. For every vertex $v \in V$, we define a new representation $\mathcal{T}_v$ of the tree $\widetilde{T}_v$ by first removing unnecessary parts and then replacing long paths with single edges. This corresponds to the two steps of the compression described below. For the first one, we need the following definition. We say a subtree $\mathcal{T}_v$ of $\widetilde{T}_v$ preserves the *source-to-leaf reachability* if, for every set F of up to f edges, there is a marked node of $\widetilde{T}_v$ that is reachable from the source v in $\widetilde{T}_v - F$ if and only if there is a leaf of $\mathcal{T}_v$ that is reachable from v in $\mathcal{T}_v - F$.

First Compression Step

Our goal is to preserve the source-to-leaf reachability when compressing $\widetilde{T}_v$. Let $S' \subseteq S$ be the set of nodes that are marked in $\widetilde{T}_v$. Using an algorithm by Petruschka [55], we compute in time $O(n + |S'|) = O(n)$ a set $\mathcal{L}_v \subseteq V$ of at most 2^{f-1} nodes such that, for every $F \subseteq E$ with $|F| \leqslant f$, a node of S' is reachable from v in $\widetilde{T}_v - F$ iff a node in $\mathcal{L}_v$ is still reachable.[8] The main difference between S' and $\mathcal{L}_v$ is that the

[6] The list *post-child*$^{\downarrow}$ only containing child nodes ensures that we find the remaining child node with maximum post-order number in constant time.

[7] The reason to compute the subset F_0' of non-dominated edges is to avoid double-counting here.

[8] The result in [55] is stated in terms of vertex failures. We can adapt it by subdividing every edge in $\widetilde{T}_v$ and applying it to a failure set of up to f of those new "edge-vertices".

size of the latter is bounded by a function of f. Of course, if $|S'| \leqslant 2^{f-1}$, we can forgo this step entirely. This is particularly relevant if already S and T have fewer than 2^{f-1} elements. We define the tree representation $\mathcal{T}_v$ as the smallest subtree of $\widetilde{T}_v$ that contains the root v and $\mathcal{L}_v$. The notation $\mathcal{L}_v$ is mnemonic of "leaves". It is clear from the construction of $\mathcal{L}_v$ that $\mathcal{T}_v$ preserves the root-to-leaf reachability of $\widetilde{T}_v$.

Second Compression Step

After the first compression step, the tree $\mathcal{T}_v$ contains at most 2^{f-1} leaves. It may still be the case that the total number of nodes is large due to the presence of very long paths connecting two branch nodes, that is, nodes with two or more children. We now contract those paths to single edges.

We use the critical path framework of Alon, Chechik, and Cohen [14]. Let x and y be two consecutive branch nodes of $\mathcal{T}_v$, i.e., x is an ancestor of y in $\mathcal{T}_v$ and the internal nodes of the path $\pi_{x,y}$ all have degree 2 in $\mathcal{T}_v$. We say that $\pi_{x,y}$ is *critical* if it has at least $L = \sqrt{n \log_2(n)/2^{f-1}}$ edges. If so, we replace $\pi_{x,y}$ in $\mathcal{T}_v$ with the *representative edge* $\{x, y\}$ and add $\pi_{x,y}$ to the set $\mathcal{P}$ of critical paths. This collection can be computed in total time $O(n^2)$ by depth-first searches in each of the n trees. Eventually, it contains at most $q = O(n^2/L) = O(2^{f/2} n^{3/2}/\sqrt{\log n})$ paths as each tree contributes $O(n/L)$ paths. A deterministic hitting set B for $\mathcal{P}$ can be computed in time $\widetilde{O}(qL + n^2/L) = \widetilde{O}(n^2)$. It has cardinality

$$|B| = O\left(n \frac{\log q}{L}\right) = O\left(2^{f/2} n^{1/2} \left(\sqrt{\log n} + \frac{f}{\sqrt{\log n}}\right)\right)$$
$$= O\left(2^{f/2} n^{1/2} \sqrt{\log n}\right).$$

The last estimate uses the assumption $f \leqslant \log_2(n)$.

For each *pivot* $b \in B$, we store the shortest-path tree $\widetilde{T}_b$ of the original graph G rooted in b. Note that a critical path $\pi_{x,y} \in \mathcal{P}$ is contained in $\widetilde{T}_b$ for any pivot b that hits $\pi_{x,y}$. Moreover, this b is the lowest common ancestor of x and y in the tree $\widetilde{T}_b$.

The final data structure consists of the trees $\{\mathcal{T}_v\}_{v \in V}$ and $\{\widetilde{T}_b\}_{b \in B}$ with some small amount of extra information. For each tree ($\widetilde{T}_b$ and $\mathcal{T}_v$ alike), we store a data structure that answers lowest common ancestor (LCA) queries in constant time. Such a structure can be constructed in time/space that is linear in the size of the tree [56]. The trees $\mathcal{T}_v$ additionally have a dictionary for the set of their representative edges. Since there are at most $O(2^f)$ branch nodes in any tree, this also upper bounds the number of representative edges. The dictionaries can be implemented to take space $O(2^f)$ and have a constant worst-case query time [57]. For any representative edge $\{x, y\}$ of $\mathcal{T}_v$, we additionally have a pointer to the tree $\widetilde{T}_b$ of an (arbitrary) pivot b that hits $\pi_{x,y}$. After the second compression step, any tree $\mathcal{T}_v$ contains at most $O(2^f L) = O(2^{f/2} n^{1/2} \sqrt{\log n})$ nodes. Since there are n such trees, the LCA structures, and dictionaries take $O(2^{f/2} n^{3/2} \sqrt{\log n})$ space. Likewise, the trees $\widetilde{T}_b$ for all the pivots in B take $O(|B|n) = O(2^{f/2} n^{3/2} \sqrt{\log n})$ space. If $M = \max(|S|, |T|)$ is at most 2^{f-1}, then, each tree $\mathcal{T}_v$ has $|\mathcal{L}_v| \leqslant M$ leaves and we can replace every occurrence of 2^{f-1} in the above derivation by M.

Regarding the preprocessing time, running Petruschka's algorithm [55] for all n trees $\widetilde{T}_v$ (if necessary) and obtaining the pivots B takes total time $\widetilde{O}(n^2)$. Computing the LCA structures for the trees $\{\mathcal{T}_v\}_{v\in V}$ and $\{\widetilde{T}_b\}_{b\in B}$ can be done in time that is linear in their total size, i.e., $O(2^{f/2}n^{3/2}\sqrt{\log n}) = \widetilde{O}(n^2)$. Therefore, the $\widetilde{O}(mn+n|S||T|)$ time to construct and mark the original trees $\{\widetilde{T}_v\}_v$ (see Section 3.2) still dominates the preprocessing.

Consider a set F of at most f failing edges. We now explain how to compute S_F from the compressed trees in time $O(2^f f^2)$ (resp., in time $O(f^2M)$). The set S_F are those vertices $v \in V(F)$ such that there is a shortest s-t-path $\pi_{s,t}$ with $s \in S$ and $t \in T$ that contains v, but the prefix $\pi_{s,v}$ is free from failing edges. Since the root-to-leaf reachability is preserved, this is equivalent to having a leaf in $\mathcal{L}_v$ that is reachable from v in $\mathcal{T}_v - F$. Conceptually speaking, we can safely remove all non-representative edges in $\mathcal{T}_v$ that are in F. For the representative edges $\{x, y\}$, we have to look up information in the *other* tree $\widetilde{T}_b$ with root $b \in B$ associated with $\{x, y\}$. That edge is removed from $\mathcal{T}_v$ if there is a failing edge in F that is contained in the path $\pi_{x,y}$ in $\widetilde{T}_b$. We then return to $\mathcal{T}_v$ check whether some leaf from $\mathcal{L}_v$ is still connected to the root.

In more detail, we first compute the non-representative failing edges in $\mathcal{T}_v$.[9] For some edge $\{u, w\} \in F$, we check whether it is in the dictionary of representative edges of $\mathcal{T}_v$. If not, we test whether the LCA of u and w in $\mathcal{T}_v$ is in $\{u, w\}$. If so, we have $\{u, w\} \in E(\mathcal{T}_v)$. This can be done for all of F with $O(f)$ constant-time look-ups and LCA queries. To handle an representative edges $\{x, y\}$, we follow the pointer to the tree $\widetilde{T}_b$. The edge $\{u, w\}$ lies on $\pi_{x,y}$ in $\widetilde{T}_b$ if and only if one of two conditions hold:

(*i*) the LCA of u and x is u and the LCA of w and x is w;
(*ii*) the LCA of u and y is u and the LCA of w and y is w.

The whole check takes time $O(2^f f)$ since there are at most $O(2^f)$ representative edges per tree. Slightly abusing notation, let F_0 be the set of edges removed from $\mathcal{T}_v$, including possibly some representative edges. F_0 may no longer be a subset of F, but we still have $|F_0| \leqslant |F|$. We test for each leaf in $\mathcal{L}_v$ and each edge in F_0 whether the latter lies on the path from the root v to that leaf. This takes a constant number of LCA queries per test and thus $O(2^f f)$ time. In summary, the tree $\mathcal{T}_v$ can be queried in time $O(2^f f)$ and we have to do this for every $v \in V(F)$, resulting in a $O(2^f f^2)$ query time.

4 Single-Source *sT*-Diameter Oracles

The next question we address is how to construct sT-diameter oracles, that is, a data structure that reports maximum replacement distance from a single source s to a set of vertices T. To build the f-FDO-sT, we use a single-source distance sensitivity oracle, giving us access to estimates of all replacement distances from s. We restate the relevant theorem below. Its proof uses similar ideas as those shown in Section 3, but the single-source setting allows for improvements in all parameters.

[9] In Section 3.2, we used the parent pointers of individual nodes for this task. The same solution is possible here and would increase the total space requirement only by a constant factor.

Theorem 3 *Let $G = (V, E)$ be an undirected positively edge-weighted graph with n vertices and m edges. Let $s \in V$ be a vertex and $T \subseteq V$ a non-empty set. Assume access to a single-source f-DSO for G with stretch σ, space $\mathtt{S}$, query time $\mathtt{Q}$, and preprocessing time $\mathtt{P}$. There exists an f-FDO-sT for G with stretch $1 + 2\sigma$, space $\mathtt{S} + O(n)$, query time $O(f\,(\log(f) + \mathtt{Q}))$, and preprocessing time $\mathtt{P} + \widetilde{O}(m)$.*

Proof Let $\mathcal{D}$ denote the underlying single-source f-DSO. The preprocessing algorithm for the f-FDO-sT constructs $\mathcal{D}$ with source s and computes the shortest-path tree $\widetilde{T}_s$ of G rooted in s. Each node $v \in V$ is annotated with a pointer to its parent node in $\widetilde{T}_s$ and its respective number in the pre-order and post-order traversal of the tree. Similarly as above, the algorithm also computes the value $count(v)$ for every v. This is the number of all vertices in the set T that are descendants of v in $\widetilde{T}_s$, possibly including v itself. (A difference to Section 3.3 is that no marking of vertices is needed here.) Finally, the preprocessing algorithm stores the maximum distance $\mathrm{diam}(G, s, T) = \max_{t \in T} d_G(s, t)$ from the root to the vertices in the set T. The preprocessing takes total time $\mathtt{P} + O(m + n \log n)$ in general weighted graphs and, again, can be reduced to $\mathtt{P} + O(m)$ for certain classes of weights [54]. Storing the oracle and the tree takes $\mathtt{S} + O(n)$ space.

For the query, let $F \subseteq E$ be a set of up to f failing edges and $F_0 = F \cap E(\widetilde{T}_s)$ those failures that are in the tree. Consider the forest of rooted trees $\widetilde{T}_s - F_0$ and let R_F be the set of roots of those trees that contain some vertex from T. For any $v \in V$, let $\mathcal{D}(v, F)$ be the σ-approximation of the replacement distance $d_{G-F}(s, v)$ computed by the DSO $\mathcal{D}$. Our sT-diameter oracle answers the query F by reporting

$$\widehat{D} = \mathrm{diam}(G, s, T) + \max_{r \in R_F} \mathcal{D}(r, F).$$

We defer the discussion of how to obtain the set R_F and first prove that $\widehat{D}$ is an $(1+2\sigma)$-approximation of $\mathrm{diam}(G-F, s, T)$. Let $t \in T$ be a target vertex and $r \in R_F$ the root of the subtree in $\widetilde{T}_s - F_0$ that contains t. By the choice of r, we have $d_{G-F}(r, t) = d_G(r, t)$. Also, it holds that $d_{G-F}(s, r) \leqslant \mathcal{D}(r, F)$ since the oracle $\mathcal{D}$ never underestimates the replacement distance. Combining this with the definition of our estimate $\widehat{D}$ gives

$$\begin{aligned} d_{G-F}(s, t) &\leqslant d_{G-F}(s, r) + d_{G-F}(r, t) = d_{G-F}(s, r) + d_G(r, t) \\ &\leqslant d_{G-F}(s, r) + d_G(s, t) \leqslant \mathcal{D}(r, F) + \mathrm{diam}(G, s, T) \leqslant \widehat{D} \end{aligned}$$

For the last inequality, we need the maximization over all roots in R_F in the definition of our estimate $\widehat{D}$. As the above argument holds for any target t, we get $\mathrm{diam}(G-F, s, T) \leqslant \widehat{D}$.

For the other direction, let $r^* = \arg\max_{r \in R_F} \mathcal{D}(r, F)$ be the root that maximizes the *approximate* replacement distance from s returned by $\mathcal{D}$. Let further $t \in T$ be some target vertex in the subtree of $\widetilde{T}_s - F_0$ that is rooted in r^*. We thus have

$$d_{G-F}(s, r^*) \leqslant d_{G-F}(s, t) + d_G(t, r^*) \leqslant d_{G-F}(s, t) + d_G(t, s) \leqslant 2 \cdot d_{G-F}(s, t).$$

The first inequality uses that G is undirected, so that we can go "up" the tree from t to r^*. Here it is important that we do not just use any root in $\widetilde{T}_s - F_0$, but one that actually has some target vertex in its tree. The last inequality stems from the replacement distance from s to t being never smaller than the original s-t-distance.

We combine this with the fact that $\mathcal{D}(r^*, F)$ is a σ-approximation of the true replacement distance $d_{G-F}(s, r^*)$.

$$\begin{aligned}\widehat{D} = \operatorname{diam}(G, s, T) + \mathcal{D}(r^*, F) &\leqslant \operatorname{diam}(G, s, T) + \sigma \cdot d_{G-F}(s, r^*) \\ &\leqslant \operatorname{diam}(G, s, T) + 2\sigma \cdot d_{G-F}(s, t) \leqslant (1 + 2\sigma) \cdot \operatorname{diam}(G-F, s, T).\end{aligned}$$

In summary, we have $\operatorname{diam}(G-F, s, T) \leqslant \widehat{D} \leqslant (1 + 2\sigma) \cdot \operatorname{diam}(G-F, s, T)$, as desired.

The forest $\widetilde{T}_s - F_0$ consists of $|F_0| + 1 \leqslant f + 1$ trees, so the set R_F contains at most this many roots. Given R_F, computing $\widehat{D}$ takes time $O(f\mathrm{Q})$, where Q is the query time of the oracle $\mathcal{D}$. It remains to show how to compute R_F from F in time $O(f \log f)$. Recall that we know the parent of every non-root node in $\widetilde{T}_s$. We use it to first obtain F_0 from F in time $O(f)$ as an edge $\{a, b\}$ is in $\widetilde{T}_s$ if and only if a is parent of b or vice versa.

For any failing edge $e \in F_0$, let $r(e)$ be the endpoint of e that is farther from the source s. Note that the roots of the trees in $\widetilde{T}_s - F_0$ are either s or such a "lower" endpoint of a failing edge. We use $R = \{r(e) \mid e \in F_0\} \cup \{s\}$ to denote them. The subset $R_F \subseteq R$ of the roots we are interested in are those whose tree contain a vertex from the target set T. We have to decide for each $r \in R$, whether it is in R_F.

We claim this can be done in time $O(f \log f)$. To do so, we first partition R depending on the relative position of its elements in the tree $\widetilde{T}_s$ without failures. Consider a different root $r' \in R \backslash \{r\}$. We say r' is an *immediate descendant* of r if r' is a descendant of r in $\widetilde{T}_s$ and there is no other element of R on the r-r'-path. Let $R(r)$ be the (possibly empty) subset of immediate descendants of r. Conversely, we say r is the *immediate ancestor* of all the $r' \in R(r)$. The source s is the only root that has no immediate ancestor and the sets $\{R(r)\}_{r \in R}$ partition $R \backslash \{s\}$, see Figure 3.

We can compute this partition using the traversal numbers again, but this time with a somewhat simpler approach since we do not need to distinguish between parent and child nodes. Let $pre(r)$ denote the pre-order number of r and $post(r)$ its post-order number. Root r is a (proper but not necessarily immediate) ancestor of r' if and only if $pre(r) < pre(r')$ and $post(r) > post(r')$. Out of all proper ancestors of r', the *immediate* ancestor r is the one that has minimum post-order number. We again use two sorted lists, both contain all roots from R. The first list $pre^{\downarrow}$ is sorted by decreasing pre-order number, and the second one $post^{\uparrow}$ is sorted by *increasing* post-order number. Let r' be the first entry of $pre^{\downarrow}$. Hence, the entries of $post^{\uparrow}$ that come after r' are exactly its proper ancestors. By the above argument, the immediate ancestor r we are looking for is the successor of r' in $post^{\uparrow}$. If we now remove r' from both lists, we ensure that again all remaining roots have smaller pre-order number than the (new) first entry of $pre^{\downarrow}$. We iterate this until both lists only contain the source s (which is without any proper ancestors), and in each step assign the current first entry

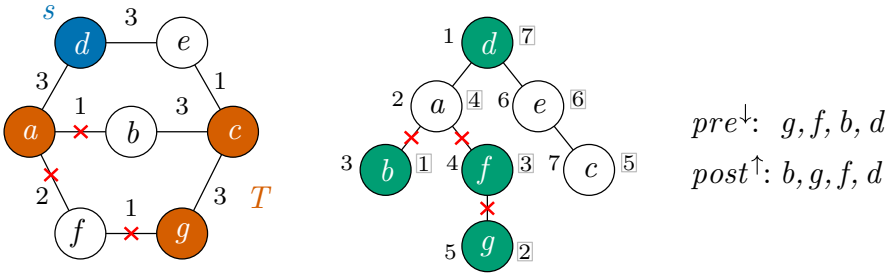

Fig. 3 Illustration of the proof of Theorem 3. On the left is the example graph, now with a single source d, target set $T = \{a, c, g\}$, and failing edges $\{a, b\}$, $\{a, f\}$, and $\{f, g\}$. In the center is the shortest-path tree $\widetilde{T}_d$ rooted in d. Each node is annotated with its pre-order number (left) and post-order number (right, boxed). The root set $R = \{b, d, f, g\}$ is highlighted (in green). The subsets of immediate descendants are $R(d) = \{b, f\}$ and $R(f) = \{g\}$, the sets $R(b)$ and $R(g)$ are empty. On the right are the two sorted copies of R

of $pre^{\downarrow}$ to its immediate ancestor. This way, any $r \in R$ gets assigned all the roots in $R(r)$. Sorting the two lists takes time $O(|R| \log |R|) = O(f \log f)$.

Finally, we can compute the subset $R_F \subseteq R$. Observe that a vertex $r \in R$ lies in R_F if and only if there is at least one vertex of T that falls into the subtree of $\widetilde{T}_s$ rooted in r but not in any of the subtrees rooted in (proper) descendants of r in R. We decide this via the values $count(r)$, that is the number of targets from T in the subtree of $\widetilde{T}_s$ rooted in r. We only need to consider the *immediate* descendants in $R(r)$. If the element of T is in some lower subtree, then it is also accounted for by a lower root. Putting this all together shows that some $r \in R$ is in R_F if and only if $count(r) > \sum_{r' \in R(r)} count(r')$. Since the sets $\{R(r)\}_r$ are a partition, those checks can be performed for all $r \in R$ in total time $O(f)$. □

We now generalize the result to multiple sources. That means, we build an f-FDO-ST for a general set S while still only having access to a *single-source* DSO. The naive solution would be to iterate the above construction for the sT-case for every source $s \in S$. We indeed reduce the case of diameter estimation from multiple sources to that from a single source, but do it more efficiently. As it turns out, it is enough to construct the sT-diameter oracle for only *two* arbitrary vertices $s \in S$ and $t \in T$.

Lemma 8 *Let $G = (V, E)$ be an undirected positively edge-weighted graph. Let $S, T \subseteq V$ be non-empty sets of vertices, and $s \in S$ and $t \in T$ be two vertices. Assume access to an f-FDO-sT and an f-FDO-tS for G with respective stretches σ_{sT} and σ_{tS}, preprocessing times $\mathtt{P}_{sT}$ and $\mathtt{P}_{tS}$, space requirements $\mathtt{S}_{sT}$ and $\mathtt{S}_{tS}$, and query times $\mathtt{Q}_{sT}$ and $\mathtt{Q}_{tS}$. There exists an f-FDO-ST for G with stretch $\sigma_{sT} + \sigma_{tS} + \min(\sigma_{sT}, \sigma_{tS})$, preprocessing time $\mathtt{P}_{sT} + \mathtt{P}_{tS}$, space $\mathtt{S}_{sT} + \mathtt{S}_{tS}$, and query time $\mathtt{Q}_{sT} + \mathtt{Q}_{tS} + O(1)$.*

Proof Consider a failure set $F \subseteq E$. Let $\mathcal{D}_{sT}(F)$ be the σ_{sT}-approximation of $\mathrm{diam}(G - F, s, T)$, and $\mathcal{D}_{tS}(F)$ the σ_{tS}-approximation of $\mathrm{diam}(G - F, t, S)$ obtained from the respective oracles. To answer the query F, the f-FDO-ST outputs

$$\widehat{D} = \mathcal{D}_{sT}(F) + \mathcal{D}_{tS}(F) + \min\Big(\mathcal{D}_{sT}(F), \mathcal{D}_{tS}(F)\Big).$$

Consider an arbitrary pair of vertices $(s_0, t_0) \in S \times T$. We get

$$\begin{aligned} d_{G-F}(s_0, t_0) &\leqslant d_{G-F}(s_0, t) + d_{G-F}(t, s) + d_{G-F}(s, t_0) \\ &\leqslant \mathcal{D}_{tS}(F) + \min\Big(\mathcal{D}_{sT}(F), \mathcal{D}_{tS}(F)\Big) + \mathcal{D}_{sT}(F) = \widehat{D}. \end{aligned}$$

Here, we use that the graph is undirected and therefore $d_{G-F}(s_0, t) \leqslant \mathcal{D}_{tS}(F)$ and $d_{G-F}(t, s) \leqslant \mathcal{D}_{sT}(F)$ both hold. So $\widehat{D}$ is never smaller than any distance between vertices from S and T, it never underestimates $\operatorname{diam}(G-F, S, T)$.

For the other part of the approximation, we insert the stretches of the two estimates $\mathcal{D}_{sT}(F)$ and $\mathcal{D}_{tS}(F)$.

$$\begin{aligned} \widehat{D} \leqslant \sigma_{sT} \cdot \operatorname{diam}(G-F, s, T) \; + \sigma_{tS} \cdot \operatorname{diam}(G-F, t, S) \\ \min\Big(\sigma_{sT} \cdot \operatorname{diam}(G-F, t, S),\; \sigma_{tS} \cdot \operatorname{diam}(G-F, s, T)\Big). \end{aligned}$$

Since both $\operatorname{diam}(G-F, s, T)$ and $\operatorname{diam}(G-F, t, S)$ are at most $\operatorname{diam}(G-F, S, T)$, this shows that $\widehat{D}$ estimates the latter with a stretch of $\sigma_{sT} + \sigma_{tS} + \min(\sigma_{sT}, \sigma_{tS})$. □

Combining Theorem 3 and Lemma 8 gives a reduction from the construction of f-FDO-ST to that of single-source f-DSOs. However, it results in a data structure with a stretch of $3 + 6\sigma$, where σ is the original stretch of the single-source f-DSO. We improve this by not treating Lemma 8 as a black box.

Theorem 2 *Let $G = (V, E)$ be an undirected positively edge-weighted graph with n vertices and m edges. Let $S, T \subseteq V$ be two non-empty sets. Assume access to a single-source f-DSO for G with stretch σ, space $\mathtt{S}$, query time $\mathtt{Q}$, and preprocessing time $\mathtt{P}$. There exists an f-FDO-ST for G with stretch $2 + 5\sigma$, space $O(\mathtt{S} + n)$, query time $O(f\,(\log(f) + \mathtt{Q}))$, and preprocessing time $\widetilde{O}(\mathtt{P} + m)$.*

Proof Let vertices $s \in S$ and $t \in T$ be arbitrary. The preprocessing algorithm of the f-FDO-ST uses the single-source f-DSO twice, once for the source s and once for the target t, to construct an f-FDO-sT $\mathcal{D}_{sT}$ and an f-FDO-tS $\mathcal{D}_{tS}$ both with stretch $1 + 2\sigma$. The details are given in Theorem 3.

For a set $F \subseteq E$ of at most f edge failures, let $\mathcal{D}_{sT}(F)$ and $\mathcal{D}_{tS}(F)$ be the respective $(1 + 2\sigma)$-approximations of $\operatorname{diam}(G - F, s, T)$ and $\operatorname{diam}(G - F, t, S)$. Further, let $\mathcal{D}_{st}(F)$ be a σ-approximation of $d_{G-F}(s, t)$, obtained from the DSO with the single source s. The query algorithm outputs the estimate

$$\widehat{D} = \mathcal{D}_{tS}(F) + \mathcal{D}_{st}(F) + \mathcal{D}_{sT}(F).$$

Let $(s_0, t_0) \in S \times T$. By the same arguments as in the proof of Lemma 8, we get

$$\begin{aligned} d_{G-F}(s_0, t_0) &\leqslant d_{G-F}(s_0, t) + d_{G-F}(t, s) + d_{G-F}(s, t_0) \\ &\leqslant \mathcal{D}_{tS}(F) + \mathcal{D}_{st}(F) + \mathcal{D}_{sT}(F) \\ &\leqslant (1{+}2\sigma) \cdot \operatorname{diam}(G{-}F, t, S) + \sigma \cdot d_{G-F}(t, s) + (1{+}2\sigma) \cdot \operatorname{diam}(G{-}F, t, S). \end{aligned}$$

Again, $\mathrm{diam}(G-F, t, S)$, $d_{G-F}(t, s)$, and $\mathrm{diam}(G-F, t, S)$ are all bounded by $\mathrm{diam}(G-F, S, T)$. The sum of the stretch coefficients is $2 \cdot (1+2\sigma) + \sigma = 2 + 5\sigma$. □

5 Unrestricted Diameter Oracles

In this section, we discuss how to construct oracles that estimate the ordinary diameter, the maximum distance between any two vertices, i.e., $S = T = V$. As before, we first use all-pairs distance sensitivity oracles to build our data structures and afterwards describe the single-source case as an alternative.

5.1 Diameter Oracles from All-Pairs DSOs

There is a simple reduction from the construction of f-FDOs to that of f-DSOs that results in a stretch of $1 + \sigma$, where σ is the stretch of the distance sensitivity oracle. The reduction can also handle directed graphs.

Theorem 4 *Let G be an (undirected or directed) positively edge-weighted graph with n vertices and m edges. Assume access to an f-DSO for G with stretch σ, space $\mathtt{S}$, query time $\mathtt{Q}$, and preprocessing time $\mathtt{P}$. There exists an f-FDO for G with stretch $1 + \sigma$, space $\mathtt{S} + O(1)$, query time $O(f^2\mathtt{Q})$, and preprocessing time $\mathtt{P} + \widetilde{O}(mn)$.*

Proof Let $\mathcal{D}$ denote the assumed f-DSO and, for any two vertices $x, y \in V(G)$ and set $F \subseteq E(G)$ of at most f failing edges, let $\mathcal{D}(u, v, F)$ be the reported σ-approximation of the replacement distance $d_{G-F}(u, v)$. We precompute the original diameter $\mathrm{diam}(G)$ by running Dijkstra's algorithm from every vertex in time $O(mn + n^2 \log n)$. Our f-FDO stores $\mathcal{D}$ and the value $\mathrm{diam}(G)$. When queried with a set F, it returns

$$\widehat{D} = \mathrm{diam}(G) + \max_{u,v \in V(F)} \mathcal{D}(u, v, F).$$

The query time is $|F|^2\mathtt{Q} = O(f^2\mathtt{Q})$.

To prove the stretch of the diameter oracle, we first show that $\widehat{D}$ is always at least the fault-tolerant diameter, namely, we claim $d_{G-F}(s, t) \leqslant \widehat{D}$ for all $s, t \in V(G)$. Let π be a shortest path from s to t in G. We only need to prove the lower bound when π contains some failing edges in F as otherwise $d_{G-F}(s, t) = d_G(s, t) \leqslant \mathrm{diam}(G) \leqslant \widehat{D}$. Let x_s (resp., x_t) be the vertex of $V(F)$ that is closest to s (resp., t) in π. By choice of x_s and x_t we have that $d_{G-F}(s, x_t) = d_G(s, x_s)$ and $d_{G-F}(x_t, t) = d_G(x_t, t)$ since no failure occurs on the subpaths $\pi[s, x_s]$ and $\pi[x_t, t]$. Moreover, $d_G(s, x_s) + d_G(x_t, t) \leqslant d_G(s, t) \leqslant \mathrm{diam}(G)$. The triangle inequality now gives

$$\begin{aligned} d_{G-F}(s, t) &\leqslant d_{G-F}(s, x_s) + d_{G-F}(x_s, x_t) + d_{G-F}(x_t, t) \\ &\leqslant d_G(s, x_s) + \max_{u,v \in V(F)} \mathcal{D}(u, v, F) + d_G(x_t, t) \\ &\leqslant \mathrm{diam}(G) + \max_{u,v \in V(F)} \mathcal{D}(u, v, F) = \widehat{D}. \end{aligned}$$

For the upper bound on the stretch, recall that the DSO $\mathcal{D}$ has stretch σ.

$$\begin{aligned}\widehat{D} = \operatorname{diam}(G) + \max_{u,v\in V(F)} \mathcal{D}(u,v,F) &\leqslant \operatorname{diam}(G-F) + \sigma \cdot \max_{u,v\in V(F)} d_{G-F}(u,v)\\ &\leqslant \operatorname{diam}(G-F) + \sigma \cdot \operatorname{diam}(G-F) = (1+\sigma)\cdot \operatorname{diam}(G-F).\end{aligned}$$

□

5.2 Diameter Oracles from Single-Source DSOs

Recall that single-source distance sensitivity oracles only report the replacement distances from one vertex s specified at preprocessing time. We now use such data structures to built f-FDOs. The following theorem is based on the intuition that in an undirected graph twice the eccentricity of an arbitrary vertex is a 2-approximation for the diameter. We transfer this intuition to directed graphs under edge failures.

Theorem 5 *Let G be an (undirected or directed) positively edge-weighted graph with n vertices and m edges. Assume access to a single-source f-DSO for G with stretch σ, space $\mathtt{S}$, query time $\mathtt{Q}$, and preprocessing time $\mathtt{P}$. There exists an f-FDO for G with stretch $2+2\sigma$, space $O(\mathtt{S})$, query time $O(f\,\mathtt{Q})$, and preprocessing time $\widetilde{O}(\mathtt{P}+m)$.*

Proof Suppose the graph $G = (V, E)$ is directed and let $\overleftarrow{G} = (V, \overleftarrow{E})$ denote the directed graph in which the orientation of every edge is inverted compared to G. Let $s \in V$ be any vertex. We precompute two single-source distance sensitivity oracles $\mathcal{D}$ and $\overleftarrow{\mathcal{D}}$ for the respective graphs G and $\overleftarrow{G}$, both with source s. In other words, for any $v \in V$ and any set F of up to f edge failures, $\mathcal{D}(v, F)$ is a σ-approximation of $d_{G-F}(s, v)$; accordingly, $\overleftarrow{\mathcal{D}}(v, F)$ approximates $d_{G-F}(v, s)$. (Of course, if G is undirected, one single-source f-DSO suffices.) Also, we compute and store the respective eccentricities of s in G and $\overleftarrow{G}$, that is, $\operatorname{diam}(G, s, V)$ and $\operatorname{diam}(G, V, s)$.

Let F be a set of up to f edge failures in G and $V(F)$ the set of its end points. Our oracles returns the value

$$\widehat{D} = \operatorname{diam}(G, s, V) + \operatorname{diam}(G, V, s) + \max_{v\in V(F)} \mathcal{D}(v, F) + \max_{v\in V(F)} \overleftarrow{\mathcal{D}}(v, F).$$

As the query time of both $\mathcal{D}$ and $\overleftarrow{\mathcal{D}}$ is $\mathtt{Q}$, the query time of our oracle is $O(f\,\mathtt{Q})$.

Consider any two vertices $u, w \in V$. As before, we first show $d_{G-F}(u, w) \leqslant \widehat{D}$. Recall that $\pi_{u,w}$ is the shortest path from u to w. If $E(\pi_{u,w}) \cap F = \emptyset$, then

$$d_{G-F}(u, w) = d_G(u, w) \leqslant d_G(u, s) + d_G(s, w) \leqslant \operatorname{diam}(G, V, s) + \operatorname{diam}(G, s, V) \leqslant \hat{D}.$$

Otherwise, let x (resp., y) be the first (resp., last) vertex from $V(F)$ on $\pi_{u,w}$. Then the prefix between u and x as well as the suffix between y and v are non-faulty edge-disjoint subpaths of $\pi_{u,w}$, thus $d_{G-F}(u, x) + d_{G-F}(y, w) \leqslant d_G(u, w)$. Using this,

we now obtain

$$\begin{aligned} d_{G-F}(u, w) &\leqslant d_{G-F}(u, x) + d_{G-F}(x, y) + d_{G-F}(y, w) \\ &\leqslant d_G(u, w) + d_{G-F}(x, y). \end{aligned}$$

We bound each of the last two terms as follows.

$$d_G(u, w) \leqslant d_G(u, s) + d_G(s, w) \leqslant \operatorname{diam}(G, V, s) + \operatorname{diam}(G, s, V),$$

$$d_{G-F}(x, y) \leqslant d_{G-F}(x, s) + d_{G-F}(y, s) \leqslant \max_{v \in V(F)} \mathcal{D}(v, F) + \max_{v \in V(F)} \overleftarrow{\mathcal{D}}(v, F).$$

So, summing these inequalities, we get $d_{G-F}(u, w) \leqslant \hat{D}$ as required. □

6 Space Lower Bound

We now discuss the limits of compression for diameter oracles. Our lower bound shows that, for sufficiently small stretch, any such data structure must take a super-linear amount of space. We prove this using an argument from information theory, which results in the bound holding unconditionally. In particular, it remains true even if we allow for an unbounded query time.

Theorem 6 *Any f-FDO or f-FDO-ST for n-vertex graphs with sensitivity $f \geqslant 2$ and a stretch of $\frac{5}{3} - \varepsilon$ for any $\varepsilon = \varepsilon(n) > 0$ requires $\Omega(n^{3/2})$ bits of space.*

The stretch requirement stems from the fact that any such f-FDO-ST can distinguish whether the graph $G-F$ has fault-tolerant diameter 3 or 5. The core of the proof of Theorem 6 is to establishing the fact that *any* data structure with this ability requires $\Omega(n^{3/2})$ bits of space.

Lemma 9 *For infinitely many positive integers n, there exists a graph $G = (V, E)$ with n vertices (and two sets $S, T \subseteq V$) such that any data structure that decides for all pairs of edges $e, e' \in E$, whether $G - \{e, e'\}$ has diameter (resp., ST-diameter) at most 3 or at least 5 requires $\Omega(n^{3/2})$ bits of space.*

6.1 The graphs *H* and *G*

For the remainder of this section, let $n = 6N$ for some integer $N \geqslant 4$ that is a perfect square. We first construct an auxiliary undirected graph H with n vertices. Let indices i, j range over the set $[\sqrt{N}] = \{1, 2, \ldots, \sqrt{N}\}$ and let k range over $\{0, 1\}$. Instead of the usual index notation like $b_{i,j,k}$, we use $b[i, j, k]$ to enhance readability. We define four pairwise disjoint sets of vertices

Table 8 The edges of the auxiliary graph H in the proof of Lemma 9. The left column lists certain subsets of $V(H) \times V(H)$, the middle column shows the respective pairs of vertices with their indices, and the right column gives the condition under which those vertices indeed form an edge. The symbol $\oplus$ denotes the exclusive disjunction. The edges are intended to be undirected

Set Pair	Vertex Pair	Edge Condition
$A \times A$		no edges
$B \times B$	$b[i, j, k],\ b[x, y, z]$	$(i = x) \oplus (j = y)$
$C \times C$	$c[i, j, k],\ c[x, y, z]$	$(i = x) \oplus (j = y)$
$D \times D$		no edges
$A \times B$	$a[i, j],\ b[x, y, z]$	$(i = x) \wedge (z = 0)$
$B \times C$	$b[i, j, k],\ c[x, y, z]$	$(i = x) \wedge (j = y)$
$C \times D$	$c[i, j, k],\ d[x, y]$	$(j = y) \wedge (k = 0)$

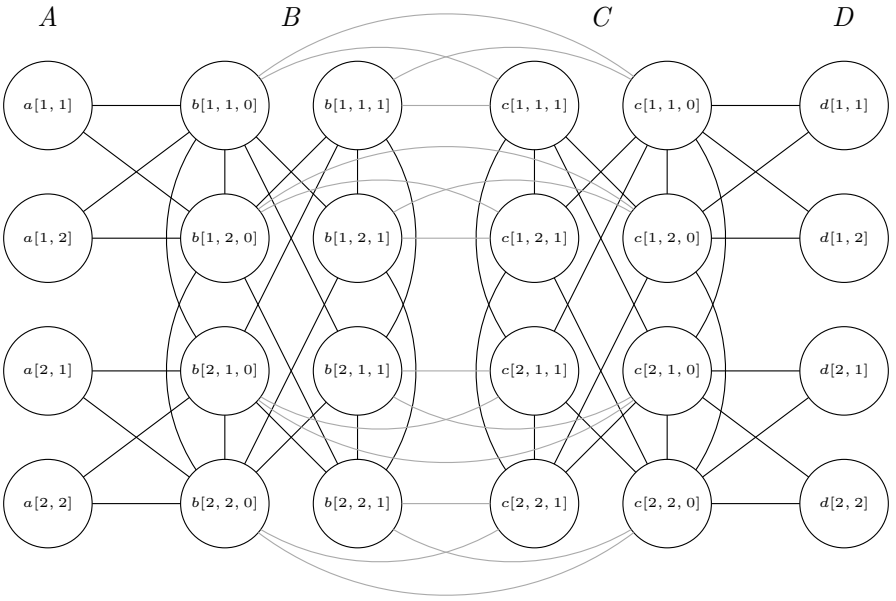

Fig. 4 The graph H for parameter $N = 4$. The edges between the vertex sets B and C are drawn in gray to avoid clutter

- $A = \{a[i, j]\}_{i,j \in [\sqrt{N}]}$
- $B = \{b[i, j, k]\}_{i,j \in [\sqrt{N}], k \in \{0,1\}}$
- $C = \{c[i, j, k]\}_{i,j \in [\sqrt{N}], k \in \{0,1\}}$
- $D = \{d[i, j]\}_{i,j \in [\sqrt{N}]}$

The sets A, B, C, D have respective cardinalities $N, 2N, 2N$, and N. The vertex set of H is $V(H) = A \cup B \cup C \cup D$. The edges in H are shown in Table 8. They are defined using relations between the indices of the participating vertices. For example, the second line states that an edge $\{b[i, j, k],\ b[x, y, z]\}$ between elements of B exists if and only if *either* $i = x$ holds *or* $j = y$. The indices $k, z \in \{0, 1\}$ can be arbitrary. Figure 4 illustrates the symmetries of H. Next, we establish some features of the auxiliary graph.

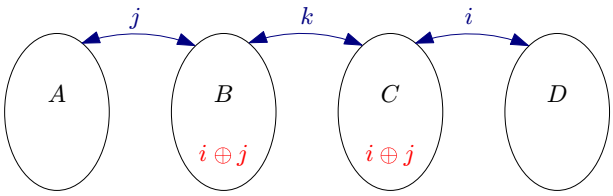

Fig. 5 Schematic representation of the transitions in graph H. The label i refers to the first index, j to the second one, and k to the third index (used only in B and C). The transition from A to B is labeled with a blue j to illustrate that an edge $\{a[i, j], b[x, y, 0]\}$ is allowed to jump from any j to any y, while $i = x$ must remain the same; likewise, for the transition from C to D which may vary index i. Going from B to C must leave both i and j in place, but may change k. The red labels mark that when moving within the sets B or C either the first or the second index change

Lemma 10 *The graph H has $\Theta(n^{3/2})$ edges.*[10]

Proof We claim that the maximum degree in H is $5\sqrt{N} - 2$, assumed by the vertices $b[i, j, 0]$ and $c[i, j, 0]$ for any $i, j \in [\sqrt{N}]$. These $2N$ vertices are a constant fraction of $V(H)$. The claim thus implies that there are $\Theta(N^{3/2}) = \Theta(n^{3/2})$ edges.

To prove the claim, we fix indices i^*, j^* and calculate the degree of the vertex $v = b[i^*, j^*, 0]$. It has no neighbors in D and $\sqrt{N}$ neighbors in A, namely, $a[i^*, 1], \ldots, a[i^*, \sqrt{N}]$. The neighbors in C are $c[i^*, j^*, 0]$ and $c[i^*, j^*, 1]$. To count v's neighbors in B, first consider the subset that shares its first component, $B[i^*] = \{b[i^*, j, k] \mid j \in [\sqrt{N}], k \in \{0, 1\}\}$. The vertex v is connected to all of those except $b[i^*, j^*, 1]$ and v itself, which makes for $2\sqrt{N} - 2$ neighbors. Finally, among the vertices in $B \setminus B[i^*]$ with a different first component, v is connected to the set $\{b[1, j^*, k], \ldots, b[i^*-1, j^*, k], b[i^*+1, j^*, k], \ldots, b[\sqrt{N}, j^*, k]\}_{k \in \{0,1\}}$, contributing the remaining $2(\sqrt{N}-1)$ neighbors.

We briefly argue that this is indeed the maximum degree. The edges incident to $c[i^*, j^*, 0]$ are symmetrical to those of $b[i^*, j^*, 0]$. The vertex $b[i^*, j^*, 1]$ (resp., $c[i^*, j^*, 1]$) is missing the $\sqrt{N}$ edges into A (into D). Also, the vertices in A and D have only degree $\sqrt{N}$ to begin with. □

Lemma 11 *The diameter of H is* 3.

Proof We give explicit shortest paths of length at most 3 between all possible pairs of vertices. The edges in H impose rules how the indices of a vertex change when transitioning between the sets A, B, C, and D. Figure 5 serves as a reference. Recall that the edges are undirected, thus all paths below work both ways. For some index $i \in [\sqrt{N}]$, we use the symbol $\bar{i}$ for an arbitrary *other* index from $[\sqrt{N}] \setminus \{i\}$. Since $N \geqslant 4$, we have $\sqrt{N} \geqslant 2$ and such an alternative index exists.

- Consider (distinct) vertices $a[i, j], a[x, y] \in A$. They are joined by any path of the form $(a[i, j],\ b[i, y, 0],\ b[x, y, 0],\ a[x, y])$. If the respective first indices

[10] If one is interested in the leading constants, the proof of Lemma 10 shows that there are $2N$ vertices of degree $5\sqrt{N}-2$, $2N$ more of degree $4\sqrt{N}-2$, and the remaining $2N$ vertices have degree $\sqrt{N}$. The handshaking lemma implies that there are $10N^{3/2} - 4N = \frac{5}{3\sqrt{6}} n^{3/2} - \frac{2}{3} n$ edges.

$i \neq x$ are different, the path has length 3; otherwise, the inner vertices $b[i, y, 0]$ and $b[x, y, 0]$ are the same and the path shortens to length 2.

- The case $d[i, j], d[x, y] \in D$, is symmetric to that of two vertices from A.
- Two vertices from B, or from C respectively, have distance at most 2. Let $b[i, j, k], b[x, y, z] \in B$ be two different vertices. If exactly one out of the equalities $i = x$ and $j = y$ hold, they have a direct edge between them. If we have both $i = x$ and $j = y$, it must be that the vertices differ exactly the third component $k \neq z$. Let $\bar{i}$ be any index different from i. The two vertices are joined by the path $(b[i, j, k],\ b[\bar{i}, j, k],\ b[i, j, z])$. Finally, if $i \neq x$ and $j \neq y$. the path is $(b[i, j, k],\ b[x, j, k],\ b[x, y, z])$. The argument for vertices $c[i, j, k], c[x, y, z] \in C$ is the same.
- For vertex pairs $(a[i, j], b[x, y, z]) \in A \times B$, the key observation is that any edge inside of B changes exactly one of the first two indices. If $i \neq x$, the path is $(a[i, j],\ b[i, y, 0],\ b[x, y, z])$. Otherwise, we have $i = x$. Hence, the target vertex is $b[x, y, z] = b[i, y, z]$ and the respective path is $(a[i, j],\ b[i, \overline{y}, 0],\ b[i, y, z])$. In summary, the distance from A to B is at most 2. The pairs $(d[i, j], c[x, y, z]) \in D \times C$ are handled symmetrically.
- The vertex pairs $(a[i, j], c[x, y, z]) \in A \times C$ have shortest paths of the form $(a[i, j],\ b[i, y, 0],\ c[i, y, z],\ c[x, y, z])$. Note that if $i = x$ the last two vertices are the same. Same holds for paths from D to B.
- Pairs $(a[i, j], d[x, y]) \in A \times D$: paths $(a[i, j],\ b[i, y, 0],\ c[i, y, 0],\ d[x, y])$.
- Pairs $(b[i, j, k], c[x, y, z]) \in B \times C$: $(b[i, j, k], b[x, j, k], c[x, j, k], c[x, y, z])$, possibly shortened if consecutive vertices are the same.

□

The construction of graph G

In the next step of the proof of Lemma 9, we show how to encode the entries of a large class[11] of $\sqrt{N} \times \sqrt{N} \times \sqrt{N}$ binary matrix (tensors) in the fault-tolerant diameter of some supergraph $G \supseteq H$. In other words, let $M \in \{0, 1\}^{\sqrt{N} \times \sqrt{N} \times \sqrt{N}}$ be a matrix and ℓ be another index ranging over $[\sqrt{N}]$. We construct a graph G that, for each triple (i, j, ℓ), has a set F of two edges such that one can infer the entry $M[i, j, \ell]$ from $\mathrm{diam}(G - F)$.

The graph G has the same vertex set $V = V(H)$ as H. It also inherits all edges of H, but additionally contains the following edges that depend on M.

- For all $i, j, \ell \in [\sqrt{N}]$ with $M[i, j, \ell] = 1$, add the edges $\{a[i, j], b[i, \ell, 1]\}$ and $\{c[i, \ell, 1], d[j, \ell]\}$ to G.

Note that $O(N^{3/2})$ edges are added and the diameter of G is also at most 3.

[11] The relevant subclass of matrices will turn out to be all $M \in \{0, 1\}^{\sqrt{N} \times \sqrt{N} \times \sqrt{N}}$ that are constraint to have entry $M[i, j, \ell] = 0$ whenever $i = j$ or $j = \ell$.

6.2 Fault-Tolerant Diameter Revealing Matrix Entries

Fix four indices $i, j, x, y \in [\sqrt{N}]$ such that $i \neq x$ and $j \neq y$. We define two sets F, E^*, each one containing exactly two pairs of vertices from V. The first one F in fact contains two edges that are present in H. Let $e_1 = \{a[i, j], b[i, y, 0]\}$ and $e_2 = \{c[i, y, 0], d[x, y]\}$ and $F = \{e_1, e_2\}$. The elements of the second set E^* are only edges of G if the entries $M[i, j, y]$ and $M[i, x, y]$ are both 1. In more detail, let E^* be the set comprising the two pairs $e_1^* = \{a[i, j], b[i, y, 1]\}$ and $e_2^* = \{c[i, y, 1], d[x, y]\}$.

The next two lemmas are crucial in showing how the fault-tolerant diameter of G depends on $M[i, j, y]$ and $M[i, x, y]$. We would like to highlight the fact that both of them hold regardless of the other entries in M.

Lemma 12 *For any four indices $i, j, x, y \in [\sqrt{N}]$ such that $i \neq x$ and $j \neq y$, if the edges in E^* are absent from G, then the diameter of $G - F$ is at least 5.*

Proof Suppose none of the edges in $E^* = \{\{a[i, j], b[i, y, 1]\}, \{c[i, y, 1], d[x, y]\}\}$ are present in G. We show that the distance between the vertices $a[i, j]$ and $d[x, y]$ in $G - F$ is at least 5. Observe that, when moving from one vertex to the other, both the first and the second index must change. The high-level idea of this proof is that we need sufficiently many edges on any path in order to accompany those index changes.

To reach a contradiction, assume that there exists three (not necessarily distinct) vertices $u, v, w \in V$ such that $P = (a[i, j], u, v, w, d[x, y])$ is a path of length at most 4. P must pass across sets $A \to B$, $B \to C$, and $C \to D$. Recall that the edge $\{a[i, j], b[i, y, 0]\} \in F$ failed in $G - F$. Since E^* is absent, the entries $M[i, j, y]$ and $M[i, x, y]$ are both 0. However, other 1-entries of M may have resulted in additional neighbors of $a[i, j]$. In summary, the neighborhood $N_{G-F}(a[i, j])$ is

$$\Big\{b[i, \overline{y}, 0] \mid \overline{y} \in [\sqrt{N}]\backslash\{y\}\Big\} \cup \Big\{b[i, \overline{y}, 1] \mid \overline{y} \in [\sqrt{N}]\backslash\{y\} \wedge M[i, j, \overline{y}] = 1\Big\}.$$

Therefore, the first edge $\{a[i, j], u\}$ of P cannot change the index i and we have $u = b[i, \overline{y}, k]$ where $\overline{y}$ is *different* from y, and k may be 0 or 1 depending on M.

Now note that also the edge $\{c[i, y, 0], d[x, y]\} \in F$ failed. Applying the same reasoning to the last edge $\{w, d[x, y]\}$ of the path P, shows that w must lie in C and have the form $w = c[\overline{i}, y, z]$, where again $z \in \{0, 1\}$ depending on the entries of M.

At least one of the edges $\{u, v\}$ or $\{v, w\}$ passes from B to C. Let us first assume that it is $\{u, v\}$. This edge must have already been present in H and cannot change *any* of the first two indices, that is, $v = c[i, \overline{y}, \xi]$ for some $\xi \in \{0, 1\}$. This implies that the edge $\{v, w\} = \{c[i, \overline{y}, \xi], c[\overline{i}, y, z]\}$ connects two vertices in C and must change the first two indices simultaneously. However, by definition all edges within C change exactly one out of the first two.

If instead the edge $\{v, w\}$ passes from B to C, then a symmetric argument shows that $\{u, v\}$ must change the first two indices while staying in B, which again is impossible. This also covers the case where P has even fewer than 4 edges as still one of those must eventually pass from B to C and the other "edge" degenerates to a single vertex. □

Lemma 13 *If the edges in E^* are present, then the diameter of $G - F$ is 3.*

Proof The proof is essentially the same as that of Lemma 11. The only difference is that whenever the edge $e_1 = \{a[i, j], b[i, y, 0]\} \in F$ would be used, the respective path instead traverses $e_1^* = \{a[i, j], b[i, y, 1]\} \in E^*$. Likewise, every occurrence of $e_2 = \{c[i, y, 0], d[x, y]\}$ is replaced by $e_2^* = \{c[i, y, 1], d[x, y]\}$. □

We now finish the proofs[12] of Lemma 9 and, in turn, Theorem 6. Suppose there exists a data structure for the graph G with sensitivity $f \geqslant 2$ that distinguishes whether the fault-tolerant diameter is bounded by 3 or at least 5. We can use it to infer the entry $M[i, j, \ell]$ for any triple $(i, j, \ell) \in [\sqrt{N}]^3$ of indices such that i and j differ from each other, and j and ℓ differ. In the notation used at the start of Section 6.2, we set $x = j$ and $y = \ell$. The resulting failure set is therefore

$$\begin{aligned} F &= \Big\{\{a[i, j], b[i, y, 0]\}, \{c[i, y, 0], d[x, y]\}\Big\} \\ &= \Big\{\{a[i, j], b[i, \ell, 0]\}, \{c[i, \ell, 0], d[j, \ell]\}\Big\} \end{aligned}$$

Our assumptions $i \neq j$ and $j \neq \ell$ ensure that $i \neq x$ and $j \neq y$, the Lemma 12 and 13 thus apply. If $M[i, j, \ell] = 1$, then $M[i, j, y] = M[i, x, y] = 1$ since they are all the same entry, and the respective edge set E^* is present in G. Lemma 13 implies that $\operatorname{diam}(G-F) = 3$. If conversely $M[i, j, \ell] = 0$, then $M[i, j, y] = M[i, x, y] = 0$, E^* is absent, and $\operatorname{diam}(G-F) \geqslant 5$. For the assertion of Lemma 9 about the ST-diameter (instead of the unrestricted diameter), we just choose $S = A$ and $T = D$.

There are $2^{\sqrt{N}(\sqrt{N}-1)^2} = 2^{\Omega(N^{3/2})} = 2^{\Omega(n^{3/2})}$ possibilities of fixing the entries in M whose indices $(i, j, \ell) \in [\sqrt{N}]^3$ satisfy $i \neq j$ and $j \neq \ell$. The data structure in question must therefore take $\Omega(n^{3/2})$ bits of space. The proof of Theorem 6 is completed by observing that any f-FDO-ST with stretch better than $5/3$ is indeed capable of distinguishing those two cases.

Acknowledgements The authors thank the anonymous Algorithmica reviewers for their many helpful comments, and for sharing constructions that improved the space overhead, query time, and preprocessing time of Theorem 1.

Author Contributions All authors contributed equally to the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

[12] Part of the proof of Lemma 9 is to reconcile the i, j, x, y index naming scheme from the start of Section 6.2 with the i, j, ℓ scheme used for M. The former is chosen to align the proofs of Lemmas 12 and 13 with the definition of graph H. The latter emphasizes the three degrees of freedom when addressing the entries of M.

References

1. Bilò, D., Choudhary, K., Cohen, S., Friedrich, T., Krogmann, S., Schirneck, M.: Fault-Tolerant ST-Diameter Oracles. In: Proceedings of the 50th International Colloquium on Automata, Languages, and Programming (ICALP), pp. 24–12420 (2023). https://doi.org/10.4230/LIPIcs.ICALP.2023.24
2. Aingworth, D., Chekuri, C., Indyk, P., Motwani, R.: Fast Estimation of Diameter and Shortest Paths (Without Matrix Multiplication). SIAM J. Comput. **28**, 1167–1181 (1999). https://doi.org/10.1137/S0097539796303421
3. Ancona, B., Henzinger, M., Roditty, L., Vassilevska Williams, V., Wein, N.: Algorithms and Hardness for Diameter in Dynamic Graphs. In: Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP), pp. 13–11314 (2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.13
4. Backurs, A., Roditty, L., Segal, G., Vassilevska Williams, V., Wein, N.: Toward Tight Approximation Bounds for Graph Diameter and Eccentricities. SIAM J. Comput. **50**, 1155–1199 (2021). https://doi.org/10.1137/18M1226737
5. Chechik, S., Larkin, D.H., Roditty, L., Schoenebeck, G., Tarjan, R.E., Vassilevska Williams, V.: Better Approximation Algorithms for the Graph Diameter. In: Proceedings of the 25th Symposium on Discrete Algorithms (SODA), pp. 1041–1052 (2014). https://doi.org/10.1137/1.9781611973402.78
6. Choudhary, K., Gold, O.: Extremal Distances in Directed Graphs: Tight Spanners and Near-Optimal Approximation Algorithms. In: Proceedings of the 31st Symposium on Discrete Algorithms (SODA), pp. 495–514 (2020). https://doi.org/10.1137/1.9781611975994.30
7. Crescenzi, P., Grossi, R., Lanzi, L., Marino, A.: On Computing the Diameter of Real-World Directed (Weighted) Graphs. In: Proceedings of the 11th Symposium on Experimental Algorithms (SEA), pp. 99–110 (2012). https://doi.org/10.1007/978-3-642-30850-5_10
8. Roditty, L.: Approximating the Diameter. In: Kao, M.-Y. (ed.) Encyclopedia of Algorithms, pp. 116–117. Springer, New York City, NY, USA (2016). https://doi.org/10.1007/978-1-4939-2864-4_566
9. Roditty, L., Vassilevska Williams, V.: Fast Approximation Algorithms for the Diameter and Radius of Sparse Graphs. In: Proceedings of the 45th Symposium on Theory of Computing (STOC), pp. 515–524 (2013). https://doi.org/10.1145/2488608.2488673
10. Takes, F.W., Kosters, W.A.: Determining the Diameter of Small World Networks. In: Proceedings of the 20th Conference on Information and Knowledge Management (CIKM), pp. 1191–1196 (2011). https://doi.org/10.1145/2063576.2063748
11. Chakraborty, D., Choudhary, K.: New Extremal Bounds for Reachability and Strong-Connectivity Preservers Under Failures. In: Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP), pp. 25–12520 (2020). https://doi.org/10.4230/LIPIcs.ICALP.2020.25
12. Duan, R., Pettie, S.: Dual-Failure Distance and Connectivity Oracles. In: Proceedings of the 20th Symposium on Discrete Algorithms (SODA), pp. 506–515 (2009). http://dl.acm.org/citation.cfm?id=1496770.1496826
13. Duan, R., Pettie, S.: Connectivity Oracles for Failure Prone Graphs. In: Proceedings of the 42nd Symposium on Theory of Computing (STOC), pp. 465–474 (2010). https://doi.org/10.1145/1806689.1806754
14. Alon, N., Chechik, S., Cohen, S.: Deterministic Combinatorial Replacement Paths and Distance Sensitivity Oracles. In: Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, (ICALP), pp. 12–11214 (2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.12
15. Bilò, D., Choudhary, K., Gualà, L., Leucci, S., Parter, M., Proietti, G.: Efficient Oracles and Routing Schemes for Replacement Paths. In: Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science (STACS), pp. 13–11315 (2018). https://doi.org/10.4230/LIPIcs.STACS.2018.13
16. Grandoni, F., Vassilevska Williams, V.: Faster Replacement Paths and Distance Sensitivity Oracles. ACM Transaction on Algorithms **16**, 15–11525 (2020). https://doi.org/10.1145/3365835

17. Baswana, S., Choudhary, K., Hussain, M., Roditty, L.: Approximate Single-Source Fault Tolerant Shortest Path. ACM Trans. Algorithms **16**, 44–14422 (2020). https://doi.org/10.1145/3397532
18. Baswana, S., Kavitha, T.: Faster Algorithms for Approximate Distance Oracles and All-Pairs Small Stretch Paths. In: Proceedings of the 47th Symposium on Foundations of Computer Science (FOCS), pp. 591–602 (2006). https://doi.org/10.1109/FOCS.2006.29
19. Bilò, D., Gualà, L., Leucci, S., Proietti, G.: Multiple-Edge-Fault-Tolerant Approximate Shortest-Path Trees. Algorithmica **84**, 37–59 (2022). https://doi.org/10.1007/s00453-021-00879-8
20. Chechik, S., Langberg, M., Peleg, D., Roditty, L.: f-Sensitivity Distance Oracles and Routing Schemes. Algorithmica **63**, 861–882 (2012). https://doi.org/10.1007/s00453-011-9543-0
21. Demetrescu, C., Thorup, M., Chowdhury, R.A., Ramachandran, V.: Oracles for Distances Avoiding a Failed Node or Link. SIAM J. Comput. **37**, 1299–1318 (2008). https://doi.org/10.1137/S0097539705429847
22. Duan, R., Gu, Y., Ren, H.: Approximate Distance Oracles Subject to Multiple Vertex Failures. In: Proceedings of the 32nd Symposium on Discrete Algorithms (SODA), pp. 2497–2516 (2021). https://doi.org/10.1137/1.9781611976465.148
23. Duan, R., Ren, H.: Maintaining Exact Distances under Multiple Edge Failures. In: Proceedings of the 54th Symposium on Theory of Computing (STOC), pp. 1093–1101 (2022). https://doi.org/10.1145/3519935.3520002
24. Weimann, O., Yuster, R.: Replacement Paths and Distance Sensitivity Oracles via Fast Matrix Multiplication. ACM Trans. Algorithms **9**, 14–11413 (2013). https://doi.org/10.1145/2438645.2438646
25. Henzinger, M., Lincoln, A., Neumann, S., Vassilevska Williams, V.: Conditional Hardness for Sensitivity Problems. In: Proceedings of the 8th Conference on Innovations in Theoretical Computer Science (ITCS), pp. 26–12631 (2017). https://doi.org/10.4230/LIPIcs.ITCS.2017.26
26. Bilò, D., Cohen, S., Friedrich, T., Schirneck, M.: Space-Efficient Fault-Tolerant Diameter Oracles. In: Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS), pp. 18–11816 (2021). https://doi.org/10.4230/LIPIcs.MFCS.2021.18
27. Bilò, D., Choudhary, K., Cohen, S., Friedrich, T., Schirneck, M.: Deterministic Sensitivity Oracles for Diameter, Eccentricities and All Pairs Distances. In: Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP), pp. 22–12219 (2022). https://doi.org/10.4230/LIPIcs.ICALP.2022.22
28. Dalirrooyfard, M., Vassilevska Williams, V., Vyas, N., Wein, N.: Tight Approximation Algorithms for Bichromatic Graph Diameter and Related Problems. In: Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP), pp. 47–14715 (2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.47
29. Bernstein, A., Karger, D.R.: Improved Distance Sensitivity Oracles via Random Sampling. In: Proceedings of the 19th Symposium on Discrete Algorithms (SODA), pp. 34–43 (2008). https://dl.acm.org/citation.cfm?id=1347082.1347087
30. Bernstein, A., Karger, D.R.: A Nearly Optimal Oracle for Avoiding Failed Vertices and Edges. In: Proceedings of the 41st Symposium on Theory of Computing (STOC), pp. 101–110 (2009). https://doi.org/10.1145/1536414.1536431
31. Karczmarz, A., Sankowski, P.: Sensitivity and Dynamic Distance Oracles via Generic Matrices and Frobenius Form. In: Proceedings of the 64th Symposium on Foundations of Computer Science (FOCS), pp. 1745–1756 (2023). https://doi.org/10.1109/FOCS57990.2023.00106
32. Baswana, S., Khanna, N.: Approximate Shortest Paths Avoiding a Failed Vertex: Near Optimal Data Structures for Undirected Unweighted Graphs. Algorithmica **66**, 18–50 (2013). https://doi.org/10.1007/s00453-012-9621-y
33. Bilò, D., Chechik, S., Choudhary, K., Cohen, S., Friedrich, T., Krogmann, S., Schirneck, M.: Approximate Distance Sensitivity Oracles in Subquadratic Space. TheoretiCS (2024). https://doi.org/10.46298/theoretics.24.15
34. Chechik, S., Cohen, S., Fiat, A., Kaplan, H.: $(1+\epsilon)$-Approximate f-Sensitive Distance Oracles. In: Proceedings of the 28th Symposium on Discrete Algorithms (SODA), pp. 1479–1496 (2017). https://doi.org/10.1137/1.9781611974782.96
35. Dey, D., Gupta, M.: Nearly Optimal Fault Tolerant Distance Oracle. In: Proceedings of the 56th Symposium on Theory of Computing, (STOC), pp. 944–955 (2024). https://doi.org/10.1145/3618260.3649697

36. Brand, J.v.d., Saranurak, T.: Sensitive Distance and Reachability Oracles for Large Batch Updates. In: Proceedings of the 60th Symposium on Foundations of Computer Science (FOCS), pp. 424–435 (2019). https://doi.org/10.1109/FOCS.2019.00034
37. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching Is as Easy as Matrix Inversion. Combinatorica **7**, 105–113 (1987). https://doi.org/10.1007/BF02579206
38. Cabello, S., Chambers, E.W., Erickson, J.: Multiple-Source Shortest Paths in Embedded Graphs. SIAM J. Comput. **42**, 1542–1571 (2013). https://doi.org/10.1137/120864271
39. Charnes, A.: Optimality and Degeneracy in Linear Programming. Econometrica **20**, 160–170 (1952). https://doi.org/10.2307/1907845
40. Hartvigsen, D., Mardon, R.: The All-Pairs Min Cut Problem and the Minimum Cycle Basis Problem on Planar Graphs. SIAM J. Discret. Math. **7**, 403–418 (1994). https://doi.org/10.1137/S0895480190177042
41. Bilò, D., Cohen, S., Friedrich, T., Schirneck, M.: Near-Optimal Deterministic Single-Source Distance Sensitivity Oracles. In: Proceedings of the 29th European Symposium on Algorithms (ESA), pp. 18–11817 (2021). https://doi.org/10.4230/LIPIcs.ESA.2021.18
42. Gupta, M., Singh, A.: Generic Single Edge Fault Tolerant Exact Distance Oracle. In: Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, (ICALP), pp. 72–17215 (2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.72
43. Bilò, D., Gualà, L., Leucci, S., Proietti, G.: Compact and Fast Sensitivity Oracles for Single-Source Distances. In: Proceedings of the 24th European Symposium on Algorithms (ESA), pp. 13–11314 (2016). https://doi.org/10.4230/LIPIcs.ESA.2016.13
44. Demetrescu, C., Thorup, M.: Oracles for Distances Avoiding a Link-Failure. In: Proceedings of the 13th Symposium on Discrete Algorithms (SODA), pp. 838–843 (2002). https://dl.acm.org/citation.cfm?id=545381.545490
45. Chechik, S., Cohen, S.: Distance Sensitivity Oracles with Subcubic Preprocessing Time and Fast Query Time. In: Proccedings of the 52nd Symposium on Theory of Computing (STOC), pp. 1375–1388 (2020). https://doi.org/10.1145/3357713.3384253
46. Ren, H.: Improved Distance Sensitivity Oracles with Subcubic Preprocessing Time. J. Comput. Syst. Sci. **123**, 159–170 (2022). https://doi.org/10.1016/j.jcss.2021.08.005
47. Gu, Y., Ren, H.: Constructing a Distance Sensitivity Oracle in $O(n^{2.5794}M)$ Time. In: Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP), pp. 76–17620 (2021). https://doi.org/10.4230/LIPIcs.ICALP.2021.76
48. Zwick, U.: All Pairs Shortest Paths Using Bridging Sets and Rectangular Matrix Multiplication. J. ACM **49**, 289–317 (2002). https://doi.org/10.1145/567112.567114
49. Alman, J., Duan, R., Vassilevska Williams, V., Xu, Y., Xu, Z., Zhou, R.: More Asymmetry Yields Faster Matrix Multiplication. In: Proceedings of the 36th Symposium on Discrete Algorithms (SODA), pp. 2005–2039 (2025). https://doi.org/10.1137/1.9781611978322.63
50. Vassilevska Williams, V., Xu, Y., Xu, Z., Zhou, R.: New Bounds for Matrix Multiplication: from Alpha to Omega. In: Proceedings of the 35th Symposium on Discrete Algorithms (SODA), pp. 3792–3835 (2024). https://doi.org/10.1137/1.9781611977912.134
51. Duan, R., Wu, H., Zhou, R.: Faster Matrix Multiplication via Asymmetric Hashing. In: Proceedings of the 64th Symposium on Foundations of Computer Science (FOCS), pp. 2129–2138 (2023). https://doi.org/10.1109/FOCS57990.2023.00130
52. Bilò, D., Chechik, S., Choudhary, K., Cohen, S., Friedrich, T., Schirneck, M.: Improved Distance (Sensitivity) Oracles with Subquadratic Space. In: Proceedings of the 65th Symposium on Foundations of Computer Science (FOCS), pp. 1550–1558 (2024). https://doi.org/10.1109/FOCS61266.2024.00097
53. Tarjan, R.E., Werneck, R.F.F.: Self-adjusting top trees. SODA **5**, 813–822 (2005)
54. Thorup, M.: Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time. J. ACM **46**, 362–394 (1999). https://doi.org/10.1145/316542.316548
55. Petruschka, A.: Fault-Equivalent Lowest Common Ancestors. CoRR arXiv:abs/2411.11049 (2024). https://doi.org/10.48550/ARXIV.2411.11049. ArXiv preprint
56. Bender, M.A., Farach-Colton, M.: The LCA problem revisited. In: Proceedings of the 5th Latin American Conference in Theoretical Informatics (LATIN), pp. 88–94 (2000). https://doi.org/10.1007/10719839_9
57. Hagerup, T., Miltersen, P.B., Pagh, R.: Deterministic Dictionaries. J. Algorithms **41**, 69–85 (2001). https://doi.org/10.1006/jagm.2001.1171

Authors and Affiliations

Davide Bilò[1] · Keerti Choudhary[2] · Sarel Cohen[3] · Tobias Friedrich[4] · Simon Krogmann[4] · Martin Schirneck[5]

✉ Martin Schirneck
martin.schirneck@kit.edu

Davide Bilò
davide.bilo@univaq.it

Keerti Choudhary
keerti@iitd.ac.in

Sarel Cohen
sarel.cohen@runi.ac.il

Tobias Friedrich
tobias.friedrich@hpi.de

Simon Krogmann
simon.krogmann@hpi.de

1 Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, L'Aquila, Italy

2 Department of Computer Science and Engineering, Indian Institute of Technology Delhi, Delhi, India

3 Efi Arazi School of Computer Science, Reichman University, Herzliya, Israel

4 Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

5 Derpartment of Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany