Deterministic Sensitivity Oracles for Diameter, Eccentricities and All Pairs Distances

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

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

Sarel Cohen
School of Computer Science, The Academic College of Tel Aviv-Yaffo, Israel

Tobias Friedrich
Hasso Plattner Institute, University of Potsdam, Germany

Martin Schirneck
Hasso Plattner Institute, University of Potsdam, Germany

Abstract

We construct data structures for extremal and pairwise distances in directed graphs in the presence of transient edge failures. Henzinger et al. [ITCS 2017] initiated the study of fault-tolerant (sensitivity) oracles for the diameter and vertex eccentricities. We extend this with a special focus on space efficiency. We present several new data structures, among them the first fault-tolerant eccentricity oracle for dual failures in subcubic space. We further prove lower bounds that show limits to approximation vs. space and diameter vs. space trade-offs for fault-tolerant oracles. They highlight key differences between data structures for undirected and directed graphs.

Initially, our oracles are randomized leaning on a sampling technique frequently used in sensitivity analysis. Building on the work of Alon, Chechik, and Cohen [ICALP 2019] as well as Karthik and Parter [SODA 2021], we develop a hierarchical framework to derandomize fault-tolerant data structures. We first apply it to our own diameter and eccentricity oracles and then show its versatility by derandomizing algorithms from the literature: the distance sensitivity oracle of Ren [JCSS 2022] and the Single-Source Replacement Path algorithm of Chechik and Magen [ICALP 2020]. This way, we obtain the first deterministic distance sensitivity oracle with subcubic preprocessing time.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis; Theory of computation → Pseudorandomness and derandomization; Mathematics of computing → Graph algorithms

Keywords and phrases derandomization, diameter, eccentricity, fault-tolerant data structure, sensitivity oracle, space lower bound

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.22

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `http://arxiv.org/abs/2204.10679`

1 Introduction

The problems of computing *shortest paths*, *graph diameter*, and *vertex eccentricities* are fundamental in many applications of both theoretical and applied computer science. We address these problems in the setting of *fault tolerance*. The interest in this problem setting stems from the fact that most real-world networks are prone to failures. These are unpredictable but usually small in numbers and transient due to some simultaneous repair process. However, in an error-prone network, it is not always practical to recompute distances

49th International Colloquium on Automata, Languages, and Programming (ICALP 2022).
Editors: Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff;
Article No. 22; pp. 22:1–22:19

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

from scratch even if the number of edge failures is bounded. A commonly adopted solution is that of designing *f-edge fault-tolerant oracles*, that is, compact data structures that can quickly report exact or approximate extremal and pairwise distances in the network after up to f edges failed. These structures are also known as *sensitivity oracles*, where the sensitivity is the maximum number f of supported failures.

Many known fault-tolerant data structures are randomized. The algorithm that preprocesses the underlying network may depend on random bits or the correctness of the oracle's answers is only guaranteed with some probability. Besides the practical difficulties of working with (true) randomness in computing, it is an interesting question to what extend randomness as a resource is needed to obtain efficient fault-tolerant oracles. In this paper, we show that for a wide range of applications randomness can be removed with only a slight loss of performance, or even none at all in some cases. For this, we develop a novel derandomization framework and combine it with known techniques to obtain the following results.

- We present new deterministic f-edge fault-tolerant oracles that report the exact/approximate diameter and vertex eccentricities in directed graphs and we show lower bounds charting the limits of approximation vs. space and diameter vs. space trade-offs.
- We derandomize the single-failure *distance sensitivity oracle* (DSO) of Ren [32] that can report exact distance for any pair of vertices in constant time. Our result gives the first deterministic exact DSO with truly sub-cubic processing time and constant query time.
- We derandomize the algorithm of Chechik and Magen [12] for the *Single-Source Replacement Paths* (SSRP) problem on directed graphs, that is, the task of finding a shortest path from a distinguished source vertex to every target, for every possible edge failure.

We believe that our techniques are of independent interest and can help derandomize also other algorithms and data structures in the fault-tolerant domain. Throughout the paper, the underlying network is modeled by a directed graph $G = (V, E)$, possibly with weights on its edges, where V is the set of n vertices and E the set of m edges.

1.1 Diameter and Eccentricity Oracles in Directed Graphs

In Section 3, we discuss fault-tolerant oracles for the diameter and vertex eccentricities of a directed graph. We abbreviate *f-edge fault-tolerant diameter oracle* as f-FDO and *f-edge fault-tolerant eccentricity oracle* as f-FEO. In case of a single failure, $f = 1$, we shorten this to FDO and FEO, respectively. The problem of designing FDOs was originally raised by Henzinger et al. [26] and recently received some renewed interest by Bilò et al. [6]. Although the major focus of the latter work was on undirected graphs, the authors also showed that, for directed graphs, one can compute, in $\widetilde{O}(mn + n^2/\varepsilon)$ time,[1] an oracle of size[2] $O(m)$ and constant query time that guarantees a *stretch* of $1 + \varepsilon$, that is, it reports an upper bound on the value of the diameter within a factor of $1+\varepsilon$, for any $\varepsilon > 0$.

Bilò et al. [6] also gave a complementary space lower bound showing that any fault-tolerant diameter oracle with a sufficiently small stretch must take $\Omega(m)$ bits of space. However, this is not the full picture in that their construction only holds for diameter 2. We show here that in reality there is a transition happening: the larger the diameter, the more space we can save, up to a point where even $o(m)$-space oracles become possible. We aim at pinpointing this transition, starting with a generalization of the bound in [6] to diameter up to $n/\sqrt{m}$.

[1] For a non-negative function $g = g(n)$, we use $\widetilde{O}(g)$ to denote $O(g \cdot \mathsf{polylog}(n))$.
[2] Unless stated otherwise, we measure the space in the number of $O(\log n)$-bit machine words.

▶ **Theorem 1.** *Let $n, m, D \geqslant 3$ be integers with $D = O(\frac{n}{\sqrt{m}})$. Any FDO with stretch $\sigma < \frac{3}{2} - \frac{1}{D}$ on n-vertex, m-edge unweighted directed graphs of diameter D requires $\Omega(m)$ bits of space, regardless of the query time.*

Given an oracle for the fault-tolerant eccentricities with query time q, one can emulate a diameter oracle with query time nq by taking the maximum over all vertices. The information-theoretic lower bound of Theorem 1 is independent of the query time and therefore every FEO also must have size $\Omega(m)$.

Notably, Theorem 1 implies that, for any $0 < \delta \leqslant 1$ and all digraphs with $n^{1+\delta}$ edges and a relatively small diameter of $O(\sqrt{n^{1-\delta}})$, an FDO of stretch essentially $3/2$ takes $\Omega(n^{1+\delta})$ bits of space. As hinted above, this approximation vs. space trade-off no longer holds when we consider directed graphs with large diameter of $\omega(n^{5/6})$, for which we can design FDOs of quasi-linear (in n) space and negligible stretch.

▶ **Theorem 2.** *Let G be a directed graph with n vertices, m edges, and diameter $D = \omega(n^{5/6})$ and let $\varepsilon = \frac{n^{5/6}}{D} = o(1)$. There is an FDO for G with stretch $1 + \varepsilon$, preprocessing time $\widetilde{O}(mn)$, space $O(n \log^2 n)$, and constant query time.*

The gap between the stretch-size trade-offs provided in Theorem 1 and Theorem 2, respectively, suggests that there must be a threshold between $n/\sqrt{m}$ and $n^{5/6}$ where low-stretch FDOs of sub-linear size and constant query time become possible. We further narrow this gap and aim to find the smallest value for the diameter for which one can design an FDO with $o(m)$ space and constant query time. We show that this is possible for directed graphs with diameter $\omega((n^{4/3} \log n)/\sqrt{m})$. We leave it as an open problem to determine the smallest function g such that directed graphs with diameter $g(n)/\sqrt{m}$ admit an FDO with $o(m)$ space. Our results show that g is of order $\omega(n)$ and $O(n^{4/3} \log n)$.

▶ **Theorem 3.** *Let G be be a directed graph with n vertices, m edges, and diameter $\omega((n^{4/3} \log n)/(\varepsilon\sqrt{m}))$. For any $\varepsilon = \varepsilon(n, m) > 0$, there is an FDO for G with stretch $1 + \varepsilon$, preprocessing time $\widetilde{O}(mn)$, space $o(m)$, and constant query time.*

For the sake of readability, the FDOs in Section 3 are randomized. Later, in Section 4, we describe our derandomization framework and show how to apply it to both FDOs.

We now move our attention to the case multiple edge failures and give bounds in terms of f and n on the minimum space requirement of f-FDOs. Bilò et al. [6] designed an f-FDO for *undirected* graphs of stretch $f + 2$ that takes space $\widetilde{O}(fn)$. The size of this oracle is optimal up to polylogarithmic factors. In the next theorem, we show that such compact oracles are impossible for directed graphs, even when allowing arbitrarily large stretch

▶ **Theorem 4.** *Let n, f be positive integers such that $2^{f/2} = O(n)$. Any f-FDO with an arbitrary finite stretch on n-vertex directed graphs requires $\Omega(2^{f/2} n)$ bits of space, regardless of the query time.*

The lower bound of Theorem 4 marks an exponential-in-f separation between the undirected and directed setting. The directed graph used in the proof is inspired by the lower-bound construction used by Baswana et al. [3] for the *f-edge fault-tolerant Single-Source Reachability* problem. This problem asks to compute the sparsest subgraph H of a directed graph G that preserves reachability from a designated source vertex s, that is, for every vertex v and every set F of $|F| \leqslant f$ edge failures, there is path from s to v in H that avoids every edge in F if and only if there is such a path in G. Baswana et al. [3] provided a class of directed graphs for which any subgraph preserving single-source reachability with sensitivity

f has $\Omega(2^f n)$ edges. Our lower bound requires non-trivial extensions of their construction as it needs to satisfy several additional properties. For example, the directed graph in [3] has unbounded diameter, while any lower bound for FDOs requires strongly connected graphs.

We also consider the design of fault-tolerant eccentricity oracles for general directed graphs as well as directed acyclic graphs (DAGs). For the single-failure case and exact eccentricities, there is a folklore solution using the DSO of Bernstein and Karger [4] that runs in $\widetilde{O}(n^3)$ time. Henzinger et al. [26] showed how to trade stretch for running time and presented an $(1+\varepsilon)$-approximate solution with preprocessing time $\widetilde{O}(mn + n^{3/2}\sqrt{Dm/\varepsilon})$, where D denotes the diameter of the underlying graph. Both oracles build a look-up table of size $O(n^2)$ using the fact that, for any vertex v, only the failure of an edge on a shortest path tree rooted in v can change the eccentricity of v. The table allows for a constant query time but generalizing this to multiple failures $f \geqslant 2$ would take $\Omega(n^{f+1})$ space. We show how to do better than that. We give a meta-theorem that turns any exact or approximate DSO for pairwise distances into an FEO for eccentricities. Plugging in any compact DSO for multiple failures then immediately gives a space improvement for the FEO. In the following, with stretch $\sigma = 1$, we mean exact oracles.

▶ **Theorem 5.** *Let G be a (undirected or directed and possibly edge-weighted) graph with n vertices and m edges. Given access to a DSO for G with sensitivity f, stretch $\sigma \geqslant 1$, preprocessing time P, space S, and query time Q, one can construct an f-FEO for G with stretch $1 + \sigma$, preprocessing time $O(mn + P)$, space $O(n + S)$, and $O(f \cdot Q)$ query time.*

There are multiple distance oracles to choose from, all with different strengths and weaknesses. When using the DSO for of Duan and Pettie [15], we get in polynomial time a 2-approximate 2-FEO with space $O(n^2 \log^3 n)$. To the best of our knowledge, this is the first eccentricity oracle for dual failures in subcubic space. Van den Brand and Saranurak [8] gave a DSO supporting an arbitrary number of failures f. On directed graphs with integer edge weights in the range $[-M, M]$ it has polynomial space and preprocessing time, but a query time that depends both on the sensitivity and the graph size. Let $\omega < 2.37286$ be the matrix multiplication exponent [1]. Plugging the DSO in [8] into our reduction gives an f-FEO with stretch 2, $O(Mn^3)$ space[3], and query time $O(Mnf^{\omega+1})$. On undirected graphs, we can make the query time independent of n by applying the very recent DSO by Duan and Ren [17] with $O(fn^4)$ space and a query time of $f^{O(f)}$. However, the preprocessing of the latter is only polynomial for constant f. Since our reduction also applies to approximate oracles, we get, for any $f = o(\log n / \log\log n)$ and $\varepsilon > 0$, an f-FEO in polynomial time with stretch $(2 + \varepsilon)$, space $O(n^2((\log n)/\varepsilon)^f f)$ and query time $O(f^6 \log n)$ via the DSO by Chechik et al. [11].

As already mentioned above, the $\Omega(2^{f/2} n)$-bits lower bound in Theorem 4 also holds for FEOs. On DAGs, however, we can improve upon this and obtain a space requirement that is reminiscent of the one Bilò et al. [6] gave for *undirected* graphs. Note that in a DAG at most one vertex can have bounded eccentricity.

▶ **Theorem 6.** *Let G be a directed acyclic graph with, m real-weighted edges, n vertices, and a distinguished source vertex s. For any integer f, there is an f-FEO for G with stretch f, preprocessing time $\widetilde{O}(m)$, space $O(nf)$, and $O(f)$ query time.*

All the results for f-FDOs and f-FEOs are presented for edge failures. However, they also hold for vertex failures using well-known transformation techniques for directed graphs.[4]

---

[3] In [8], the space of the DSO is phrased as $O(Mn^3 \log n)$ *bits*.

[4] Indeed, we can transform the directed graph G into some graph G' with $2n$ vertices. We represent each vertex v of G with an edge (v^-, v^+) in G', and replace each edge (u, v) of G with the edge (u^+, v^-) in G'. For edge-weighted G, the weight of the new vertex-edge is set to 0 keeping eccentricities. For unweighted G, the eccentricity of v in any subgraph H of G is half the eccentricity of v^- in $H' \subseteq G'$.

1.2 Derandomization Technique

We now turn to the derandomization of fault-tolerant data structures. In Section 4, we develop the *Hierarchical Double Pivots Hitting Sets* (HDPH) algorithm as the center piece of a framework to derandomize known replacement paths algorithms and oracles. The aim of the HDPH algorithm is to compute a sequence of sets $B_1, \dots, B_{\log n} \subseteq V(G)$ such that each B_i has size $\widetilde{O}(n/2^i)$ and hits a set $\mathcal{P}_i$ of (replacement) paths each of length $\Omega(2^i)$. Unfortunately, the paths $\mathcal{P}_i$ that need to be hit by B_i are not known in advance. Our algorithm fixes this issue by iteratively computing the set of paths $\mathcal{P}_i$ using the previous sets $B_0, \dots, B_{i-1}$. The algorithm relies on the ability of the oracle we want to derandomize to be *path-reporting*, that is, to report a path representing the exact or approximate distance between the queried vertices for DSOs, the diameter for FDOs, or the vertex eccentricity for FEOs. We show how to implement the HDPH algorithm to derandomize the FDOs in Theorems 2 and 3, the DSO of Ren [32] for directed graphs with integer edge weights in the range $[1, M]$, and the algorithm of Chechik and Magen [12] for the SSRP problem in directed graphs.

Distance Sensitivity Oracles. The concept of DSOs was introduced by Demetrescu et al. [13] who showed how to compute an exact DSO of size $O(n^2 \log n)$ and constant query time in $\widetilde{O}(mn^2)$ time. Later, Bernstein and Karger [4] improved the preprocessing time to $\widetilde{O}(mn)$ and Duan and Zhang [18] reduced the space to $O(n^2)$, which is asymptotically optimal. Algebraic algorithms are known to further improve the preprocessing times, if one is willing to employ fast matrix multiplication, see [10, 23] and the references therein. For more results on approximate DSOs for both single and multiple failures, see [11, 14, 15].

We combine the HDPH framework with a recent breakthrough result by Karthik and Parter [28] to derandomize the path-reporting DSO of Ren [32] for directed graphs with integer edge weights in the range $[1, M]$. This was the first DSO that achieved a constant query time with a randomized subcubic preprocessing time of $O(Mn^{2.7233})$. On undirected graphs, the preprocessing improves to $\widetilde{O}(Mn^{(\omega+3)/2}) = O(Mn^{2.6865})$. Our derandomization of Ren's DSOs in both settings incurs a slight loss of efficiency. Nevertheless, we obtain the first deterministic DSO with constant query time and truly sub-cubic preprocessing. This improves significantly over the result by Alon, Chechik, and Cohen [2] who designed a DSO with $O(mn^{4-\alpha})$ preprocessing time and $\widetilde{O}(n^{2\alpha})$ query time, for any $\alpha \in (0, 1)$.

▶ **Theorem 7.** *For any n-vertex directed graph G with integer edge weights in the range $[1, M]$, there exists a deterministic path-reporting DSO with $O(Mn^{2.8068})$ preprocessing time and constant query time. If G is undirected, the preprocessing time decreases to $\widetilde{O}(Mn^{(\omega+6)/3}) = O(Mn^{2.7910})$.*

Recently, Gu and Ren [23] presented a new randomized DSO with a preprocessing time of $O(Mn^{2.5794})$. Unfortunately, our HDHP algorithm cannot be used to derandomize it for the following two reasons. First, the DSO of Gu and Ren is not path-reporting. Secondly, it internally relies on probabilistic polynomial identity testing. It is a long-standing open question how to derandomize this, far beyond the field of fault-tolerant data structures.

Single Source Replacement Paths Problem. In the SSRP problem we want to compute replacement paths from a designated source to each destination vertex, under each possible edge failure. Grandoni and Vassilevska Williams [21, 22] first developed an algorithm for both directed and undirected graphs with integer edge weights in the range $[1, M]$ that uses fast matrix multiplication and runs in $\widetilde{O}(Mn^{\omega})$ time. Chechik and Cohen [9] presented an $\widetilde{O}(m\sqrt{n} + n^2)$ time SSRP algorithm for *undirected* graphs that was later simplified and generalized to deal with multiple sources by Gupta et al. [24]. In this paper we use our

HDPH framework to derandomize the recent $\widetilde{O}(m\sqrt{n}+n^2)$ time randomized algorithm for *directed* graphs developed by Chechik and Magen [12], without any loss in the time complexity. Specifically, we prove the following result.

▶ **Theorem 8.** *There exists a deterministic algorithm for the Single Source Replacement Path problem in unweighted directed graphs running in time* $\widetilde{O}(m\sqrt{n}+n^2)$.

2 Preliminaries

We let $G=(V,E)$ denote a directed graph on n vertices and m edges, potentially edge-weighted by some function $w\colon E\to\mathbb{R}$. We tacitly assume that G is strongly connected, in particular, $m=\Omega(n)$. For any (weighted) directed graph H (possibly different from G), we denote by $V(H)$ and $E(H)$ the set of its vertices and edges, respectively. Let P be a path in H from $s\in V(H)$ to $t\in V(H)$, we say that P is an *s-t-path* in H. We denote by $|P|=\sum_{e\in E(P)}w(e)$ the *length* of P, that is, its total weight. If H is unweighted, we let $|P|=|E(P)|$ denote the number of its edges. For $u,v\in V(P)$, we let $P[u..v]$ denote the subpath of P from u to v. For $s,t\in V(H)$, the *distance* $d_H(s,t)$ is the minimum length of any s-t-path in H; if s and t are disconnected, we set $d_H(s,t)=+\infty$. When talking about the base graph G, we drop the subscripts if this does not create any ambiguities. The *eccentricity* of a vertex $s\in V(H)$ is $\mathrm{ecc}_H(s)=\max_{t\in V(H)}d_H(s,t)$, the *diameter* is $\mathrm{diam}(H)=\max_{s\in V(H)}\mathrm{ecc}_H(s)$. For a set $F\subseteq E(H)$ of edges, let $H-F$ be the graph obtained from H by removing all edges in F. A *replacement path* $P_H(s,t,F)$ is a shortest path from s to t in $H-F$. Its length $d_H(s,t,F)=|P_H(s,t,F)|$ is the *replacement distance*. The *fault-tolerant eccentricity* of a vertex $s\in V$ of the base graph with respect to F is $\mathrm{ecc}_{G-F}(s)$, the *fault-tolerant diameter* is $\mathrm{diam}(G-F)$.

For a positive integer f, an *f-edge fault-tolerant eccentricity oracle* (f-FEO) for G reports, upon query (s,F) with $|F|\leqslant f$, the value $\mathrm{ecc}_{G-F}(s)$. An *f-edge fault-tolerant diameter oracle* returns $\mathrm{diam}(G-F)$ upon query F. For a single edge failure, we write FEO for 1-FEO and abbreviate $F=\{e\}$ to e. For any real number $\sigma=\sigma(n,m,f)\geqslant 1$, an f-FEO is said to have *stretch* σ, or be *σ-approximate*, if the returned value $\widehat{\mathrm{ecc}}(s,F)$ on query (s,F) satisfies $\mathrm{ecc}_{G-F}(s)\leqslant\widehat{\mathrm{ecc}}(s,F)\leqslant\sigma\cdot\mathrm{ecc}_{G-F}(s)$, analogously for f-FDOs. The *preprocessing time* is the time needed to compute the data structure, its *query time* is the time needed to return an answer. For weighted graphs, we assume the weight function being such that all distances can be stored in a single word on $O(\log n)$ bits. Unless stated otherwise, we measure the *space* of the oracles in the number of words. The oracles cannot access features of graph G except those stored during preprocessing. The size of the input does not count against the space of the data structures.

3 Diameter and Eccentricity Oracles

This section discusses fault-tolerant oracles for the diameter and vertex eccentricity in directed graphs. We start by presenting space lower bounds for FDOs that guarantee a certain stretch when supporting single or multiple edge failures, respectively. In the single-failure case, the bound depends on the diameter of the graph. Roughly speaking, if the base graph has low diameter, we cannot save much space over just storing all edges. The picture changes if the diameter grows larger. We show that then we can obtain FDOs with $o(m)$ space, or even $\widetilde{O}(n)$. We then turn the discussing to eccentricity oracles for dual and multiple failures. Note that there we need to report not only one value per graph $G-F$, but one per vertex in each of those, so special techniques are needed to handle the space increase.

3.1 Space Lower Bounds for Diameter Oracles

Bilò et al. [6] showed that any FDO with a stretch $\sigma < 3/2$ on *undirected* m-edge graphs must take $\Omega(m)$ bits of space. In particular, any data structure that can distinguish between a fault-tolerant diameter of 2 and 3 has this size. Their construction transfers to directed graphs (by merely doubling each undirected edge into two directed ones). However, they do not parameterize the graphs by their diameter, namely, if the FDO has to distinguishing between diameter D and $3D/2$ for some $D \geqslant 3$. We generalize their result by showing that there is an intermediate range of D where the $\Omega(m)$-bit bound still applies. However, here the situation is more intricate in that large values of D do allow for significant space reductions.

The construction[5] by Bilò et al. [6] cannot be extended to $D \geqslant 3$. The proof of the next lemma had to be deferred to the full version due to space reasons.

▶ **Lemma 9.** *Let n, m, D be integers such that $n^2 \geqslant m \geqslant n \geqslant 4$, and $n/\sqrt{m} > D \geqslant 3$. There exists a family $\mathcal{G}$ of n-vertex directed graphs with diameter D and $\Theta(m)$ edges such that any data structure for graphs in $\mathcal{G}$ that decides whether the fault-tolerant diameter remains at D or increases to $(3D-1)/2$ for odd D (or $(3D/2)-1$ for even D) requires $\Omega(m)$ bits of space.*

It is now easy to obtain the $\Omega(m)$-bit lower bound of Theorem 1 since any FDO of stretch $\sigma < \frac{3}{2} - \frac{1}{D}$ must tell the two cases apart.

We now turn to diameter oracles that support more than one edge failure, $f > 1$. Theorem 4 states that they require space that is exponential in f, even if we allow the stretch and query time to be arbitrarily large (but finite). It follows from the next lemma together with the observation that such f-FDOs have to detect whether the edge failures disconnect the graph.

▶ **Lemma 10.** *Any data structure for n-vertex digraphs that decides for at most $2f = O(\log n)$ edge failures whether the fault-tolerant diameter is finite requires $\Omega(2^f n)$ bits of space.*

3.2 Improved Upper Bounds

The above discussion shows that for graphs with *small* diameter, there is no hope to obtain an FDO whose space is much smaller than what is needed to store the full graph. At least not while retaining good stretch at the same time. The lower bound in Theorem 1, however, breaks down for a *large* diameter. Indeed, we show next that in this regime we can do much better in terms of space, without sacrificing stretch or query time.

Theorems 2 and 3 will follow from the same construction. The initial way we present it in Lemma 12 uses randomization in the form of a well-known sampling lemma, see [22, 33]. We will later discuss how to derandomize the oracles.

▶ **Lemma 11** (Sampling Lemma)**.** *Let H be a n-vertex directed graph, $c > 0$ a positive constant, and $L \geqslant c \ln n$. Define a random set $B \subseteq V(H)$ by sampling each vertex of H independently with probability $(c \ln n)/L$. With probability at least $1 - \frac{1}{n^c}$, the cardinality of B is $O((n \log n)/L)$. Let further $\mathcal{P}$ be a set of ℓ simple paths in H, each of which spans L vertices. With probability at least $1 - \frac{\ell}{n^c}$, we have $V(P) \cap B \neq \emptyset$ for every $P \in \mathcal{P}$.*

▶ **Lemma 12.** *For any n-vertex, m-edge unweighted directed graph G with diameter $D = \omega(\log n)$ and any $\varepsilon = \varepsilon(n, m, D) > 0$, we can compute in time $\widetilde{O}(mn + n^4/(\varepsilon^3 D^3))$ an FDO with $1+\varepsilon$ stretch, $O(n + (n^{8/3} \log^2 n)/(\varepsilon^2 D^2))$ space, and constant query time.*

[5] The graph used in the proof of [6, Lemma 12] has the property that failing any edge can increase the diameter by at most 1.

Proof. Let $D = \operatorname{diam}(G)$, $b = n/(\varepsilon D)$, and $c > 0$ a sufficiently large constant. We sample a set $B \subseteq V$ of *pivots* by including each vertex independently with probability $(2bc \ln n)/n$. By Lemma 11 with $L = n/2b = \varepsilon D/2$, there are $O(b \log n)$ many pivots w.h.p.

For the graph G, compute in $\widetilde{O}(mn)$ time the $O(1)$-query time distance sensitivity oracle of Bernstein and Karger [4]. We further compute a subgraph H of G that is just the union of $|B|$ shortest-path trees, one rooted at each pivot. We iterate over the edges of H and compute the collection $\mathcal{X}$ of all those $e \in E(H)$ such that $d(b_1, b_2, e) > d(b_1, b_2)$ for some pair $(b_1, b_2) \in B \times B$. The time to compute $\mathcal{X}$ is $O(n|B|^3) = \widetilde{O}(n^4/(\varepsilon^3 D^3))$ since processing an edge in H requires $|B|^2$ calls to the DSO. Observe that any subgraph of G that exactly preserves distances between all pairs in $B \times B$ must contain all the edges of $\mathcal{X}$. Bodwin [7] showed that there are distance-preserving subgraphs with respect to $B \times B$ with at most $O(n + n^{2/3}|B|^2)$ edges. Thus, the size of $\mathcal{X}$ is bounded by $O(n + n^{2/3}|B|^2)$.

Next, we build a dictionary $\mathcal{D}_{\mathcal{X}}$ in which we store the the edges in $\mathcal{X}$ together with the maximum distance between any pair of pivots if the edge fails (or $\operatorname{diam}(G)$ if this is larger). In other words, for each $e \in \mathcal{X}$, we store $\phi(e) = \max\{\max_{b_1,b_2 \in B} d(b_1, b_2, e), \operatorname{diam}(G)\}$. Let $\mathcal{Y}$ be the set of all edges in E such that $G - e$ is no longer strongly connected. We build a dictionary $\mathcal{D}_{\mathcal{Y}}$ in which we store information about the edges $\mathcal{Y}$. It is well-known that $\mathcal{Y}$ contains $O(n)$ edges and can be computed in time $O(m)$ [27].

Recall that $b = n/(\varepsilon \operatorname{diam}(G))$. The oracle's output $\widehat{D}(e)$ is defined as follows: if $e \in \mathcal{Y}$, then $\widehat{D}(e) = \infty$; if $e \in \mathcal{X}$, $\widehat{D}(e) = \phi(e) + n/b$; otherwise, the oracle outputs $\widehat{D}(e) = \operatorname{diam}(G) + n/b = (1 + \varepsilon) \operatorname{diam}(G)$.

Evidently, the oracle is correct for all $e \in \mathcal{Y}$. It is also easy to verify that all outputs are at most $\phi(e) + n/b \leqslant \operatorname{diam}(G - e) + n/b = \operatorname{diam}(G - e) + \varepsilon \operatorname{diam}(G) \leqslant (1 + \varepsilon) \operatorname{diam}(G - e)$. To prove that they are also at least $\operatorname{diam}(G - e)$, consider a vertex pair $(u, v) \in V \times V$ such that $d(u, v, e) = \operatorname{diam}(G - e) < \infty$. With high probability[6] by Lemma 11, there exists a shortest u-v-path in $G - e$ and two pivots $b_u, b_v \in B$ on that path such that $d(u, b_u, e), d(b_v, v, e) \leqslant L = n/2b$. We have $\operatorname{diam}(G-e) = d(u, b_u, e)+d(b_u, b_v, e)+d(b_v, v, e)$. Suppose $e \notin \mathcal{X}$. Then, $d(b_u, b_v, e) = d(b_u, b_v) \leqslant \operatorname{diam}(G)$ holds and therefore $\operatorname{diam}(G - e) \leqslant \operatorname{diam}(G) + n/b = \widehat{D}(e)$. If $e \in \mathcal{X}$, then $d(b_u, b_v, e) \leqslant \phi(e)$ and $\operatorname{diam}(G - e) \leqslant \phi(e) + n/b = \widehat{D}(e)$.

There are k-element dictionaries of size $O(k)$ and $O(1)$ query time computable in time $\widetilde{O}(k)$ [25]. The dictionaries have total size $O(n+n^{2/3}|B|^2) = O(n+(n^{8/3} \log^2 n)/(\varepsilon^2 D^2))$. ◀

The oracle in Lemma 12 can also be extended to handle vertex failures. The only modification required is to add to set $\mathcal{X}$ those vertices $v \in V$ that satisfy $d(b_1, b_2, v) > d(b_1, b_2)$ for some $(b_1, b_2) \in B \times B$, and to add to $\mathcal{Y}$ to be those vertices v for which $G - v$ is not strongly connected. Suppose $D = \omega(n^{5/6})$, inserting any $\varepsilon \geqslant n^{5/6}/D = o(1)$ above gives an FDO with near linear space and $1 + o(1)$ stretch that is computable in time $\widetilde{O}(mn)$, which proves Theorem 2. Furthermore, for graphs with diameter $\omega((n^{4/3} \log n)/(\varepsilon\sqrt{m}))$, we obtain in $\widetilde{O}(mn)$ time an FDO with constant query time and $o(m)$ space (Theorem 3).

3.3 Eccentricity Oracles

We now prove Theorem 5 that constructs an f-edge fault-tolerant eccentricity oracle from a DSO supporting f failures. The improved f-FEO for DAGs can be found in the full version.

[6] We say an event occurs *with high probability* (w.h.p.) if it has success probability $1 - n^{-c}$ for some constant $c > 0$ that can be made arbitrarily large.

Let $\mathcal{D}$ be a DSO with sensitivity f and stretch σ that, on (un-)directed possibly weighted graphs, can be computed in time P, uses S space, and has a query time of Q. For any given source $s \in V$ and query set F of $|F| \leqslant f$ edges, our oracle reports an $(1+\sigma)$-approximation of the eccentricity of s in $G - F$. We simply store $\mathcal{D}$ and, for each $x \in V$, the value $ecc_G(x)$. All eccentricities in the base graph G can be obtained with a BFS from each vertex in $O(mn)$.

Upon query $(s, F = \{(x_1, y_1), \dots, (x_f, y_f)\})$, we use $\mathcal{D}$ to compute $d(s, y_i, F)$, for all $1 \leqslant i \leqslant f$. Our estimate is $\widehat{\mathrm{ecc}}_{G-F}(s) = \mathrm{ecc}_G(s) + \max_{1 \leqslant i \leqslant f} d(s, y_i, F)$. The time taken to compute $\widehat{\mathrm{ecc}}_{G-F}(s)$ is $O(f \cdot Q)$ and the space requirement of the oracle is $O(n + S)$.

Now we show that $\widehat{\mathrm{ecc}}_{G-F}(s)$ is a $(1 + \sigma)$-approximation of $\mathrm{ecc}_{G-F}(s)$. Let F_0 be the subset of F consisting of those edges in F that lie on some shortest-path tree T rooted in s. If F_0 is empty, we immediately get $\mathrm{ecc}_{G-F}(s) = \mathrm{ecc}_G(s) \leqslant \widehat{\mathrm{ecc}}_{G-F}(s)$. Otherwise, for any $v \in V$, either $d(s, v) = d(s, v, F)$ or there exists an $(x, y) \in F_0$ such that y is an ancestor of v in T. In this latter case $d(y, v, F) \leqslant \mathrm{ecc}_G(s)$. This proves that $d(s, v, F) \leqslant d(s, y, F) + d(y, v, F) \leqslant d(s, y, F) + \mathrm{ecc}_G(s) \leqslant \widehat{\mathrm{ecc}}_{G-F}(s)$. Thus, $\mathrm{ecc}_{G-F}(s) \leqslant \widehat{\mathrm{ecc}}_{G-F}(s)$. Next observe that $\mathrm{ecc}_G(s) \leqslant \mathrm{ecc}_{G-F}(s)$ and $\max_{1 \leqslant i \leqslant f} d(s, y_i, F) \leqslant \sigma \cdot \mathrm{ecc}_{G-F}(s)$, which proves that $\widehat{\mathrm{ecc}}_{G-F}(s) \leqslant (1 + \sigma) \cdot \mathrm{ecc}_{G-F}(s)$.

4 Derandomization Framework

The fault-tolerant diameter oracles in Theorems 2 and 3 are randomized. They both follow from Lemma 12 which in turn relies on a random hitting set to intersect all replacement paths of a certain length. In fact, many more data structures and algorithms in the fault-tolerant setting follow a sampling-based approach similar to Lemma 11, see e.g. [9, 12, 22, 32, 33, 35]. It is an interesting question whether these algorithms can be derandomized efficiently. Currently there is no single approach to derandomize Lemma 11 in the same $O(n)$ time it uses to go through all vertices. Therefore, the literature focuses on the specific applications. The goal is to replace the sampling step by a deterministic construction of the hitting set that, while taking $\omega(n)$ time, does not (or only marginally) increase the asymptotic running time of the whole algorithm. Recently, there was some progress on notable special cases. Karthik and Parter [28] gave a derandomization for the algebraic version of the distance sensitivity oracle of Weimann and Yuster [35] with a slightly higher running time (for a detailed discussion see Lemma 15 below). Bilò et al. [5] derandomized the SSRP algorithms of Grandoni and Vassilevska Williams [22] as well as Chechik and Cohen [9]. Their derandomization succeeds in the same time bounds as the original randomized algorithm, but the technique only works for undirected graphs. Here, we develop a framework for directed graphs. We first apply it to our own FDOs and then show its versatility by also derandomizing the DSO of Ren [32] and the SSRP algorithm of Chechik and Magen [12].

We build on the work of Alon, Chechik, and Cohen [2]. We first review some technical details of their result and then describe our additions. For now, we assume the base graph G to be unweighted and only later (in Section 5) incorporate positive integer edge weights. For concreteness, consider the task in Lemma 12 of finding a set $B \subseteq V$, the pivots, such that for all $s, t \in V$ and edge $e \in E$ with replacement distance $d(s, t, e)$ at least $L = \varepsilon \operatorname{diam}(G)/2$, there exists some replacement path $P(s, t, e)$ that contains a pivot $x \in B$. Other fault-tolerant algorithms pose similar requirements on B. The technique in [2] consists of computing a small set of *critical* paths, much smaller than the set of all $O(mn^2)$ replacement paths. Once we have those, a hitting set can be computed with the folklore greedy algorithm, called GreedyPivotSelection in [2], that always selects a vertex that is contained in the most unhit paths.[7] Alternatively, one can use the blocker set algorithm of King [29].

[7] To achieve the performance of Lemma 13, one has to truncate all paths by selecting L vertices arbitrarily from each $P \in \mathcal{P}$. This is non-issue for us as, by construction, all our paths will have length $\Theta(L)$.

▶ **Lemma 13** (Alon, Chechik, and Cohen [2]). *Let $1 \leqslant L \leqslant n$ and $1 \leqslant q = \mathsf{poly}(n)$ be two integers. Let $P_1, \ldots, P_q \subseteq V$ be sets of vertices that, for every $1 \leqslant k \leqslant q$, satisfy $|P_k| \geqslant L$. The algorithm* GreedyPivotSelection *computes in time $\widetilde{O}(qL + n^2/L)$ a set $B \subseteq V$ of $|B| = O((n \log q)/L) = \widetilde{O}(n/L)$ pivots such that, for every index k, it holds that $B \cap P_k \neq \emptyset$.*

The crucial part is to quickly find the paths P_k such that hitting them is sufficient to hit all long replacement path. Of course, this could be done by computing all-pairs shortest paths in each graph $G-e$ in total time $\widetilde{O}(m^2 n)$ using Dijkstra's algorithm (or $\widetilde{O}(mn^{2.5302})$ if one is willing to use fast rectangular matrix multiplication [20, 36]). However, this is much more than the $\widetilde{O}(mn + n^4/(\varepsilon^3 \operatorname{diam}(G)^3))$ time bound we had in Lemma 12. For the applications in [2], a single set of paths and therefore a single hitting set was sufficient. Bilò et al. [5] (with slightly different requirements on the set B) were able to make do with three sets, exploiting the undirectedness of the underlying graph.

We extend this to directed graphs using a hierarchical approach to find the critical paths. Observe how the length parameter L in Lemma 13 serves two roles. The longer the paths, the longer it takes to compute B, but the *fewer* vertices suffice to intersect all paths. Additionally, we have to compute the set of critical paths which takes (at least) linear time in their length. So L has to fall just in the right range for the computation to be fast. To achieve this, we use an exponentially growing sequence of lengths $L_1, L_2, \ldots, L_{O(\log n)}$ and, instead of a single set, compute a sequence $B_1, B_2, \ldots$ of exponentially shrinking sets such that, in the i-th stage, B_i hits, again for all $s, t \in V$ and $e \in E$, some replacement path of length at least L_i. However, this poses some new difficulties because now the collection of critical paths has to be computed step by step. Imagine in the i-th stage, we have already obtained the all the subsets $\mathcal{P}_j$, $j < i$, of paths with respective lengths L_j. The key observation is that the hitting sets B_j from the previous rounds carry valuable information that can be harnessed to find the new set $\mathcal{P}_i$ faster, this then offsets the run time penalty incurred by the greater length of the new paths.

The HDPH Algorithm. We now describe the Hierarchical Double Pivots Hitting Sets (HDPH) algorithm that makes these ideas concrete. It can be seen as a "reference implementation" of the framework. For a specific application, one still has to adapt the details. The algorithm is more general than what is needed for diameter oracles in Theorems 2 and 3. For example, it also pertains to vertex failures. Later, in Section 5, we show an example how to modify the algorithm for other problems (more are found in the full version).

Let $C \geqslant 3/2$ be a constant. The aim of the HDPH algorithm is to compute a sequence of sets $B_1, \ldots, B_{\lceil \log_C n \rceil} \subseteq V$ of size $|B_i| = \widetilde{O}(n/C^i)$ such that for all vertices $s, t \in V$ and failure $f \in E \cup V$ with $d(s, t, f) \in (C^i, C^{i+1}]$ there exists a replacement path $P(s, t, f)$ that contains a pivot $z \in B_i$. It assumes access to the "APSP data" of the original graph G, that is, the distance $d(s, t)$ for all s, t and a corresponding shortest path $P(s, t)$. Also, it requires a deterministic path-reporting distance sensitivity oracle with constant query time (both for the distance and each reported edge) as a black box.

The HDPH algorithm is sketched in Algorithm 1. In lines 1 and 2 it initializes $B_i = V$ for $i \leqslant 2$. In lines 3-12, for $3 \leqslant i \leqslant \lceil \log_C n \rceil$, we iteratively compute the hitting sets B_i by using the hitting sets from the previous 3 iterations to obtain a set of shortest and replacement paths $\mathcal{P}_i$ of length $\Theta(n/C^i)$ that one needs to hit, and then use the greedy algorithm GreedyPivotSelection to compute the set of pivots B_i which hits this set of paths $\mathcal{P}_i$. The paths are defined as follows. First, in line 4 we add to $\mathcal{P}_i$ shortest paths $P(x, y)$ whose length is in the range $(C^i, C^{i+1}]$ such that $x, y \in B_{i-3} \cup B_{i-2} \cup B_{i-1}$ are pivots from the last 3 iterations. Then, in lines 5-11, for every pair of pivots $x, y \in B_{i-3} \cup B_{i-2} \cup B_{i-1}$ whose shortest path

Algorithm 1 Hierarchical Double Pivots Hitting Sets (HDPH) Algorithm.

Input: APSP data and a deterministic path-reporting DSO with $O(1)$ query time.
Output: The hitting sets $B_0, \dots, B_{\lceil \log_C n \rceil}$.

1 **for** $i \in [0, 2]$ **do**
2 $B_i \leftarrow V$
3 **for** $i \in [3, \lceil \log_C n \rceil]$ **do**
4 Let $\mathcal{P}_i = \{P(x,y) \mid x, y \in B_{i-3} \cup B_{i-2} \cup B_{i-1}$ such that $d(x,y) \in (C^{i-6}, C^{i+1}]\}$
5 **for** $x, y \in B_{i-3} \cup B_{i-2} \cup B_{i-1}$ **do**
6 **if** $d(x,y) \leqslant C^{i+1}$ **then**
7 **for** $f \in E(P(x,y)) \cup V(P(x,y))$ **do**
8 query the DSO for $d(x,y,f)$
9 **if** $d(x,y,f) \in (C^{i-6}, C^{i+1}]$ **then**
10 query the DSO for $P(x,y,f)$
11 $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \{P(x,y,f)\}$
12 $B_i \leftarrow$ GreedyPivotSelection$(\mathcal{P}_i)$
13 **return** $B_0, \dots, B_{\lceil \log_C n \rceil}$

$P(x,y)$ is of length at most C^{i+1}, and for every edge or every $f \in E(P(x,y)) \cup V(P(x,y))$ we query the DSO with (x,y,f) to compute the distance $d(x,y,f)$. If $d(x,y,f) \in (C^{i-6}, C^{i+1}]$ then we use the DSO to also report a replacement path $P(x,y,f)$ and add it to $\mathcal{P}_i$.

The next lemma proves the properties of the resulting hitting sets and the run time.

▶ **Lemma 14.** *Given the APSP data and a deterministic path-reporting DSO with $O(1)$ query time, the HDPH algorithm deterministically computes, in $\widetilde{O}(n^2)$ time, all the hitting sets B_i, with $0 \leqslant i \leqslant \lceil \log_C n \rceil$. For every $0 \leqslant i \leqslant \lceil \log_C n \rceil$, it holds that $|B_i| = \widetilde{O}(n/C^i)$. For every pair of vertices $s, t \in V$ and for every failing edge or vertex $f \in E \cup V$ such that $d(s,t,f) \in (C^i, C^{i+1}]$ there exists a pivot $z \in B_i$ such that $d(s,t,f) = d(s,z,f) + d(z,t,f)$. Finally, for every pair of vertices $s, t \in V$ such that $d(s,t) \in (C^i, C^{i+1}]$, there exists a pivot $z \in B_i$ such that $d(s,t) = d(s,z) + d(z,t)$.*

Proof. We first prove by induction that for every $i \in [0, \lceil \log_C n \rceil]$ it holds that $|B_i| = \widetilde{O}(n/C^i)$. The claim trivially holds for $i \leqslant 2$ as $B_0 = B_1 = B_2 = V$. For the inductive step, we assume that $|B_j| = \widetilde{O}(n/C^j)$ for every $j < i$. We show that the set of paths $\mathcal{P}_i$ contains $\widetilde{O}(n^2/C^i)$ paths, each of length $\Theta(C^i)$, and thus the result of the greedy algorithm $B_i \leftarrow$ GreedyPivotSelection$(\mathcal{P}_i)$ contains, by Lemma 13, at most $\widetilde{O}(n/C^i)$ vertices. Moreover, the runtime of the GreedyPivotSelection procedure is $\widetilde{O}(n^2)$.

For every $s, t \in V$, let $P(s,t)$ denote the shortest s-t-path in the APSP data. There are two places where paths are added to $\mathcal{P}_i$. In line 4, the algorithm adds shortest paths between vertices $x, y \in B_{i-3} \cup B_{i-2} \cup B_{i-1}$ whenever $d(x,y) = |P(x,y)| \in (C^{i-6}, C^{i+1}]$, and by the induction hypothesis there are $\widetilde{O}(n^2/C^{2(i-1)}) = \widetilde{O}(n^2/C^i)$ such pairs of vertices (since $|B_j| = \widetilde{O}(n/C^j)$ for every $j < i$). Thus, the claim holds for the paths in line 4. In line 11, the algorithm adds paths $P(x,y,f)$ to $\mathcal{P}_i$ only for pairs $x, y \in B_{i-3} \cup B_{i-2} \cup B_{i-1}$ and edges or vertices $f \in E(P(x,y)) \cup V(P(x,y))$ with $d(x,y) \leqslant C^{i+1}$, there are $\widetilde{O}(C^{i+1} \cdot (n^2/C^{2(i-1)})) = \widetilde{O}(n^2/C^i)$ such triples (x,y,f). The only paths added there are such that $d(x,y,f) \in (C^{i-6}, C^{i+1}]$ (due to the condition in line 9) and thus the length of $P(x,y,f)$ is $\Theta(C^i)$. So the claim holds here as well.

Next, we prove that the runtime of the algorithm is $\widetilde{O}(n^2)$. We show that a single iteration of the for loop in lines 4-16 takes $\widetilde{O}(n^2)$ time, and as there are $O(\log n)$ iterations for $i \in [3, \lceil \log_C n \rceil]$. The number of pairs $x, y \in B_{i-3} \cup B_{i-2} \cup B_{i-1}$ is $\widetilde{O}(n^2/C^{2(i-1)})$. The inner loop in lines 7-11 is executed only if $d(x,y) \leqslant C^{i+1}$, therefore the number of edges $e \in P(x,y)$ is bounded by C^{i+1} and hence the loop is executed at most $O(C^i)$ times. Each iteration of this loop uses the black-box DSO to compute $d(x,y,f)$ in $O(1)$, and only if $d(x,y,f) \in (C^{i-6}, C^{i+1}]$ then we use the DSO to actually obtain the path $P(x,y,f)$ in $O(|P(x,y,f)|) = O(C^i)$ time. This gives $\widetilde{O}(n^2)$ for the second-most outer loop. We have already seen that computing $\mathsf{GreedyPivotSelection}(\mathcal{P}_i)$ in line 12 takes $\widetilde{O}(n^2)$ time as well.

We claim that for all $s, t \in V$ and every edge or vertex $f \in E \cup V$ such that $d(s,t,f) \in (C^i, C^{i+1}]$, there exists a pivot $z \in B_i$ such that $d(s,t,f) = d(s,z,f) + d(z,t,f)$. That means, there is some s-t-replacement path that contains z. This is clear for $i \leqslant 2$. Let $3 \leqslant i \leqslant \lceil \log_C n \rceil$ and suppose the claim holds for every $j < i$. Let $P(s,t,f) = (v_0 = s, v_1, \ldots, v_k = t)$ be an replacement path with $k = d(s,t,f) \in (C^i, C^{i+1}]$. We define the prefix $P_1 = P(s,t,f)[s..v_{\lceil k/C^3 \rceil}]$ and suffix $P_2 = P(s,t,f)[v_{\lceil (1-1/C^3)k \rceil}..t]$. Both subpaths have length in $(C^{i-3}, C^{i-2}]$. It follows that there are pivots $x_1, x_2 \in B_{i-3}$ with $x_1 \in V(P_1)$, $x_2 \in V(P_2)$. (Strictly speaking, we are merely guaranteed *some* s-$v_{\lceil k/C^3 \rceil}$-replacement path that contains x_1, but we can choose $P(s,t,f)$ so that its prefix is that path; same with P_2.)

Let $P(x_1, x_2, f)$ be the replacement paths returned by the DSO on query (x_1, x_2, f). We claim that it is added to $\mathcal{P}_i$. Observe that $d(x_1, x_2, f) \geqslant d(s,t,f) - |P_1| - |P_2| \geqslant (1 - \frac{2}{C^2})C^i > C^{i-6}$, where we used the assumption $C \geqslant 3/2$ and thus $1 - \frac{2}{C^2} > C^{-6}$. Also, we have $d(x_1, x_2) \leqslant d(x_1, x_2, f) \leqslant d(s,t,f) \leqslant C^{i+1}$. If $d(x_1, x_2, f) = d(x_1, x_2)$, we may assume $P(x_1, x_2, f) = P(x_1, x_2)$, whence it was added in line 4. Otherwise, the failure $f \in V(P(x_1, x_2))$ is on the path. Since $x_1, x_2 \in B_{i-3} \cup B_{i-2} \cup B_{i-1}$, $d(x_1, x_2) \leqslant C^{i+1}$, and $d(x_1, x_2, f) \in (C^{i-6}, C^{i+1}]$ the path $P(x_1, x_2, f)$ is indeed added to $\mathcal{P}_i$ in line 11. Due to $B_i \leftarrow \mathsf{GreedyPivotSelection}(\mathcal{P}_i)$, there exists a vertex $z \in B_i$ such that z is on the path $P(x_1, x_2, f) \subseteq P(s,t,f)$ and thus $d(s,t,f) = d(s,z,f) + d(z,t,f)$.

The proof that for all $s, t \in V$ with $d(s,t) \in (C^i, C^{i+1}]$, there exists a pivot $z \in B_i$ such that $d(s,t) = d(s,z) + d(z,t)$ follows the same argument but is somewhat simpler because the subpaths P_1 and P_2 are guaranteed to be added in line 4. ◀

Derandomizing Theorems 2 and 3. Recall that the oracle in Lemma 12 has preprocessing time $\widetilde{O}(mn + n^4/(\varepsilon^3 \operatorname{diam}(G)^3))$. For its derandomization, and that of Theorems 2 and 3, it is enough to choose $C = 2$, compute APSP *only* in the original graph G, and preprocess the DSO of Bernstein and Karger[8] [4], which takes $\widetilde{O}(mn)$ time. Let i^* be the largest integer i such that $2^i < L = \varepsilon \operatorname{diam}(G)/2$. The set B_{i^*} then hits, for all $s, t \in V$ and $e \in E$ with $d(s,t,e) = \Theta(L)$, some replacement path $P(s,t,e)$, and it has the desired cardinality $\widetilde{O}(n/L)$.

5 Derandomizing Existing Sensitivity Oracles and Algorithms

We now show how the HDPH algorithm can be adapted to derandomize existing sensitivity oracles. In addition to our own technique, we also extensively use a recent breakthrough by Karthik and Parter [28] in the derandomization of fault-tolerant algorithms. We combine both tools and apply them to the distance sensitivity oracle of Ren [32] and the SSRP algorithm of Chechik and Magen [12] In the main part, we concentrate on the DSO because we think that it is a good illustration of the combination of our work and that of Karthik and Parter [28]. The treatment of the SSRP algorithm had to be deferred to the full version.

[8] Bernstein and Karger [4] derandomized their own DSO using a technique by King [29].

5.1 The Distance Sensitivity Oracle of Ren

We start with the oracle of Ren [32]. Recall that, for any two vertices $s, t \in V$ and edge $e \in E$, the replacement distance $d(s, t, e)$ is the length of a shortest s-t-path in $G - e$. A distance sensitivity oracle (DSO) is a data structure that answers query (s, t, e) with $d(s, t, e)$. Ren [32] presented an algebraic DSO with a randomized preprocessing time of $O(Mn^{2.7233})$ on graphs with positive integer edge weights in the range $[1, M]$ and $\widetilde{O}(Mn^{(\omega+3)/2}) = O(Mn^{2.6865})$ time on undirected graphs. Notably, this was the first DSO with both constant query time and subcubic preprocessing, improving over previous work [2, 4, 22, 35]. We derandomize it with a slight increase in running time and obtain a deterministic DSO in time $O(Mn^{2.8068})$ on directed graphs and $\widetilde{O}(Mn^{(\omega+6)/3}) = O(Mn^{2.7910})$ on undirected graphs.

The construction starts with a Core oracle that only reports very small distances, this is then grown iteratively to cover longer paths until the distance between all pairs of vertices are correctly determined. More formally, for a positive real r, let an *r-truncated DSO* report, upon query (s, t, e), the value $d(s, t, e)$ if it is at most r, and $+\infty$ otherwise. The Core is an n^α-truncated DSO for some carefully chosen exponent $\alpha \in (0, 1)$. Each iteration invokes the procedure Extend to turn an r-truncated DSO into an $(3/2)r$-truncated DSO. Note that we can assume $M = \widetilde{O}(n^{(3-\omega)/2})$ as otherwise the deterministic oracle in [4] with an $\widetilde{O}(mn)$ preprocessing time already achieves $\widetilde{O}(Mn^{(\omega+3)/2})$, even on directed weighted graphs. Hence, $\log_{3/2}(Mn) = O(\log n)$ rounds of growing suffice to built the full oracle.

The iterative approach has the advantage that r-truncated DSOs for small r can be computed fast. A bridging-set idea, see [36], is used for the extension. This significantly increases the query time as the oracle has to cycle through the whole bridging set to compute the distance. Ren [32] uses a clever observation, there attributed to Bernstein and Karger [4], to reduce the query time of the extended DSO back to a constant, called the Fast procedure.

Randomness is employed at two points. First, the Core uses a series random subgraphs of G. Secondly, Extend randomly samples a set of pivots to hit all replacement paths of length between r and $(3/2)r$. The subsequent reduction in query time is deterministic.[9] The Core can be derandomized using a recent result by Karthik and Parter [28]. To derandomize Extend, we adapt our technique introduced above. The key differences are that we now have to take care of the edge weights, that is, the number of vertices of a path may be much smaller than its length. Also, due to the iterative approach of not only the derandomization but the actual construction via truncated DSOs, we cannot assume to have access to all relevant paths right from the beginning. Instead, we have to make sure that all intermediary oracles are *path-reporting* and that for the construction of the current hitting set we only use paths of length at most r. The deterministic Core oracle hinges on the following lemma.

▶ **Lemma 15** (Karthik and Parter [28]). *Given a (possibly weighted) graph G on n vertices and a positive real $r = n^\alpha$ for some $\alpha \in (0, 1)$, there is a deterministic algorithm computing $k = O(r^2)$ spanning subgraphs $G_1, \dots, G_k$ of G in time $\widetilde{O}(kn^2)$ such that for any pair of vertices $s, t \in V$, edge $e \in E$, and replacement path $P(s, t, e)$ on at most r edges, there exists an index i such that G_i does not contain the edge e but all edges of $P(s, t, e)$.*

This derandomizes a construction by Weimann and Yuster [35] with the crucial difference that the latter is only required to produce subgraphs such that for all pairs of vertices s, t and edges e that admit possibly multiple replacement paths on at most r edges *at least one* (instead of all) of them survives in one of the graphs G_i in which e was removed. This

[9] The relevant [32, Section 3] is phrased as randomized, but based on the derandomizable algorithm in [4].

relaxed condition is actually enough to build an r-truncated DSO and allows one to make do with only $\widetilde{O}(r)$ random subgraphs, while we have $O(r^2)$ deterministic graphs. See also the discussion in Section 1.3 of [28]. This is the sole reason for the increased running time compared to the original result of Ren [32].

Given a graph G with integer edge weights in the range $[1, M]$, we invoke Lemma 15 to obtain the subgraphs G_i. Recall that $r = n^\alpha$ and let $\omega(1-\alpha)$ be the infimum over all w such that rectangular integer matrices with dimensions $n \times n^{1-\alpha}$ and $n^{1-\alpha} \times n$ can be multiplied using $O(n^w)$ arithmetic operations, $\omega = \omega(1)$ is the usual square matrix multiplication coefficient. Using a variant of Zwick's algorithm [36],[10] we compute APSP restricted to paths on at most r edges in time $\widetilde{O}(Mn^{\omega(1-\alpha)}r)$ per subgraph. If G is undirected, then this can be done faster, namely, in $\widetilde{O}(Mn^\omega)$ per graph with the algorithm of Shoshan and Zwick [34]. Both algorithms in [34, 36] can be adjusted to also compute the actual paths, represented as predecessor trees, which increases the running time only by logarithmic factors.

To answer a query (s, t, e) we cycle through the G_i and, in case the edge e is missing, retrieve the distance $d_{G_i}(s, t)$. By Lemma 15, the minimum over all retrieved distances is the correct replacement distance $d(s, t, e)$. If this minimum is larger than r or no distance has been retrieved at all (as the paths take more than r edges), we return $+\infty$. Since the edge weights are positive integers, every path of *length* at most r uses at most r edges, so we indeed obtain an r-truncated DSO. If an actual replacement path is requested, we return a shortest s-t-path in one of the G_i that attain the minimum. The resulting oracle has query time $\widetilde{O}(r^2)$ and a $\widetilde{O}(n^2r^2 + Mn^{\omega(1-\alpha)}r^3) = \widetilde{O}(Mn^{\omega(1-\alpha)}r^3)$ preprocessing time on directed graphs (using $\omega(1-\alpha) \geqslant 2$). On undirected graphs, this improves to $\widetilde{O}(n^2r^2 + Mn^\omega r^2) = \widetilde{O}(Mn^\omega r^2)$.

As a technical subtlety, the Fast procedure needed to reduce the query time requires *unique* shortest paths[11] of the original graph G. They can be computed in time $\widetilde{O}(M^{1/2}n^{(\omega+3)/2})$ [16, 32]. We will see later that this is not the bottleneck of the preprocessing.

▶ **Lemma 16** (Ren [32]). *From a directed graph G with integer edge weights in $[1, M]$, unique shortest paths, and an r-truncated DSO with preprocessing time P and query time Q, one can built in deterministic time $P + \widetilde{O}(n^2) \cdot Q$ a r-truncated DSO for G with $O(1)$ query time.*

Without access to unique paths, the running time increases to $P + \widetilde{O}(Mn^2) \cdot Q$, see [31]. If the oracle with query time Q (for the distance) is path-reporting (in $O(1)$ time per edge), then the new oracle is path-reporting with $O(1)$ query time (for distances and edges) [32].

We now turn to the main part, where we derandomize the Extend procedure that turns an r-truncated DSOs into $(3/2)r$-truncated DSOs. We adapt our technique to the iterative manner of construction and to the integer weights on the edges. In each stage, we only have access to a truncated DSO. Still, we show how to deterministically compute a sequence $B_1, B_2, \ldots$ of smaller and smaller sets, where B_i is used to derandomize the i-th application of Extend. Again, the construction of B_i depends on the previous sets of pivots, namely, on B_{i-2}. We first describe how to obtain the B_i satisfying certain useful properties and afterwards verify that these properties indeed suffice to make Extend deterministic.

▶ **Lemma 17.** *Let $r_1 \geqslant 1$ be a real number and define $r_{i+1} = (3/2)r_i$. For a graph G with integer edge weights in $[1, M]$, let $\{B_i\}_{i \geqslant 1}$ be a family of subsets of V, such that, for each i,*

(i) $|B_i| = \widetilde{O}(Mn/r_i)$;

[10] The algorithm in [36] is also phrased as randomized, in the same work it is explained how to derandomize it, increasing the running time only by polylogarithmic factors. The same holds for [34].

[11] By *unique shortest paths*, we mean a collection $\mathcal{P}$ containing one shortest path for each pair of vertices such that, for all $s, t \in V$, if $P(s, t) \in \mathcal{P}$ is the shortest path from s to t, then for every vertex u on $P(s, t)$, the path $P(s, u) \in \mathcal{P}$ is the prefix $P(s, t)[s..u]$ and $P(u, t) \in \mathcal{P}$ is the suffix $P(s, t)[u..t]$.

(ii) *for every pair of* $s,t \in V$ *and* $e \in E$ *with* $r_i/2 - M \leqslant d(s,t,e) \leqslant r_i$*, there exists a replacement path* $P(s,t,e)$ *that contains a vertex of* B_i*.*

With access to the shortest paths, a path-reporting r_i*-truncated DSO with* $O(1)$ *query time, and the sets* B_j *with* $j < i$*, one can compute each* B_i *deterministically in time* $\widetilde{O}(M^2n^2 + n^2r_1^2)$*.*

Proof. The proof is by induction over i. For the construction of B_i, we use the previous set B_{i-2}. We set $B_{-1} = B_0 = V$ to unify notation. Following the outline of the derandomization technique, we first assemble a set $\mathcal{P}$ of paths and then greedily compute a hitting set.

For each pair of vertices $x, y \in B_{i-2}$, we check whether the x-y-distance in the base graph G is at most r_i and, if so, retrieve a shortest path $P(x,y)$. If $P(x,y)$ additionally has length at least $r_i/18$, we add it to $\mathcal{P}$. For each edge e on $P(x,y)$ (regardless of the path being added to $\mathcal{P}$), we query the r_i-truncated DSO whether the replacement distance is $d(x,y,e) \in [\frac{r_i}{18}, r_i]$. If so, we request a corresponding replacement path $P(x,y,e)$ to add it to $\mathcal{P}$.

Due to the positive weights, those paths have at most r_i edges and can be obtained in time $O(r_i)$. Assembling $\mathcal{P}$ thus takes time $O(|B_{i-2}|^2 r_i^2)$. If $i \leqslant 2$, this is $O(n^2 r_1^2)$ since $r_2 = (3/2)r_1$. For $i \geqslant 3$, we get $\widetilde{O}((Mn/r_{i-2})^2 \cdot r_i^2) = \widetilde{O}(M^2n^2)$ instead.

We deterministically compute a hitting set B_i for $\mathcal{P}$. Since $\mathcal{P}$ contains at most $|B_{i-2}|^2 \cdot r_i$ paths with at least $r_i/(18M)$ edges each, whence $\Omega(r_i/M)$ vertices, the set B_i has $\widetilde{O}(n/(r_i/M)) = \widetilde{O}(Mn/r_i)$ vertices and is computable in time $\widetilde{O}(|\mathcal{P}| \cdot (r_i/M))$. As before, for $i \leqslant 2$, this is $\widetilde{O}(n^2r_1^2/M)$; and $\widetilde{O}(Mn^2)$ otherwise. We get a running time of $\widetilde{O}(M^2n^2 + n^2r_1^2)$.

It is left to prove that B_i indeed hits at least one replacement path for all $s,t \in V$ and $e \in E$ that satisfy $d(s,t,e) \in [\frac{r_i}{2} - M, r_i]$. Let $P(s,t,e)$ be such a path and define u to be the first vertex on $P(s,t,e)$ (starting from s) such that $d(s,u,e) \geqslant (2/9)r_i - M$. If $i \geqslant 3$, then $r_{i-2} = (4/9)r_i$, whence $d(s,u,e) \in [\frac{r_{i-2}}{2} - M, \frac{r_{i-2}}{2})$ and the induction hypothesis implies that there is *some* replacement path P' from s to u avoiding the edge e such that B_{i-2} contains one vertex of P'. Otherwise, if $i \leqslant 2$, the same fact simply follows from $B_{i-2} = V$.

The path P' is not necessarily equal to the prefix of $P(s,t,e)[s..u]$ (but they have the same length $d(s,u,e)$). Replacing $P(s,t,e)[s..u]$ with P' gives a new replacement path that now has a pivot $x \in B_{i-2}$ on its prefix. Slightly abusing notation, we use $P(s,t,e)$ to denote also the updated path. Let v be the last vertex on $P(s,t,e)$ with $d(v,t,e) \geqslant (2/9)r_i - M$. By the same argument, we can assume that the suffix of $P(s,t,e)[v..t]$ contains a pivot $y \in B_{i-2}$. In the remainder, we show that there is some replacement path $P(x,y,e)$ that is hit by a vertex in B_i. If so, replacing the middle part $P(s,t,e)[x..y]$ with $P(x,y,e)$ finally proves the existence of a replacement path from s to t avoiding e and containing a vertex of B_i.

By the choice of the pivots x, y and the assumption $d(s,t,e) \in [\frac{r_i}{2} - M, r_i]$, the replacement distance $d(x,y,e)$ satisfies

$$r_i \geqslant d(s,t,e) \geqslant d(x,y,e) \geqslant d(s,t,e) - d(s,u,e) - d(v,t,e) \geqslant d(s,t,e) - 2\left(\frac{2r_i}{9} - M\right) \geqslant \frac{r_i}{18}.$$

First, assume that the shortest path $P(x,y)$ in the base graph G does not contain the edge e. Then, $P(x,y)$ can serve as the replacement path. It has length $d(x,y) = d(x,y,e)$ between $r_i/18$ and r_i, and we added it to $\mathcal{P}$. Otherwise, it holds that $e \in P(x,y)$. Observe that $d(x,y) \leqslant d(x,y,e) \leqslant r_i$ remains true. Therefore, we have queried the r_i-truncated DSO with the triple (x,y,e). Due to $d(x,y,e) \geqslant r_i/18$, we received a replacement path $P(x,y,e)$, which we added to $\mathcal{P}$. In both cases, some replacement path is hit by B_i, as desired. ◀

At first glance, it looks like the quadratic dependence on M is too high to be used in the derandomization. However, recall that we can assume $M = \widetilde{O}(n^{(3-\omega)/2})$. Over the $O(\log n)$ rounds with $i \geqslant 3$ and with access to the appropriately truncated DSOs, we can compute the sets $B_3, B_4, \dots$ in total time $\widetilde{O}(M^2n^2) = \widetilde{O}(Mn^{(7-\omega)/2}) = \widetilde{O}(Mn^{2.5})$ even if $\omega = 2$.

The next lemma is the last tool we need to construct the deterministic DSO.

▶ **Lemma 18.** *Let G be a graph with integer edge weights in the range $[1, M]$, $r \geqslant 1$ a real number, and $B \subseteq V$ a set of $\widetilde{O}(Mn/r)$ vertices such that for every pair of $s, t \in V$ and $e \in E$ with $r/2 - M \leqslant d(s,t,e) \leqslant r$, there exists a replacement path $P(s,t,e)$ that contains a vertex of B. Given an r-truncated DSO for G with $O(1)$ query time and the set B, one can, without further preprocessing, construct an $(3/2)r$-truncated DSO with query time $\widetilde{O}(Mn/r)$. Moreover, if the r-truncated DSO is path-reporting, so is the $(3/2)r$-truncated one.*

Proof. For any query (s,t,e), let $D(s,t,e)$ denote the returned value by the r-truncated DSO. If $D(s,t,e) \neq +\infty$, we also take this as the answer of the $(3/2)r$-truncated DSO. Otherwise, define $\ell = \min_{z \in B}\{D(s,z,e) + D(z,t,e)\}$. If $\ell \leqslant (3/2)r$, we return ℓ, and $+\infty$ else. Path queries are handled in the same fashion. In the case of $D(s,t,e) \neq +\infty$, we pass on the path $P(s,t,e)$ returned by the r-truncated DSO. If $\ell \leqslant (3/2)r$, we return the concatenation of $P(s,z,e)$ and $P(z,t,e)$ for some pivot $z \in B$ that attains the minimum ℓ. The query time is $O(|B|) = \widetilde{O}(Mn/r)$ for the distance, after which the path can be returned in $O(1)$ per edge.

It is clear that the query algorithm is correct whenever $d(s,t,e) \leqslant r$ as those queries are entirely handled by the given truncated DSO. Moreover, even if $d(s,t,e) > r$, then ℓ is an upper bound for $d(s,t,e)$ because all sums $D(s,z,e) + D(z,t,e)$ correspond to some path from s to t avoiding e, but not necessarily a shortest path.

Let $P = P(s,t,e)$ be a replacement path of length between r and $(3/2)r$, u the first vertex on P (seen from s) with $d(u,t,e) \leqslant r$, and v the last vertex on P with $d(s,v,e) \leqslant r$. Note that v lies between u and t on the path, whence $d(u,v,e) \leqslant r$. We further have

$$d(u,v,e) \geqslant d(s,t,e) - d(s,u,e) - d(v,t,e) = d(u,t,e) + d(s,v,e) - d(s,t,e) \geqslant 2r - \frac{3}{2}r = \frac{r}{2}.$$

By the properties of B, there exists some replacement path from u to v avoiding e that contains a pivot $z \in B$. With the usual argument of swapping parts of the path, we can assume z lies on the middle section of $P(s,t,e)$ between u and v. By construction, we have $\max\{d(s,z,e), d(z,t,e)\} \leqslant r$ so both distances (and corresponding paths) are correctly determined by the r-truncated DSO. In summary, we get $\ell \leqslant d(s,z,e) + d(z,t,e) = d(s,t,e)$ and the returned value ℓ is indeed the correct replacement distance. ◀

We are left to prove the final running time of the construction. Let $r = n^\alpha$ be the cut-off point for the distances at which we start the iterative growing. We build the Core DSO using the $O(r^2)$ subgraphs, compute unique shortest paths in G, followed by $O(\log n)$ iterations of Extend and Fast invocations, including the computation of the B_i. First, suppose the graph G is undirected. The total time is then

$$\begin{aligned}
&\widetilde{O}(Mn^\omega r^2) + \widetilde{O}(M^{1/2} n^{(\omega+3)/2}) + \widetilde{O}(Mn^{2.5} + n^2 r^2) + \widetilde{O}(n^2) \cdot \sum_{i=1}^{O(\log n)} \widetilde{O}\left(\frac{Mn}{(3/2)^i r}\right) \\
&= \widetilde{O}\left(M^{1/2} n^{(\omega+3)/2} + Mn^{2.5} + Mn^\omega r^2 + \frac{Mn^3}{r}\right) \\
&= \widetilde{O}\left(M^{1/2} n^{(\omega+3)/2} + Mn^{\max\{2.5,\ \omega+2\alpha,\ 3-\alpha\}}\right).
\end{aligned}$$

This is minimum for $\alpha = 1 - (\omega/3)$, where we get a running time of $\widetilde{O}(Mn^{(\omega+6)/3})$.

For directed graphs, determining the best α is a bit more involved. Recall that $O(n^{\omega(1-\alpha)})$ is the time needed to multiply a $n \times n^{1-\alpha}$ matrix with an $n^{1-\alpha} \times n$ matrix. Computing the Core oracle takes time $\widetilde{O}(Mn^{\omega(1-\alpha)}r^3) = \widetilde{O}(Mn^{\omega(1-\alpha)+3\alpha})$. With a similar calculation as above, we obtain a total preprocessing time of $\widetilde{O}\big(M^{1/2}n^{(\omega+3)/2} + Mn^{\max\{2.5,\ \omega(1-\alpha)+3\alpha,\ 3-\alpha\}}\big)$. This is minimized if α solves the equation $\omega(1-\alpha) = 3-4\alpha$. Le Gall and Urrutia [19] gave the current-best estimates for the values of the function ω. This shows that $1-\alpha$ is between 0.8 and 0.85, and we have $\omega(0.8) \leqslant 2.222256$ as well as $\omega(0.85) \leqslant 2.258317$. We exploit the fact that ω is convex [30], giving

$$\omega(1-\alpha) \leqslant \frac{(\alpha-0.15)\omega(0.8) + (0.2-\alpha)\omega(0.85)}{0.85-0.8} \leqslant 2.3665 - 0.72122\alpha$$

Equating the latter with $3-4\alpha$ yields the estimate $\alpha \leqslant 0.193212$, which in turn implies a preprocessing time of $\widetilde{O}(Mn^{2.806788})$.

References

1 Josh Alman and Virginia Vassilevska Williams. A Refined Laser Method and Faster Matrix Multiplication. In *Proceedings of the 32nd Symposium on Discrete Algorithms (SODA)*, pages 522–539, 2021. doi:10.1137/1.9781611976465.32.

2 Noga Alon, Shiri Chechik, and Sarel Cohen. Deterministic Combinatorial Replacement Paths and Distance Sensitivity Oracles. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 12:1–12:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.12.

3 Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault-tolerant subgraph for single-source reachability: General and optimal. *SIAM Journal on Computing*, 47:80–95, 2018. doi:10.1137/16M1087643.

4 Aaron Bernstein and David R. Karger. A Nearly Optimal Oracle for Avoiding Failed Vertices and Edges. In *Proceedings of the 41st Symposium on Theory of Computing (STOC)*, pages 101–110, 2009. doi:10.1145/1536414.1536431.

5 Davide Bilò, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. Near-Optimal Deterministic Single-Source Distance Sensitivity Oracles. In *Proceedings of the 29th European Symposium on Algorithms (ESA)*, pages 18:1–18:17, 2021. doi:10.4230/LIPIcs.ESA.2021.18.

6 Davide Bilò, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. Space-Efficient Fault-Tolerant Diameter Oracles. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 18:1–18:16, 2021. doi:10.4230/LIPIcs.MFCS.2021.18.

7 Greg Bodwin. Linear Size Distance Preservers. In *Proceedings of the 28th Symposium on Discrete Algorithms (SODA)*, pages 600–615, 2017. URL: http://dl.acm.org/citation.cfm?id=3039686.3039725.

8 Jan van den Brand and Thatchaphol Saranurak. Sensitive Distance and Reachability Oracles for Large Batch Updates. In *Proceedings of the 60th Symposium on Foundations of Computer Science (FOCS)*, pages 424–435, 2019. doi:10.1109/FOCS.2019.00034.

9 Shiri Chechik and Sarel Cohen. Near Optimal Algorithms for the Single Source Replacement Paths Problem. In *Proceedings of the 30th Symposium on Discrete Algorithms (SODA)*, pages 2090–2109, 2019. doi:10.1137/1.9781611975482.126.

10 Shiri Chechik and Sarel Cohen. Distance Sensitivity Oracles with Subcubic Preprocessing Time and Fast Query Time. In *Proccedings of the 52nd Symposium on Theory of Computing (STOC)*, pages 1375–1388, 2020. doi:10.1145/3357713.3384253.

11 Shiri Chechik, Sarel Cohen, Amos Fiat, and Haim Kaplan. $(1+\epsilon)$-Approximate f-Sensitive Distance Oracles. In *Proceedings of the 28th Symposium on Discrete Algorithms (SODA)*, pages 1479–1496, 2017. doi:10.1137/1.9781611974782.96.

12 Shiri Chechik and Ofer Magen. Near Optimal Algorithm for the Directed Single Source Replacement Paths Problem. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 81:1–81:17, 2020. doi:10.4230/LIPIcs.ICALP.2020.81.

13 Camil Demetrescu, Mikkel Thorup, Rezaul A. Chowdhury, and Vijaya Ramachandran. Oracles for Distances Avoiding a Failed Node or Link. *SIAM Journal on Computing*, 37:1299–1318, 2008. doi:10.1137/S0097539705429847.

14 Ran Duan, Yong Gu, and Hanlin Ren. Approximate Distance Oracles Subject to Multiple Vertex Failures. In *Proceedings of the 32nd Symposium on Discrete Algorithms (SODA)*, pages 2497–2516, 2021. doi:10.1137/1.9781611976465.148.

15 Ran Duan and Seth Pettie. Dual-Failure Distance and Connectivity Oracles. In *Proceedings of the 20th Symposium on Discrete Algorithms (SODA)*, pages 506–515, 2009. URL: https://dl.acm.org/citation.cfm?id=1496770.1496826.

16 Ran Duan and Seth Pettie. Fast Algorithms for (max,min)-Matrix Multiplication and Bottleneck Shortest Paths. In *Proceedings of the 20th Symposium on Discrete Algorithms (SODA)*, pages 384–391, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496813.

17 Ran Duan and Hanlin Ren. Maintaining Exact Distances under Multiple Edge Failures. In *Proceedings of the 54th Symposium on Theory of Computing (STOC)*, 2022. To appear.

18 Ran Duan and Tianyi Zhang. Improved Distance Sensitivity Oracles via Tree Partitioning. In *Proceedings of the 15th International Symposium on Algorithms and Data Structures (WADS)*, pages 349–360, 2017. doi:10.1007/978-3-319-62127-2_30.

19 François Le Gall and Florent Urrutia. Improved Rectangular Matrix Multiplication using Powers of the Coppersmith-Winograd Tensor. In *Proceedings of the 29th Symposium on Discrete Algorithms (SODA)*, pages 1029–1046, 2018. doi:10.1137/1.9781611975031.67.

20 François Le Gall. Faster Algorithms for Rectangular Matrix Multiplication. In *Proceedings of the 53rd Symposium on Foundations of Computer Science (FOCS)*, pages 514–523, 2012. doi:10.1109/FOCS.2012.80.

21 Fabrizio Grandoni and Virginia Vassilevska Williams. Improved Distance Sensitivity Oracles via Fast Single-Source Replacement Paths. In *Proceedings of the 53rd Symposium on Foundations of Computer Science (FOCS)*, pages 748–757, 2012. doi:10.1109/FOCS.2012.17.

22 Fabrizio Grandoni and Virginia Vassilevska Williams. Faster Replacement Paths and Distance Sensitivity Oracles. *ACM Transaction on Algorithms*, 16:15:1–15:25, 2020. doi:10.1145/3365835.

23 Yong Gu and Hanlin Ren. 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)*, pages 76:1–76:20, 2021. doi:10.4230/LIPIcs.ICALP.2021.76.

24 Manoj Gupta, Rahul Jain, and Nitiksha Modi. Multiple Source Replacement Path Problem. In *Proceedings of the 39th Symposium on Principles of Distributed Computing (PODC)*, pages 339–348, 2020. doi:10.1145/3382734.3405714.

25 Torben Hagerup, Peter Bro Miltersen, and Rasmus Pagh. Deterministic Dictionaries. *Journal of Algorithms*, 41:69–85, 2001. doi:10.1006/jagm.2001.1171.

26 Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. Conditional Hardness for Sensitivity Problems. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 26:1–26:31, 2017. doi:10.4230/LIPIcs.ITCS.2017.26.

27 Giuseppe F. Italiano, Luigi Laura, and Federico Santaroni. Finding Strong Bridges and Strong Articulation Points in Linear Time. *Theoretical Computer Science*, 447:74–84, 2012. doi:10.1016/j.tcs.2011.11.011.

28 Karthik C.S. and Merav Parter. Deterministic Replacement Path Covering. In *Proceedings of the 32nd Symposium on Discrete Algorithms (SODA)*, pages 704–723, 2021. doi:10.1137/1.9781611976465.44.

29 Valerie King. Fully Dynamic Algorithms for Maintaining All-Pairs Shortest Paths and Transitive Closure in Digraphs. In *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS)*, pages 81–91, 1999. doi:10.1109/SFFCS.1999.814580.

30 Grazia Lotti and Francesco Romani. On the Asymptotic Complexity of Rectangular Matrix Multiplication. *Theoretical Computer Science*, 23:171–185, 1983. doi:10.1016/0304-3975(83)90054-3.

31 Hanlin Ren. Improved Distance Sensitivity Oracles with Subcubic Preprocessing Time. In *Proceedings of the 28th European Symposium on Algorithms (ESA)*, pages 79:1–79:13, 2020. doi:10.4230/LIPIcs.ESA.2020.79.

32 Hanlin Ren. Improved Distance Sensitivity Oracles with Subcubic Preprocessing Time. *Journal of Computer and System Sciences*, 123:159–170, 2022. Journal version of [31]. doi:10.1016/j.jcss.2021.08.005.

33 Liam Roditty and Uri Zwick. Replacement Paths and k Simple Shortest Paths in Unweighted Directed Graphs. *ACM Transaction on Algorithms*, 8:33:1–33:11, 2012. doi:10.1145/2344422.2344423.

34 Avi Shoshan and Uri Zwick. All Pairs Shortest Paths in Undirected Graphs with Integer Weights. In *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS)*, pages 605–615, 1999. doi:10.1109/SFFCS.1999.814635.

35 Oren Weimann and Raphael Yuster. Replacement Paths and Distance Sensitivity Oracles via Fast Matrix Multiplication. *ACM Transactions on Algorithms*, 9:14:1–14:13, 2013. doi:10.1145/2438645.2438646.

36 Uri Zwick. All Pairs Shortest Paths Using Bridging Sets and Rectangular Matrix Multiplication. *Journal of the ACM*, 49:289–317, 2002. doi:10.1145/567112.567114.