# Near-Optimal Deterministic Single-Source Distance Sensitivity Oracles

**Davide Bilò**
Department of Humanities and Social Sciences, University of Sassari, Italy

**Sarel Cohen**
Hasso Plattner Institute, Universität Potsdam, Germany

**Tobias Friedrich**
Hasso Plattner Institute, Universität Potsdam, Germany

**Martin Schirneck**
Hasso Plattner Institute, Universität Potsdam, Germany

## Abstract

Given a graph with a distinguished source vertex $s$, the Single Source Replacement Paths (SSRP) problem is to compute and output, for any target vertex $t$ and edge $e$, the length $d(s,t,e)$ of a shortest path from $s$ to $t$ that avoids a failing edge $e$. A Single-Source Distance Sensitivity Oracle (Single-Source DSO) is a compact data structure that answers queries of the form $(t,e)$ by returning the distance $d(s,t,e)$. We show how to deterministically compress the output of the SSRP problem on $n$-vertex, $m$-edge graphs with integer edge weights in the range $[1,M]$ into a Single-Source DSO that has size $O(M^{1/2}n^{3/2})$ and query time $\widetilde{O}(1)$. We prove that the space requirement is optimal (up to the word size). Our techniques can also handle vertex failures within the same bounds.

Chechik and Cohen [SODA 2019] presented a combinatorial, randomized $\widetilde{O}(m\sqrt{n}+n^2)$ time SSRP algorithm for undirected and unweighted graphs. We derandomize their algorithm with the same asymptotic running time and apply our compression to obtain a deterministic Single-Source DSO with $\widetilde{O}(m\sqrt{n}+n^2)$ preprocessing time, $O(n^{3/2})$ space, and $\widetilde{O}(1)$ query time. Our combinatorial Single-Source DSO has near-optimal space, preprocessing and query time for unweighted graphs, improving the preprocessing time by a $\sqrt{n}$-factor compared to previous results with $o(n^2)$ space.

Grandoni and Vassilevska Williams [FOCS 2012, TALG 2020] gave an algebraic, randomized $\widetilde{O}(Mn^\omega)$ time SSRP algorithm for (undirected and directed) graphs with integer edge weights in the range $[1,M]$, where $\omega<2.373$ is the matrix multiplication exponent. We derandomize it for undirected graphs and apply our compression to obtain an algebraic Single-Source DSO with $\widetilde{O}(Mn^\omega)$ preprocessing time, $O(M^{1/2}n^{3/2})$ space, and $\widetilde{O}(1)$ query time. This improves the preprocessing time of algebraic Single-Source DSOs by polynomial factors compared to previous $o(n^2)$-space oracles.

We also present further improvements of our Single-Source DSOs. We show that the query time can be reduced to a constant at the cost of increasing the size of the oracle to $O(M^{1/3}n^{5/3})$ and that all our oracles can be made path-reporting. On sparse graphs with $m=O\big(\frac{n^{5/4-\varepsilon}}{M^{7/4}}\big)$ edges, for any constant $\varepsilon>0$, we reduce the preprocessing to randomized $\widetilde{O}(M^{7/8}m^{1/2}n^{11/8})=O(n^{2-\varepsilon/2})$ time. To the best of our knowledge, this is the first truly subquadratic time algorithm for building Single-Source DSOs on sparse graphs.

**2012 ACM Subject Classification** Theory of computation → Shortest paths; Theory of computation → Data structures design and analysis; Theory of computation → Cell probe models and lower bounds; Theory of computation → Pseudorandomness and derandomization

**Keywords and phrases** derandomization, distance sensitivity oracle, single-source replacement paths, space lower bound

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2021.18

**Related Version** *Full Version*: https://arxiv.org/abs/2106.15731

**Funding** *Davide Bilò*: This work was partially supported by the Research Grant FBS2016_BILO, funded by "Fondazione di Sardegna" in 2016.





# 1 Introduction

One of the basic problems in computer science is the computation of shortest paths and distances in graphs that are subject to a small number of transient failures. We study two central problems of this research area on undirected graphs $G$ with $n$ vertices and $m$ edges, namely, the *Single-Source Replacement Paths* (SSRP) problem and *Single-Source Distance Sensitivity Oracles* (Single-Source DSOs).

**The SSRP Problem.** In the SSRP problem, we are given a graph $G$ with a fixed source vertex $s$ and are asked to compute, for every vertex $t$ and edge $e$, the replacement distance $d(s,t,e)$, which is the length of the shortest $s$-$t$-path in the graph $G-e$, obtained by dropping the edge $e$. By first computing any shortest path tree for $G$ rooted at $s$, one can see that there are only $O(n^2)$ relevant distances $d(s,t,e)$, namely, those for which $e$ is in the tree.

Chechik and Cohen [10] presented an $\widetilde{O}(m\sqrt{n}+n^2)$ time[1] combinatorial[2] SSRP algorithm for unweighted graphs. They also showed that the running time cannot be improved by polynomial factors, assuming that any combinatorial algorithm for Boolean Matrix Multiplication (BMM) on $n \times n$ matrices containing $m$ 1's requires $mn^{1-o(1)}$ time. Gupta et al. [21] simplified the SSRP algorithm and generalized it to multiple sources. For a set of $\sigma$ sources, they presented a combinatorial algorithm that takes $\widetilde{O}(m\sqrt{n\sigma}+\sigma n^2)$ time. Grandoni and Vassilevska Williams [17, 18] gave an algorithm for both directed and undirected graphs with integer edge weights in the range $[1,M]$ that uses fast matrix multiplications and runs in $\widetilde{O}(Mn^\omega)$ time, where $\omega < 2.37286$ is the matrix multiplication exponent [2, 25, 35]. We are only concerned with *positive* integer weights, but it is worth noting that SSRP with weights in $[-M,M]$ is strictly harder, modulo a breakthrough in Min-Plus Product computation, with a current best running time of $O(M^{0.8043}n^{2.4957})$ as shown by Gu et al. [20].

All the SSRP algorithms above are randomized, it is an interesting open problem whether they can be derandomized in the same asymptotic running time.

**Single-Source DSOs.** A Distance Sensitivity Oracle (DSO) is a data structure that answers queries $(u,v,e)$, for vertices $u,v$ and edge $e$, by returning the replacement distance $d(u,v,e)$, A Single-Source DSO, with fixed source $s$, answers queries $(t,e)$ with $d(s,t,e)$.

Of course, any SSRP algorithm gives a Single-Source DSO by just tabulating the whole output in $O(n^2)$ space, the replacement distances can then be queried in constant time. However, the space usage is far from optimal. Parter and Peleg [29] developed a deterministic algorithm that computes an $O(n^{3/2})$ size subgraph of $G$ containing a breadth-first-search tree of $G-e$ for every failing edge $e$. The subgraph can also be thought of as a Single-Source DSO with $O(n^{3/2})$ space and query time. Bilò et al. [7] presented a Single-Source DSO of the same size with $\widetilde{O}(\sqrt{n})$ query time and $\widetilde{O}(mn)$ preprocessing time. Gupta and Singh [22] later designed a randomized Single-Source DSO of $\widetilde{O}(n^{3/2})$ size, $\widetilde{O}(mn)$ preprocessing time,[3] but with a better $\widetilde{O}(1)$ query time. The results in the latter two works generalize to the case of $\sigma$ sources in such a way that the time and size scale by $o(\sigma)$ factors.

For the case of $\sigma = n$ sources, that is, general (all-pairs) DSOs, Bernstein and Karger [5, 6] designed an oracle taking $\widetilde{O}(n^2)$ space with constant query time, even for directed graphs with real edge weights. The space was subsequently improved to $O(n^2)$ by Duan and Zhang [16],

[1] For a non-negative function $f = f(n)$, we use $\widetilde{O}(f)$ to denote $O(f \cdot \mathsf{polylog}(n))$.

[2] The term "combinatorial algorithm" is not well-defined, and is often interpreted as not using any matrix multiplication. Arguably, combinatorial algorithms can be considered efficient in practice as the constants hidden in the matrix multiplication bounds are rather high.

[3] The authors of [22] do not report the preprocessing time, but it can be reconstructed as $\widetilde{O}(mn)$.

which is optimal [34]. The combinatorial $\widetilde{O}(mn)$ time preprocessing for building the DSOs is conditionally near-optimal as it matches the best known bound (up to polylogarithmic factors) for the simpler problem of finding the All-Pairs Shortest Paths (APSP). The conditional lower bound in [10], stating that there exists no combinatorial algorithm solving the undirected SSRP problem with real edge weights in $O(mn^{1-\varepsilon})$ time for any positive $\varepsilon > 0$, unless there is a combinatorial algorithm for the APSP problem in $O(mn^{1-\varepsilon})$ time, also implies that there exists no *Single-Source* DSO with $\widetilde{O}(1)$ query time and $O(mn^{1-\varepsilon})$ preprocessing time for real edge weights. Therefore, the DSOs in [6, 16], are also conditionally near-optimal for the single source case with real edge weights.

Several algebraic all-pairs DSOs with subcubic preprocessing time have been developed in the last decade for graphs with integer edge weights in $[1, M]$ [9, 11, 17, 30, 37]. Very recently, Gu and Ren [19] presented a randomized DSO achieving a $O(Mn^{2.5794})$ preprocessing time with $O(1)$ query time, improving upon the one by Ren [30, 31] with an $\widetilde{O}(Mn^{2.6865})$ preprocessing time. Those DSOs can also be used in the single-source case, but the requirement to store the information for all pairs forces them to take $\Omega(n^2)$ space [34]. The algebraic SSRP algorithm in [18], seen as a data structure, has a better preprocessing time than any known (general) DSO but also takes $O(n^2)$ space, which we have seen to be wasteful.

We are not aware of an algebraic Single-Source DSO that simultaneously achieves $o(n^2)$ space and has a better preprocessing time than their all-pairs counterparts. It is interesting whether we can construct space-efficient oracles faster when focusing on a single source.

Additional results on replacement paths and DSOs (for single or multiple failures and directed graphs) can be found in [3, 9, 12, 13, 14, 15, 18, 23, 24, 26, 27, 28, 32, 36]. The most efficient Single-Source DSOs in their respective settings are shown in Table 1 below.

## 1.1 Our Contribution

We research SSRP algorithms, Single-Source DSO data structures, and the connection between the two. Our first contribution is presented in Section 5. We derandomize the near-optimal combinatorial SSRP algorithm of Chechik and Cohen [10] for undirected, unweighted graphs and the algebraic algorithm of Grandoni and Vassilevska Williams [17, 18] for undirected graphs with integer weights in the range $[1, M]$. (The second algorithm can be found in the full version.) Both deterministic algorithms have the same asymptotic runtime as their randomized counterparts.

▶ **Theorem 1.** *There is a deterministic, combinatorial SSRP algorithm for undirected, unweighted graphs running in time $\widetilde{O}(m\sqrt{n} + n^2)$ and a deterministic, algebraic SSRP algorithm for undirected graphs with integer weights in the range $[1, M]$ running in $\widetilde{O}(Mn^\omega)$.*

We present in Section 3 a deterministic reduction from the problem of building a Single-Source DSO to SSRP on undirected graphs with small integer edge weights.

▶ **Theorem 2.** *Let $G$ be an undirected graph with integer edge weights in the range $[1, M]$ and let $s$ be the source vertex. Suppose we are given access to a shortest path tree $T_s$ of $G$ rooted in $s$ and all values $d(s, t, e)$ for vertices $t$ of $G$ and edges $e$ in $T_s$. There is a deterministic, combinatorial algorithm that in time $O(n^2)$ builds a Single-Source DSO of size $O(M^{1/2}n^{3/2})$ with $\widetilde{O}(1)$ query time. The same statement holds for vertex failures if instead we are given access to the values $d(s, t, v)$ for all vertices $t$ and $v$ of $G$*

The algorithm does not require access to the graph $G$ itself. As there can be up to $O(n^2)$ relevant distances $d(s, t, e)$, the running time is linear in the input. If the algorithm additionally has access to $G$ and is given $O(m\sqrt{Mn} + n^2)$ time, the Single-Source DSO also reports the replacement paths $P(s, t, e)$ in time $\widetilde{O}(1)$ per edge. The query time of the oracle can be improved to $O(1)$ at the cost of increasing the size of the oracle to $O(M^{1/3}n^{5/3})$.

Plugging the deterministic SSRP algorithms of Theorem 1 into our reduction of Theorem 2, gives the following Single-Source DSOs as corollaries.

▶ **Theorem 3.** *There is a deterministic, combinatorial Single-Source DSO for undirected, unweighted graphs taking $O(n^{3/2})$ space, with $\widetilde{O}(m\sqrt{n}+n^2)$ preprocessing time, and $\widetilde{O}(1)$ query time. There is a deterministic, algebraic Single-Source DSO for undirected graphs with integer weights in the range $[1,M]$ taking $O(M^{1/2}n^{3/2})$ space, with $\widetilde{O}(Mn^{\omega})$ preprocessing time, and $\widetilde{O}(1)$ query time.*

When comparing the results with other Single-Source DSO with $o(n^2)$ space, the preprocessing time of our combinatorial solution is better by a factor of $\sqrt{n}$ compared to previous oracles [7, 22]. The preprocessing time of the algebraic part of Theorem 3 improves (ignoring polylogarithmic factors) by a factor of $n^{2.5794-\omega} > n^{0.2}$ over the current best algebraic (all-pairs) DSO [19]. See Table 1 for more details. In fact, we combine the efficient preprocessing of SSRP algorithms (seen as DSOs) with a compression scheme that achieves nearly-optimal space. To the best of our knowledge, Theorem 3 presents the first algebraic Single-Source DSO with $o(n^2)$ space that achieves a better performance than any all-pairs DSO. It is also the first space-efficient Single-Source DSO for graphs with small integer weights.

We further study lower bounds for Single-Source DSOs. Note that given an oracle whose preprocessing time is $P$ and query time is $Q$, one can solve the SSRP problem in time $P+n^2\cdot Q$ by building the DSO and running the queries $(t,e)$ for every $t\in V, e\in E(T_s)$. Therefore, if $n^2\cdot Q = O(P)$, the $mn^{1/2-o(1)}+\Omega(n^2)$ conditional[4] time-lower bound for the SSRP problem [10], obtained by a reduction from BMM, implies the same lower bound for $P$. The preprocessing of our combinatorial oracle in Theorem 3 is thus nearly optimal. We further investigate how much a Single-Source DSO can be compressed. In contrast to [10], we obtain an *unconditional space*-lower bound using an argument from information theory.

▶ **Theorem 4.** *Any Single-Source DSO must take $\Omega(\min\{M^{1/2}n^{3/2}, n^2\})$ bits of space on at least one $O(n)$-vertex graph with integer edge weights in the range $[1,M]$.*

A small gap remains between Theorems 2 and 4 as the space is bounded at $\Omega(M^{1/2}n^{3/2})$ bits, while the oracle takes this many machine words. Nevertheless, it shows that on dense graphs our Single-Source DSOs in Theorem 3 have near-optimal space.

The Single-Source DSOs presented above all have $\Omega(n^2)$ preprocessing time, which cannot be avoided for graphs with $m=\Omega(n^{3/2})$, assuming the BMM conjecture. SSRP algorithms require $\Omega(n^2)$ time simply to output the solution. It is not clear whether this lower bound also applies to Single-Source DSO on sparse graphs. We partially answer this question negatively by developing a truly subquadratic, randomized Single-Source DSO in Section 6. We use new algorithmic techniques and structural properties of independent interest.

▶ **Theorem 5.** *There is a randomized Single-Source DSO taking $O(M^{1/2}n^{3/2})$ space that has $\widetilde{O}(1)$ query time w.h.p.[5] The oracle also reports a replacement path in $\widetilde{O}(1)$ time per edge w.h.p. On graphs with $m=O(M^{3/4}n^{7/4})$ edges, the preprocessing time is $\widetilde{O}(M^{7/8}m^{1/2}n^{11/8})$. If the graph is sparse, meaning $m=O(n^{5/4-\varepsilon}/M^{7/4})$ for any $\varepsilon>0$, this is $\widetilde{O}(n^{2-\varepsilon/2})$.*

## 1.2 Comparison with Previous Work

Table 1 shows a comparison of the most efficient Distance Sensitivity Oracles in their respective setting, as well as the results presented in this work. We distinguish four dimensions of different problem types.

[4] The $\Omega(n^2)$ term is unconditional and stems from the size of the output, see [10].
[5] An event occurs *with high probability* (w.h.p.) if it has probability at least $1-n^{-c}$ for some $c>0$.

**Table 1** Comparison of results. †The preprocessing time is for graphs with $m = O(M^{3/4}n^{7/4})$.

| Preprocessing time | Space | Query time | Setting | Reference |
|---|---|---|---|---|
| $\widetilde{O}(mn)$ | $O(n^2)$ | $O(1)$ | D C W Ap | [6, 16] |
| $\widetilde{O}(Mn^{2.5794})$ | $\widetilde{O}(n^2)$ | $O(1)$ | R A I Ap | [19] |
| $\widetilde{O}(mn^{1/2}+n^2)$ | $O(n^2)$ | $O(1)$ | R C U Ss | [10] |
| $\widetilde{O}(Mn^{\omega})$ | $O(n^2)$ | $O(1)$ | R A I Ss | [18] |
| $\widetilde{O}(mn)$ | $\widetilde{O}(n^{3/2})$ | $\widetilde{O}(n^{1/2})$ | D C U Ss | [7] |
| $\widetilde{O}(mn)$ | $\widetilde{O}(n^{3/2})$ | $\widetilde{O}(1)$ | R C U Ss | [22] |
| $\widetilde{O}(mn^{1/2}+n^2)$ | $O(n^{5/3})$ | $O(1)$ | D C U Ss | Lemma 10 |
| $\widetilde{O}(mn^{1/2}+n^2)$ | $O(n^{3/2})$ | $\widetilde{O}(1)$ | D C U Ss | Theorem 3 |
| $\widetilde{O}(Mn^{\omega})$ | $O(M^{1/3}n^{5/3})$ | $O(1)$ | D A I Ss | Lemma 10 |
| $\widetilde{O}(Mn^{\omega})$ | $O(M^{1/2}n^{3/2})$ | $\widetilde{O}(1)$ | D A I Ss | Theorem 3 |
| $\widetilde{O}(M^{7/8}m^{1/2}n^{11/8})^{\dagger}$ | $O(M^{1/2}n^{3/2})$ | $\widetilde{O}(1)$ | R C I Ss | Theorem 5 |

1. Randomized (R) vs. deterministic (D),
2. Combinatorial (C) vs. algebraic (A),
3. Unweighted (U) vs. real weights (W) vs. integer weights in $[1, M]$ (I),
4. All-Pairs (Ap) vs. single-source (Ss).

Our deterministic, combinatorial Single-Source DSO from Theorem 3 has near-optimal space, preprocessing and query time for dense graphs. It improves the preprocessing time of the randomized DSOs by Bernstein and Karger [6], Bilò et al. [7], and Gupta and Singh [22] by a factor of $O(\sqrt{n})$. When viewing the randomized SSRP algorithm of Chechik and Cohen as an oracle, our solution has the same preprocessing time but reduces the space requirement, by an near-optimal factor of $O(n^{1/3})$ while increasing the query time to only $\widetilde{O}(1)$.

Our algebraic combinatorial Single-Source DSO from Theorem 3 has near-optimal space and query time for dense graphs, its preprocessing time improves over the randomized, algebraic DSOs of Chechick and Cohen [11], Ren [30, 31], as well as Gu and Ren [19] by a factor of $\widetilde{O}(n^{2.5794-\omega})$. It has the same preprocessing time as the SSRP algorithm by Grandoni and Vassilevska Williams [18], but compresses the output to $O(M^{1/2}n^{3/2})$ space.

Our Single-Source DSO from Lemma 10 even achieves constant query time at the expense of larger $O(n^{5/3})$ (respectively, $O(M^{1/3}n^{5/3})$) space. All of our oracles can handle vertex failures and are path-reporting, the query time then corresponds to the time needed per edge of the replacement path. In Theorem 5, we also obtain Single-Source DSO with subquadratic preprocessing time for sparse graphs.

## 1.3 Techniques

**Multi-stage derandomization.** To derandomize the SSRP algorithms, we extend the techniques by Alon, Chechik, and Cohen [3] to identify a small set of critical paths we need to hit. In [3], a single set of paths was sufficient, we extend this to a hierarchical multi-stage framework. The set of paths in each stage depends on the hitting set found in the previous ones. For example, a replacement path from $s$ to $t$ avoiding the edge $e$ decomposes into two shortest paths $P(s, q)$ and $P(q, t)$ in the original graph for some unknown vertex $q$, see [1]. It is straightforward to hit all of the components $P(s, q)$. We then use this hitting set in a more involved way to find sets of vertices that also intersect all of the subpaths $P(q, t)$.

**Versatile compression.** The key observation of our reduction to SSRP is that any shortest $s$-$t$-path can be partitioned into $O(\sqrt{Mn})$ segments such that all edges in a segment have the same replacement distance. Gupta and Singh [22] proved this for unweighted graphs. However, it is not obvious how to generalize their approach to the weighted case. We give a simpler proof in the presence of small integer weights, which immediately transfers also to vertex failures. We further show how to extend this to multiple targets and even reuse it to obtain the subquadratic algorithm on sparse graphs. In [22], a randomized oracle was presented that internally uses the rather complicated data structures of Demetrescu et al. [14]. We instead give a deterministic construction implementable with only a few arrays. Unfortunately, the compression scheme crucially depends on the graph being undirected.

**Advanced search for replacement paths.** The randomized algorithm building the DSO in subquadratic time for sparse graphs needs to find the $O(\sqrt{Mn})$ segments partitioning the $s$-$t$-path. Naively, this takes $O(n)$ time per target vertex $t$ as we need to explore the whole path for potential segment endpoints and do not know the corresponding replacement paths in advance. We use standard random sampling to hit all such replacements paths with only a few vertices and exploit the path's monotonicity properties to develop more advanced search techniques. This reduces the time needed per target to $O(n^{1-\varepsilon})$, after some preprocessing. The analysis uses the fact that entire subpaths can be discarded without exploration.

**Open problems.** Our compression scheme and the randomized, subquadratic Single-Source DSO on sparse graphs can also handle vertex failures rather than only edge failures. It remains an open question whether one can obtain efficient deterministic SSRP algorithms in the vertex-failure scenario. If an analog of Theorem 1 held for vertex failures, then Theorem 2 would directly transfer the extension also to the DSOs of Theorem 3. Another interesting open question is whether there is a Single-Source DSO with deterministic, truly subquadratic time preprocessing on graphs with $m = O(n^{3/2-\varepsilon})$ edges. Can one obtain better Single-Source DSOs, and prove matching lower bounds, for sparse graphs?

## 2 Preliminaries

We let $G = (V, E, w)$ denote the undirected, edge-weighted base graph on $n$ vertices and $m$ edges, and tacitly assume $m \geqslant n$. The weights $w(e)$, $e \in E$, are integers in $[1, M]$ with $M = \mathsf{poly}(n)$. For an undirected, weighted graph $H$, we denote by $V(H)$ the set of its vertices, and by $E(H)$ edge set of its edges. We write $e \in H$ for $e \in E(H)$ and $v \in H$ for $v \in V(H)$. Let $P$ be a simple path in $H$. The *length* or *weight* $w(P)$ of $P$ is $\sum_{e \in E(P)} w(e)$. For $u, v \in V(H)$, we denote by $P_H(u, v)$ a shortest path (one of minimum weight) from $u$ to $v$. If a particular shortest path is intended, this will be made clear from the context. The *distance* of $u$ and $v$ is $d_H(u, v) = w(P_H(u, v))$. We drop the subscript when talking about the base graph $G$. The restriction on the maximum weight $M$ allows us to store any graph distance in a single machine word on $O(\log n)$ bits. Unless explicitly stated otherwise, we measure space complexity in the number of words.

Let $x, y \in V(P)$ be two vertices on the simple path $P$. We denote by $P[x..y]$ the subpath of $P$ from $x$ to $y$. Let $P_1 = (u_1, \ldots, u_i)$ and $P_2 = (v_1, \ldots, v_j)$ be two paths in $H$. Their *concatenation* is $P_1 \circ P_2 = (u_1, \ldots, u_i, v_1, \ldots, v_j)$, provided that $u_i = v_1$ or $\{u_i, v_1\} \in E(H)$.

Fix some *source vertex* $s \in V$ in the base graph $G$. For any *target vertex* $t \in V$ and edge $e \in E$, we let $P(s, t, e)$ denote a *replacement path* for $e$, that is, a shortest path from $s$ to $t$ in $G$ that does not use the edge $e$. Its weight $d(s, t, e) = w(P(s, t, e))$ is the *replacement*

*distance*. Given a specific shortest path $P(s,t)$ in $G$ and a replacement path $P(s,t,e)$, we can assume w.l.o.g. that the latter is composed of the *common prefix* that it shares with $P(s,t)$, the *detour* part which is edge-disjoint from $P(s,t)$, and the *common suffix* after $P(s,t,e)$ remerges with $P(s,t)$. All statements apply to vertex failures as well.

## 3 Using SSRP to Build Single-Source DSOs

In this section, we prove Theorem 2. We describe how to deterministically reduce the task of building a Single-Source DSO to computing the replacement distances in the SSRP problem. Recall that we assume we are given a shortest path tree $T_s$ of the base graph $G$ rooted in the source $s$. This does not loose generality as we could as well compute it in time $O(m)$ via Thorup's algorithm [33]. However, the tree $T_s$ focuses our attention to the $O(n^2)$ *relevant* replacement distances in $G$. The failure of an edge $e$ can only increase the distance from $s$ to some target $t$ if $e$ lies on the $s$-$t$-path $P(s,t)$ in $T_s$. Given a query $(e,t)$, we can thus check whether $e$ is relevant for $t$ in $O(1)$ time using a lowest common ancestor (LCA) data structure of size $O(n)$ [4]. If the maximum weight $M$ is larger than $n$, we are done as we store the relevant replacement distances, original graph distances, and the LCA data structure.

However, for $M \leqslant n$, there are more space-efficient solutions. Using time $O(n^2)$, that is, linear in the number of relevant distances, we compress the space needed to store them down to $O(M^{1/2}n^{3/2})$ while increasing the query time only to $\widetilde{O}(1)$. This scheme also allows several extensions, namely, handling vertex failures, reporting fault-tolerant shortest path trees, or retaining constant query time by using slightly more space. We first give an overview of the reduction. Suppose we have a set of *pivots* $D \subseteq V$ such that any $s$-$t$-path $P(s,t)$ in $T_s$ has at least one element of $D$ among its last $\sqrt{n}$ vertices. For a target $t$, let $x$ be the pivot on $P(s,t)$ that is closest to $t$. We distinguish three cases depending on the failing edge $e$.

- **Near case.** The edge $e$ belongs to the near case if it is on the subpath $P(s,t)[x..t]$ from the last pivot to the target. We construct a data structure to quickly identify those edges It is then enough to store the associated replacement distances explicitly.
- **Far case I.** The edge $e$ belongs to the far case I if it is on the subpath $P(s,t)[s..x]$ and there is a replacement path for $e$ that uses the vertex $x$. We handle this by storing a linear number of distances for every pivot in $D$.
- **Far case II.** We are left with edges $e$ on $P(s,t)[s..x]$ for which no replacement path uses $x$. We show that there are only $O(M^{1/2}n^{3/2})$ many different replacement distances of this kind. We can find the correct distance in $\widetilde{O}(1)$. This is the only case with a quadratic running time, space requirements depending on $M$, and a super-constant query time. We also show how to avoid the latter at the expense of a higher space complexity.

**Near case.** We first describe how to obtain the set $D$. We also take $D$ to denote a representing data structure. That is, for all $t \in V$, $D[t]$ shall denote the last pivot on the path $P(s,t)$ in $T_s$. A deterministic greedy algorithm efficiently computes a small sets $D$.

▶ **Lemma 6.** *There exists a set $D \subseteq V$ with $|D| \leqslant \sqrt{n}$, computable in time $\widetilde{O}(n)$, such that every $s$-$t$-path in $T_s$ contains a pivot in $D$ among its last $\sqrt{n}$ vertices. In the same time, we can compute a data structure taking $O(n)$ space that returns $D[t]$ in constant time.*

Let $x = D[t]$ be the pivot assigned to $t$. An edge $e$ belongs to the *near case* with respect to $t$ if it lies on $P(x,t) = P(s,t)[x..t]$. Observe that $P(x,t)$ has less than $\sqrt{n}$ edges. We store $d(s,t,e)$ for the near case in an array with $(t,e)$ as key. With access to the distances, the array can be computed in $O(n^{3/2})$ total time and space. Consider a query $(t,e)$ such that $e$ has already been determined above to be on the path $P(s,t)$. The edge $e = \{u,v\}$ thus belongs to the near case iff $D[u] = D[v] = x$. If so, we look up $d(s,t,e)$ in the array.

**Far case I.** We say a query $(t, e)$ belongs to the *far case* if $e$ is on the subpath $P(s, x) = P(s, t)[s..x]$. These are the queries not yet handled by the process above. Note that $d(s, t, e) \leqslant d(s, x, e) + d(x, t)$ holds for all queries in the far case. If a replacement path $P(s, x, e)$ exists, $P(s, x, e) \circ P(s, t)[x..t]$ is some $s$-$t$-path that avoids $e$ whose length is the right-hand side; otherwise, we have $d(s, x, e) = \infty$ and $d(s, t, e) \leqslant d(s, x, e) + d(x, t)$ holds vacuously. We split the far case depending on the existence of certain replacement paths. Recall that we can assume that any replacement path consists of a common pre- and suffix with the original path $P(s, t)$ and a detour that is edge-disjoint from $P(s, t)$. We let $(t, e)$ belong to the *far case I* if $e$ is on $P(s, x)$ and there is a replacement path $P(s, t, e)$ that uses the vertex $x$. It is readily checked that for a query in the far case this holds iff $d(s, t, e) = d(s, x, e) + d(x, t)$. Otherwise, that is, if no replacement path $P(s, t, e)$ uses $x$ or, equivalently, $d(s, t, e) < d(s, x, e) + d(x, t)$, the query is said to be in the *far case II*.

It takes too much space to store the replacement distances for all edges in the far case, or memorize which edge falls in which subcase. Instead, we build two small data structures and, at query time, compute two (potentially different) distances. We show that always the smaller one is correct, which we return as the final answer. First, for every pivot $x \in D$ and edge $e \in P(s, x)$, we store the replacement distance $d(s, x, e)$. Since $|D| \leqslant \sqrt{n}$ and $|E(P(s, x))| \leqslant n$, we can do so in $O(n^{3/2})$ time and space. Given a query $(t, e)$ in the far case, we access the storage corresponding to $D[t] = x$, retrieve $d(s, x, e)$, and add $d(x, t) = d(s, t) - d(s, x)$. This gives the first candidate distance. It may overestimate $d(s, t, e)$, namely, if $e$ belongs to the far case II.

**Far case II.** This case is more involved than the previous. We make extensive use of what we call break points. Let $e_1, \ldots, e_k$ be the edges of $P(s, t)[s..x]$ in the far case II (w.r.t. $t$) in increasing distance from $s$. We then have $d(s, t, e_1) \geqslant \ldots \geqslant d(s, t, e_k)$. This is due to the fact that any replacement path $P(s, t, e_i)$ avoids the whole subpath starting with $e_i$ and ending in $x$. Its length is thus at least the replacement distance for any $e_j$, $j \geqslant i$. Let $u_i$ be vertex of $e_i$ that is closer to $s$. We say $u_i$ is a *break point* if $d(s, t, e_i) > d(s, t, e_{i+1})$. A break point is the beginning of a segment in which the edges in the far case II have equal replacement distance. We show that there are only $O(\sqrt{Mn})$ break points/replacement distances.

For the analysis, we let the edges choose a *representative* replacement path. They do so one after another in the above order. Edge $e_i$ first checks whether one of its replacement paths has previously been selected by an earlier edge $e_h$, $h < i$. If so, it takes the same one; otherwise, it chooses a possible replacement path arbitrarily. Let $\mathcal{R}$ denote the set of representatives and let $R \in \mathcal{R}$. We define $z_R$ to be the first vertex on the detour part of $R$. The vertices $z_R$, $R \in \mathcal{R}$, are also important for the subquadratic algorithm in Section 6.

▶ **Lemma 7.** *Edges $e_i$ and $e_j$ that belong to the far case II choose the same representative iff $d(s, t, e_i) = d(s, t, e_j)$. All representatives have different lengths and $|\mathcal{R}|$ equals the number of break points. Let $R, R' \in \mathcal{R}$ be such that $R'$ is the next shorter representative after $R$. We have $d(s, z_R) < d(s, z_{R'})$ and all edges represented by $R$ lie on the subpath $P(s, t)[z_R..z_{R'}]$. There is exactly one break point on $P(s, t)[z_R..z_{R'}]$, the one corresponding to length $w(R)$.*

**Proof.** Edges with different replacement distances have disjoint sets of replacement paths to choose from. Now suppose the replacement distances $d(s, t, e_i) = d(s, t, e_j)$ are equal. Without loosing generality, the edge $e_j$, $j \geqslant i$, is further away from $s$ and selects its representative after $e_i$. The representative replacement path $R$ for edge $e_i$ also avoids $e_j$ since it does not remerge with $P(s, t)$ prior to pivot $x$. As the distances $d(s, t, e_j) = d(s, t, e_i) = w(R)$ are the same, $R$ is in fact a replacement path for $e_j$ and is selected again as representative. The assertions of the different lengths and the total number of representatives easily follow.

Let $R \in \mathcal{R}$ be a representative replacement path. The first vertex $z_R$ of its detour part must be closer to $s$ than all edges it represents as $R$ avoids them. Let $e^*$ be the edge closest to $s$ that belongs to the far case II and is represented by $R$. The break point $u^*$ starting the segment with replacement distance $w(R)$ is thus the vertex of $e^*$ that is closer to $s$.

Let now $R' \in \mathcal{R}$ be the next shorter representative after $R$. If we had $d(s, z_{R'}) \geqslant d(s, z_R)$, then $R'$ would be a path that avoids $e^*$ and has length $w(R') < w(R) = d(s, t, e^*)$ strictly smaller than the replacement distance, a contradiction. Reusing the same arguments as before, we also get that the break point corresponding to $w(R')$ lies after $z_{R'}$ and that the break point $u^* \in e^*$ cannot lie below $z_{R'}$ (on subpath $P(s,t)[z_{R'}..t]$). In summary, the subpath $P(s,t)[z_R..z_{R'}]$ contains exactly one break point, namely, $u^*$. ◀

It is left to prove that $|\mathcal{R}| = O(\sqrt{Mn})$. The following lemma is the heart of our compression scheme. It simplifies and thereby generalizes a result by Gupta and Singh [22] for unweighted undirected graphs. The argument we use is versatile enough to not only cover integer-weighted graphs, it extends to vertex failures as well (Lemma 9). A similar idea also allows us to design an oracle with constant query time (Lemma 10) and the subquadratic preprocessing algorithm on sparse graphs (Theorem 5). Unfortunately, the argument crucially depends on the graph being undirected. New techniques are needed to compress the fault-tolerant distance information in directed graphs.

▶ **Lemma 8.** *The number of representatives for edges on $P(s,t)$ is $|\mathcal{R}| \leqslant 3\sqrt{Mn}$.*

**Proof.** All representatives are of different length by Lemma 7. Also, they have length at least $d(s,t)$, the weight of the original $s$-$t$-path $P$. Hence, there are only $2\sqrt{Mn}$ many of length at most $d(s,t) + 2\sqrt{Mn}$. We now bound the number of *long* representatives, which are strictly longer than that. Let $R$ be a long representative. Its detour part is longer than $2\sqrt{Mn}$, whence it must span at least $2\sqrt{n/M}$ vertices. Consider the path on the first $\sqrt{n/M}$ vertices of the detour starting in $z_R$, we call it $\text{Stub}_R$. If $\text{Stub}_R$ does not intersect with $\text{Stub}_{R'}$ for any other long $R' \in \mathcal{R}$, $R' \neq R$, there can only be $n/(\sqrt{n/M}) = \sqrt{Mn}$ stubs in total and thus as many long representatives.

To reach a contradiction, assume the stubs of $R$ and $R'$ intersect. Let $e$ be an edge represented by $R$ and $y \in V(\text{Stub}_R) \cap V(\text{Stub}_{R'})$ a vertex on both stubs. W.l.o.g. $R'$ is strictly shorter than $R$ and thus $z_{R'}$ comes behind $z_R$ on the path $P$ and $e$ is on $P[z_R..z_{R'}]$ (Lemma 7). Note that $w(\text{Stub}_R), w(\text{Stub}_{R'}) \leqslant \sqrt{Mn}$. Therefore, the path $P^* = P[s..z_R] \circ R[z_R..y] \circ R'[y..z_{R'}] \circ P[z_{R'}..t]$ avoids $e$ and has length $w(P^*) \leqslant d(s,t) + w(R[z_R..y]) + w(R'[y..z_{R'}]) \leqslant d(s,t) + 2\sqrt{Mn} < w(R)$. This is a contradiction to $R$ being the representative of $e$. ◀

Observe how the argument in the proof above depends on the fact that we can traverse the segment $R'[z_{R'}..y] \subseteq \text{Stub}_{R'}$ in both directions. When following $R'$ from $s$ to $t$, we visit $z_{R'}$ prior to $y$, while for $P^*$ it is the other way around. This is not necessarily true in a directed graph. Indeed, one can construct examples that have a directed path on $\Omega(n)$ edges in which each of them has its own replacement distance.

With access to the replacement distances, all break points can be revealed by a linear scan of the path $P(s,t)$ in time $O(n)$. Let $z_{i_1}, \ldots, z_{i_{|\mathcal{R}|}}$ be the break points ordered by increasing distance to the source $s$ and $e_{i_1}, \ldots, e_{i_{|\mathcal{R}|}}$ the corresponding edges. For the data structure, we compute an ordered array of the original distances $d(s, z_{i_1}) < \cdots < d(s, z_{i_{|\mathcal{R}|}})$ associated with the replacement distances $d(s, t, e_{i_j})$, taking $O(\sqrt{Mn})$ space. Let $(e, t)$ be a query with $e = \{u, v\}$. We compute the index $j = \arg\max_{1 \leqslant k \leqslant |\mathcal{R}|} \{d(s, z_{i_k}) \leqslant d(s, u)\}$, with a binary search on the array in $O(\log n)$ time and retrieve $d(s, t, e_{i_j})$ as the second candidate distance.

The edge $e$ lies on the subpath $P(s,t)[z_{i_j}..z_{i_{j+1}}]$ (respectively, on $P(s,t)[z_{i_{|\mathcal{R}|}}..x]$ if $j = |\mathcal{R}|$). It thus has replacement distance *at most* $d(s,t,e_{i_j})$. If $e$ belongs to the far case II, the second candidate distance is exact and (strictly) smaller than the first one $d(s,x,e) + d(x,t)$; otherwise, the first candidate is smaller (or equal) and correct.

Scaling this solution to all targets $t \in V$ gives a total space requirement of $O(M^{1/2}n^{3/2})$. However, the preprocessing time is $O(n^2)$, dominated by the linear scans for each target.

## 3.1 Extensions

There are several possible extensions for our Single-Source DSO. While the transfer to vertex failures comes for free, reducing the query time to a constant, making the oracle path-reporting, or returning the whole fault-tolerant shortest path tree incurs additional costs of a higher space requirement or preprocessing time, respectively. We still assume the setting of Theorem 2, i.e., oracle access to the replacement distances for failing edges/vertices.

**Vertex failures.** The solutions for the near case, and far case I hold verbatim also for vertex failures. A vertex on the path $P(s,t)[s..x]$, except $s$ itself, belongs to the *far case II* iff it satisfies $d(s,t,v) < d(s,t,x) + d(x,t)$. Let $\mathcal{R}_V$ be the sets of representatives, now chosen by the *vertices*. The advantage of the proof of Lemma 8 is that it easily transfers to vertex failures. While the stubs of the detours may no longer be unique, they now intersect at most one other stub and identify *pairs* of representatives.

▶ **Lemma 9.** *The number of representatives for vertices on $P(s,t)$ is $|\mathcal{R}_V| \leqslant 5\sqrt{Mn}$.*

**Constant query time.** If we could query the break point of an edge in the far case II in $O(1)$ time, our Single-Source DSO had a constant overall query time. However, since the break points also depend on the target $t$, hard-coding them would yield a $O(n^2)$ space solution, which is wasteful for $M = o(n)$. Instead, we improve the analysis in Lemma 8. It hardly made any use of the fact that the pivot $x$ is among the last $\sqrt{n}$ vertices on the $s$-$t$-path in $T_s$ and considered only a single target. We now strike a balance between selecting more pivots and grouping targets with the same assigned pivot together.

▶ **Lemma 10.** *There is an algorithm that, when given oracle access to the replacement distances for failing edges (vertices), preprocesses in $O(n^2)$ time a Single-Source DSO for edge (vertex) failures taking $O(\min\{M^{1/3}n^{5/3}, n^2\})$ space and having constant query time.*

**Path-reporting oracles.** We can adapt our Single-Source DSOs to also report the replacement paths using the same space. However, to do so it is not enough to have access to the replacement distances as the paths depend on the structure of $G$. Also, making the oracle path-reporting increases in preprocessing time, which now also depends on $m$.

▶ **Lemma 11.** *With access to $G$, there is a path-reporting Single-Source DSO for edge (vertex) failures with $O(\min\{m\sqrt{Mn}, mn\}+n^2)$ preprocessing time and either $O(\min\{M^{1/2}n^{3/2}, n^2\})$ space and $\widetilde{O}(1)$ query time per edge, or $O(\min\{M^{1/3}n^{5/3}, n^2\})$ space and $O(1)$ query time.*

**Fault-tolerant shortest path tree oracles.** We are going one step further in the direction of fault-tolerant subgraphs, see for example [8, 29]. We enable our oracle to report, for any failing edge or vertex, the whole fault-tolerant single-source shortest path tree. Compared to the path-reporting version, we make sure to return every tree edge only once.

▶ **Lemma 12.** *With access to $G$, there is a data structure with $O(\min\{m\sqrt{Mn}, mn\} + n^2)$ preprocessing time, taking $O(\min\{M^{1/2}n^{3/2}, n^2\})$ space that, upon query $e \in E$ (respectively, $v \in V$), returns a shortest path tree for $G - e$ (respectively, $G - v$) rooted in $s$ in time $O(n)$.*

## 4 Space Lower Bound

We now present an information-theoretic lower bound showing that the space of the Single-Source DSO resulting from our reduction is optimal up to the word size.

▶ **Theorem 4.** *Any Single-Source DSO must take $\Omega(\min\{M^{1/2}n^{3/2}, n^2\})$ bits of space on at least one $O(n)$-vertex graph with integer edge weights in the range $[1, M]$.*

**Proof.** Let $M' = \min\{M, n\}$. We give an incompressibility argument in that we show that one can store any binary $n \times n$ matrix $X$ across $\sqrt{n/M'}$ Single-Source DSOs. Not all of them can use only $o(\sqrt{M'}\,n^{3/2})$ bits of space as otherwise this would compress $X$ to $o(n^2)$ bits. We create graphs $G_1, G_2, \ldots, G_{\sqrt{n/M'}}$. Each of them has $O(n)$ vertices and maximum edge weight $M$. The graph $G_k$ will be used to store the $\sqrt{M'n}$ rows of $X$ with indices from $(k-1)\sqrt{M'n}+1$ to $k\sqrt{M'n}$.

We first describe the parts that are common to all of the $G_k$. Let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ be two sets of $n$ vertices each, we connect $a_i$ and $b_j$ by an edge of weight 1 iff $X[i, j] = 1$. There are no other edges between $A$ and $B$. We also add a path $P = (v_1, \ldots, v_{\sqrt{M'n}})$ all of whose edges have weight 1. The vertex $s = v_{\sqrt{M'n}}$ is the source in each graph. Also, let $\{v_0, v_1\}$ be an edge of weight $M$, it serves to raise the maximum edge weight to $M$, if needed. Specifically in $G_k$ and for each $1 \leqslant i \leqslant \sqrt{M'n}$, we connect the vertex $v_i$ with $a_{(k-1)\sqrt{M'n}+i}$ by a path $P_{k,i}$ of total weight $2i - 1$. Due to the edge weights, we can make the path $P_{k,i}$ so that it uses at most $2i/M'$ edges and thus so many new vertices. In total, $G_k$ has at most $2n + (\sqrt{M'n} + 1) + \sum_{i=1}^{\sqrt{M'n}} \frac{2i}{M'} = O(n)$ vertices due to $M' \leqslant n$.

Let $e_i$ denote the edge $\{v_{i-1}, v_i\}$ on $P$. We claim that $X[(k-1)\sqrt{M'n} + i, j] = 1$ if and only if the replacement distance in $G_k$ is $d_{G_k}(v_{\sqrt{M'n}}, b_j, e_i) = \sqrt{M'n} + i$. We assume $k = 1$, larger $k$ follow in the same fashion. Observe that one has to go through a vertex in $A' = \{a_i, a_{i+1}, \ldots, a_{\sqrt{M'n}}\}$ to reach $b_j$ from the source $s = v_{\sqrt{M'n}}$. Conversely, $A'$ is the only part of $A$ that is reachable from $s$ in $G_1 - e_i$ without using any vertex of $B$.

If there is no replacement path from $s$ to $b_j$ avoiding $e_i$, we have $d_{G_1}(s, b_j, e_i) = \infty$ and $X[i', j] = 0$ for all $i \leqslant i' \leqslant \sqrt{M'n}$, as desired. Let thus $P(s, b_j, e_i)$ be a replacement path and further $a_{i^*}$ its first vertex that is in $A$ (the one closest to the source $s$). Therefore, $i^* \geqslant i$ and $P(s, b_j, e_i)$ has the form $(v_{\sqrt{M'n}}, \ldots, v_{i^*}) \circ P_{1,i^*} \circ P'$ for some $a_{i^*}$-$b_j$-path $P'$. It holds that $d_{G_1}(v_{\sqrt{M'n}}, b_j, e_i) = (\sqrt{M'n} - i^*) + (2i^* - 1) + w(P') = \sqrt{M'n} + i^* - 1 + w(P') \geqslant \sqrt{M'n} + i$. Equality holds only if $i^* = i$ and $w(P) = 1$, thus $a_i$ must be a neighbor of $b_j$ and $X[i, j] = 1$ follows; otherwise, the replacement distance is strictly larger. ◀

## 5 Derandomizing Single-Source Replacement Paths Algorithms

In this section, we derandomize the combinatorial $\widetilde{O}(m\sqrt{n} + n^2)$ time algorithm for SSRP of Chechik and Cohen [10] obtaining the same asymptotic running time. In the full version, we also derandomize the algebraic SSRP algorithm of Grandoni and Vassilevska Williams. When combined with the reduction of Section 3, they give deterministic Single-Source DSOs.

Suppose the base graph $G = (V, E)$ is unweigted. It follows from a result by Afek et al. [1, Theorem 1] that for every target $t \in V$, edge $e \in E$, and replacement path $P(s, t, e)$ in $G - e$, there exists a vertex $q$ on $P(s, t, e)$ such that both subpaths $P(s, t, e)[s..q]$ and $P(s, t, e)[q..t]$

are shortest paths in the original graph $G$. Computing the vertex $q$ directly for each pair $(t, e)$ is too expensive. Instead, the algorithm in [10] employs a random hitting set for the subpaths. The only randomization used in [10] is to sample every vertex independently with probability $O((\log n)/\sqrt{n})$ to create a set $B \subseteq V$ of so-called *pivots*. The set $B$ contains $\widetilde{O}(\sqrt{n})$ such pivots w.h.p. The correctness of the algorithm relies on the following important property. With high probability, there exists a vertex $x \in B \cup \{s\}$ before $q$ on $P(s, t, e)$ and a vertex $y \in B \cup \{t\}$ after $q$ such that the subpath of $P(s, t, e)[x..y]$ has length only $\widetilde{O}(\sqrt{n})$. Here, we describe how to compute the set $B$ deterministically with the same properties. We defer the proof of correctness of the algorithm to the full version.

We derandomize the vertex selection using an approach similar to the one of Alon, Chechik, and Cohen [3]. Given paths $D_1, \ldots, D_k$, where each contains at least $L$ vertices, the folklore greedy algorithm constructs a hitting set of size $\widetilde{O}(n/L)$, by iteratively covering the maximum number of unhit paths, in $\widetilde{O}(kL)$ time. The challenge is to quickly compute a suitable set of paths. We construct three systems of path $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ to obtain $B$.

We prepare some notation. For a rooted tree $T$, a vertex $v \in V(T)$, and an integer parameter $L \geqslant 0$, let $\text{Last}_{T,L}(v)$ be the subpath containing the last $L$ edges of the path in the tree $T$ from the root to $v$, or the whole path if it has length less than $L$. Let $|\text{Last}_{T_s,L}(v)|$ denote the number of edges on the path.

- **Paths $\mathcal{L}_1$ and hitting set $B_1$.** Set $\mathcal{L}_1$ contains the last $\sqrt{n}/2$ edges of every path in $T_s$, $\mathcal{L}_1 = \{\text{Last}_{T_s,\sqrt{n}/2}(v) \mid v \in V,\ |\text{Last}_{T_s,\sqrt{n}/2}(v)| = \sqrt{n}/2\}$. As an alternative, we can also use Lemma 6 to compute in $\widetilde{O}(n)$ time a deterministic hitting set $B_1$ for $\mathcal{L}_1$ of size $2\sqrt{n}$.
- **Paths $\mathcal{L}_2$ and hitting set $B_2$.** We run a breadth-first search from every vertex $x \in B_1$ to compute the shortest paths trees $T_x$ rooted in $x$, and define the second set to be $\mathcal{L}_2 = \{\text{Last}_{T_x,\sqrt{n}/2}(y) \mid x \in B_1, y \in V,\ |\text{Last}_{T_x,\sqrt{n}/2}(y)| = \sqrt{n}/2\}$ Greedy selection computes a hitting set $B_2$ for $\mathcal{L}_2$ of size $\widetilde{O}(\sqrt{n})$ in total time $\widetilde{O}(n^2)$.

Before we can define $\mathcal{L}_3$, we need additional notation. Let $e = \{u, v\}$ be an edge in $T_s$ such that $u$ is closer to $s$ than $v$ and let $T_{s,v}$ be the subtree of $T_s$ rooted in $v$. Let further $G_e = (V_e, E_e, w_e)$ be a weighted graph such that $V_e$ contains $s$ and the vertices $x \in V(T_{s,v})$ with $d(s, x) \leqslant d(s, v) + 4\sqrt{n}$. The edges of $G_e$ that are inside of $T_{s,v}$ are the same as in $G$, and additionally every shortest path $P(s, x)$ from $s$ to every vertex $x \in V_e$ such that $P(s, x)$ passes only through vertices outside of $V_e$ (except for its first vertex $s$ and its last vertex $x \in V_e$) is replaced with a shortcut edge $(s, x)$ whose weight is equal to the length $d(s, x)$ of the corresponding shortest path $P(s, x)$, preserving the original paths distances (using weights). The SSRP algorithm in [10] computes Dijkstra's algorithm from $s$ in each $G_e$. We let $T_{G_e}$ denote the resulting shortest path tree.

- **Paths $\mathcal{L}_3$ and hitting set $B_3$.** The third set $\mathcal{L}_3 := \{\text{Last}_{T_{G_e},\sqrt{n}/2}(x) \mid x \in V, e \in E(T_s),\ |\text{Last}_{T_{G_e},\sqrt{n}/2}(x)| = \sqrt{n}/2\}$ contains $O(n^{3/2})$ paths as every vertex $x \in V$ belongs to at most $4\sqrt{n}$ graphs $G_e$. We thus get a hitting set $B_3$ of size $\widetilde{O}(\sqrt{n})$ in time $\widetilde{O}(n^2)$.

The deterministic set $B = B_1 \cup B_2 \cup B_3$ can then be used as pivots in the SSRP algorithm.

## 6 Subquadratic Preprocessing on Sparse Graphs

Finally, we show how to obtain a Single-Source DSO with subquadratic preprocessing at least on sparse graphs. In order to prove Theorem 5, we present an algorithm running in time $\widetilde{O}(M^{7/8} m^{1/2} n^{11/8} + \frac{M^{1/8}\, m^{3/2}}{n^{3/8}})$. If $m = O(M^{3/4} n^{7/4})$, then the dominating term is $\widetilde{O}(M^{7/8} m^{1/2} n^{11/8})$. If the graph even satisfies $m = O(n^{5/4-\varepsilon}/M^{7/4})$ for any $\varepsilon > 0$, then

the preprocessing time is $\widetilde{O}(n^{2-\varepsilon/2})$. We explain the main part of the randomized algorithm that allows us to design the Single-Source DSO. The algorithm is easily adaptable to deal with vertex failures as well. The proofs and some of the technical details are deferred to the full version due to the lack of space. In the following, we assume that the graph is indeed sparse, that is, $m = O(n^{5/4-\varepsilon}/M^{7/4})$. The next sampling lemma is folklore, see e.g. [18, 32].

▶ **Lemma 13.** *Let $H$ be a graph with $n$ vertices, $c > 0$ a positive constant, and $L$ such that $L \geqslant c \ln n$. Define a random set $R \subseteq V$ by sampling each vertex to be in $R$ independently with probability $(c \ln n)/L$. Then, with probability at least $1 - \frac{1}{n^c}$, the size of $R$ is $\widetilde{O}(n/L)$. Let further $\mathcal{P}$ be a set of $\ell$ simple paths in $H$, each of which spans at least $L$ vertices. Then, with probability at least $1 - \frac{\ell}{n^c}$, we have $V(P) \cap R \neq \emptyset$ for every $P \in \mathcal{P}$.*

We employ random sampling to hit one shortest path on at least $L = \frac{n^{11/8}}{M^{1/8}\, m^{1/2}}$ edges for every pair of vertices. Any vertex is included in the set $R$ of *random pivots* independently with a probability of $(3 \ln n)/L$. We also include the source $s$ in $R$ to hit all short $s$-$t$-paths. By Lemma 13, we have $|R| = \widetilde{O}(n/L) = \widetilde{O}\big(\frac{M^{1/8}\, m^{1/2}}{n^{3/8}}\big)$ w.h.p. Randomization is used here since it takes too long to handle the $O(n^2)$ paths explicitly.

We additionally construct a set $D$ of (possibly different, regular) *pivots* that are used to classify replacement paths into near case, far case I, and far case II similar to Section 3. The set $D$ is computed deterministically using Lemma 6, where we select a pivot every $\sqrt{n}$ levels. For a target vertex $t \neq s$, the *proper pivot* of $t$ shall be that pivot $x \in D$ on the path $P(s,t)$ in $T_s$ that is closest to $t$ but satisfies $d(x,t) \geqslant 4ML$, or $x = s$ if there is no such pivot. We let $D_1[t]$ denote the proper pivot of $t$ and $D_2[t] = D_1[D_1[t]]$, provided that $D_1[t] \neq s$.

For every random pivot $\chi \in R$ and every edge $e$ on the path $P(s,\chi)$, we compute $d(s,\chi,e)$ in $\widetilde{O}(m)$ time per pivot using the algorithm of Malik, Mittal, and Gupta [26]. In the same time bound, we also get the vertex of $P(s,\chi)$ at which $P(s,\chi,e)$ diverges and we assume that $P(s,\chi,e)$ represents the path that diverges from $P(s,\chi)$ at a vertex that is as close as possible to $s$.[6] For each pivot $x \in D$ and every $e$ on $P(s,x)$, we also compute $d(s,x,e)$. This takes total time $\widetilde{O}(m(|D| + |R|)) = \widetilde{O}\big(mn^{1/2} + \frac{M^{1/8}\, m^{3/2}}{n^{3/8}}\big) = \widetilde{O}\big(\frac{m^{1/2} n^{9/8-\varepsilon/2}}{M^{7/8}} + \frac{M^{1/8}\, m^{3/2}}{n^{3/8}}\big)$ and allows us to answer replacement distance queries in $O(1)$ time if the target is in $D \cup R$.

We are left to handle non-pivot targets. Fix a $t \in V \setminus (D \cup R)$ and let $x_1 = D_1[t]$, and $x_2 = D_2[t]$. We use similar cases as before.

- **Near case.** The edge $e$ is on $P(s,t)[x_2..t] = P(x_2,t)$.
- **Far case I.** The edge $e$ is on $P(s,t)[s..x_2] = P(s,x_2)$ and there is a replacement path $P(s,t,e)$ that passes through $x_2$.
- **Far case II.** The edge $e$ is on $P(s,x_2)$ and there is no replacement path $P(s,t,e)$ that passes through $x_2$.

In the remainder, we show how to efficiently compute the replacement distances in the far case II as previously this was the only case with quadratic run time. The technical details of the near case are reported in full version. A shortest path tree of $G$ and the replacement distances to targets in $D$ are enough to handle the far case I, see Section 3.

Since in the far case II the pivot $x_2$ lies on $P(s,t)$, we can assume $P(s,t)$ to have length $d(s,t) \geqslant d(x_2,t) \geqslant 4ML$ and at least $4L$ edges. In the following, we use different indexing schemes pointing to objects and distances related to $P(s,t)$, all of them are ordered from the source $s$ to pivot $x_2$. First, we denote by $R_1, \ldots, R_k$ the $k$ representative replacement paths

[6] The replacement path $P(s,\chi,e)$ computed in [26] is obtained as the concatenation of a subpath $P(s,u)$ of $T_s$, an edge $\{u,v\}$ of $G - e$, and a subpath $P(v,\chi)$ in $T_\chi$ (the shortest paths tree of $G$ rooted at $\chi$).

for edges in the far case II. We have $k \leqslant 3\sqrt{Mn}$ by Lemma 8. Let the *distinguished* edge $e^*_\ell \in P(s,t)$ be the one that is closest to $s$ such that $R_\ell$ represents $e^*_\ell$, i.e., $R_\ell$ is a replacement path in $G - e^*_\ell$ and we fall in far case II. Set $d_\ell = w(R_\ell)$. As no replacement path from $s$ to $t$ for edge $e^*_\ell$ uses vertex $x_2$, we have $d_\ell < d(s, x_2, e^*_\ell) + d(x_2, t)$. The distinguished edges $e^*_1, \ldots, e^*_k$ are ordered by increasing distance from $s$, this implies $d_1 > \cdots > d_k$ for their replacement distances, see Section 3. Furthermore, let $N$ be the number of *all* edges (of the far cases I and II) on the path $P(s, x_2) = (e_1, e_2, \ldots, e_N)$, seen in order from $s$ to $x_2$. This way, we identify $P(s, x_2)$ with the interval $[1, N]$, an index $j \in [1, N]$ stands for the $j$-th edge $e_j$ on $P(s, x_2)$. With a slight abuse of notation, we also say that $e_j \in [a, b]$ in case $j \in [a, b]$.

We employ the random pivots to efficiently compute all the $k$ pairs $(d_\ell, e^*_\ell)$ w.h.p. The key idea is that, for each failing edge $e$ on $P(s, x_2)$, there exists w.h.p. a random pivot $\chi \in R$ such that $d(\chi, t) \leqslant ML$ and $d(s, t, e) = d(s, \chi, e) + d(\chi, t)$ simultaneously hold. To see this, recall that any replacement path $P(s, t, e)$ has at least $4L$ edges and let $y$ be the vertex such that $P(s, t, e)[y..t]$ consists of the last $L$ of them. We claim that $P(s, t, e)[y..t]$ is in fact a shortest path in $G$. Assume there were a shorter $y$-$t$-path, then it must contain $e$ and have length at least $d(x_2, t) \geqslant 4ML$, a contradiction. Therefore, *some* shortest $y$-$t$-path in $G$ has at least $L$ edges and is thus hit by a random pivot $\chi$ w.h.p., which gives the equality. Any reference to high probability refers to this fact. We use it to design a recursive algorithm that finds the pairs $(d_\ell, e^*_\ell)$ w.h.p. in time $O(|R|\, M^{3/4} n^{3/4}) = \widetilde{O}(M^{7/8} m^{1/2} n^{3/8})$ per target.

Recall that we view $P(s, x_2)$ as $[1, N]$. When exploring a subinterval $[a, b]$, the algorithm searches for a pair $(d_\ell, e^*_\ell)$ with a distinguished edge $e^*_\ell \in [a, b]$. The algorithm knows both an upper bound $\Delta_{[a,b]}$ and a lower bound $\delta_{[a,b]}$ on the admissible values for $d_\ell$. More precisely, $\Delta_{[a,b]} + 1$ corresponds w.h.p. to the smallest possible value $d_{\ell'}$ such that $e^*_{\ell'} \in [1, a{-}1]$ (the lower the index, the higher is $d_{\ell'}$); similarly, $\delta_{[a,b]} - 1$ is the the largest possible value $d_{\ell'}$ for $e^*_{\ell'} \in [b{+}1, N]$. In the beginning, we set $\Delta_{[1,N]} = \infty$, $\delta_{[1,N]} = 0$ and the algorithm explores the entire interval $[1, N]$. It terminates when there are no more unexplored subintervals.

We now describe the search for $d_\ell$ with $e^*_\ell \in [a, b]$. We assume $a \leqslant b$ and $\delta_{[a,b]} \leqslant \Delta_{[a,b]}$ as otherwise no such pair exists. Set $\mu = \max_{j \in [a,b]} \{d(s, x_2, e_j) + d(x_2, t)\}$. The algorithm keeps searching in the interval only if $\mu > \delta_{[a,b]}$. Indeed, if $\mu \leqslant \delta_{[a,b]}$, we know for sure that such a pair does not exist as there must be a replacement path (of type far case I) that passes through vertex $x_2$. We first compute the largest index $j \in [a, b]$ for which $\mu = d(s, x_2, e_j) + d(x_2, t)$. We do so by employing a range minimum query (RMQ) data structure to support such queries in constant time after an $O(N) = O(n)$ time preprocessing [4]. Observe that the same data structure can be reused for all the target vertices $t'$ for which $D_2[t'] = x_2$. It is enough that it stores the values $d(s, x_2, e)$, instead of $d(s, x_2, e) + d(x_2, t)$. The former distances are independent of the considered target and we already computed them above. We use only $O(|D|)$ RMQ data structures, which we prepare in $O(n|D|) = O(n^{3/2})$ time.

In the following, we assume $\mu > \delta_{[a,b]}$. We select a candidate replacement path for $e_j$ by choosing the shortest one that runs through a random pivot in $O(|R|)$ time via brute-force search in the data we computed above for the targets in $R$. Ties are broken in favor of the replacement path $P(s, \chi, e_j)$ that diverges from $P(s, x_2)$ at the vertex that is closest to $s$. Let $\delta = \min_{\chi \in R} \big\{d(s, \chi, e_j) + d(\chi, t)\big\}$ be the length of such a replacement path, w.h.p. it is the actual replacement distance $P(s, t, e_j)$. Let further $\chi_j$ be the minimizing random pivot, and $z_j$ the vertex of $P(s, x_2)$ at which $P(s, \chi_j, e_j)$ diverges. We check whether $\delta < \mu$ and $\delta \leqslant \Delta_{[a,b]}$ holds. If either of the two conditions is violated, then there is no need to keep searching in the interval $[a, j]$, as shown in the next lemma. In this case, the algorithm makes a recursive call on the *lower* interval $[j{+}1, b]$ (the one with smaller replacement distances) by setting $\Delta_{[j+1,b]} = \Delta_{[a,b]}$ and $\delta_{[j+1,b]} = \delta_{[a,b]}$. We say that the search was *unsuccessful*.

▶ **Lemma 14.** *If $\delta \geqslant \mu$ or $\delta > \Delta_{[a,b]}$, then, w.h.p. we have $e^*_\ell \notin [a,j]$ for all $\ell \in [k]$.*

Suppose the search is *successful*, that is, $\delta < \mu$ and $\delta \leqslant \Delta_{[a,b]}$. We then use binary search techniques[7] to compute in $O(\log n)$ time the smallest index $i \in [a,j]$ for which the edge $e_i$ lies on the subpath $P(s,x_2)[z_j..x_2]$ and $\delta < d(s,x_2,e_i) + d(x_2,t)$ holds. The case $i = j$ is possible. The condition on $e_i$ is such that $P(s,\chi_j,e_j)$ also avoids $e_i$, which implies $d(s,t,e_i) \leqslant \delta < d(s,x_2,e_i) + d(x_2,t)$. The edge $e_i$ must belong to the far case II w.r.t. target $t$. We show that in fact $(\delta, e_i)$ is w.h.p. the sought pair with $e^*_\ell \in [a,j]$ and minimum $d_\ell$.

▶ **Lemma 15.** *Let $\ell \in [k]$ be maximal such that $e^*_\ell \in [a,j]$. Then, w.h.p. $\delta = d_\ell$ and $e_i = e^*_\ell$.*

The algorithm outputs $(\delta, e_i)$ and recurses on the lower interval $[j{+}1,b]$ with new bounds $\Delta_{[j+1,b]} = \delta - 1$ and $\delta_{[j+1,b]} = \delta_{[a,b]}$, as well as on the *upper* interval $[a,i{-}1]$, with $\Delta_{[a,i-1]} = \Delta_{[a,b]}$ and $\delta_{[a,i-1]} = \delta + 1$. This is justified since the edges in $[i,j]$ that belong to the far case II are w.h.p. precisely the ones represented by the path $R_\ell$ of length $d_\ell = \delta$.

The time needed for one target $t$ is proportional (up to a log-factor) to the number of random pivots and the overall number of searches. There are $k = O(\sqrt{Mn})$ successful searches by Lemma 8. The following lemma bounds the number of unsuccessful searches.

▶ **Lemma 16.** *The number of unsuccessful searches for a single target vertex is $O(M^{3/4}n^{3/4})$.*

The algorithm computes w.h.p. all pairs for one target vertex in time $\widetilde{O}(|R|\,M^{3/4}n^{3/4}) = \widetilde{O}(M^{7/8}m^{1/2}n^{3/8})$, scaling this to all targets gives $\widetilde{O}(M^{7/8}m^{1/2}n^{11/8})$.

## References


1. Yehuda Afek, Anat Bremler-Barr, Haim Kaplan, Edith Cohen, and Michael Merritt. Restoration by Path Concatenation: Fast Recovery of MPLS Paths. *Distributed Computing*, 15:273–283, 2002. doi:10.1007/s00446-002-0080-6.
2. 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.
3. 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.
4. Michael A. Bender and Martin Farach-Colton. The LCA Problem Revisited. In *Proceedings of the 4th Latin American Symposium Theoretical Informatics (LATIN)*, pages 88–94, 2000. doi:10.1007/10719839_9.
5. Aaron Bernstein and David R. Karger. Improved Distance Sensitivity Oracles via Random Sampling. In *Proceedings of the 19th Symposium on Discrete Algorithms (SODA)*, pages 34–43, 2008. URL: https://dl.acm.org/citation.cfm?id=1347082.1347087.
6. 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.
7. Davide Bilò, Keerti Choudhary, Luciano Gualà, Stefano Leucci, Merav Parter, and Guido Proietti. Efficient Oracles and Routing Schemes for Replacement Paths. In *Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 13:1–13:15, 2018. doi:10.4230/LIPIcs.STACS.2018.13.


[7] Let interval $[a',b'] \subseteq [a,j]$ lie entirely below $z_j$. We divide it into subintervals $[a',j']$ and $[j'{+}1,b']$ of roughly equal sizes and check whether the maximum value returned by the RMQ data structure on query $[a',j']$ is still larger than $\delta$. If so, we recurse on the interval $[a',j']$; otherwise, on $[j'{+}1,b']$.

8 Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Fault-Tolerant Approximate Shortest-Path Trees. *Algorithmica*, 80:3437–3460, 2018. doi:10.1007/s00453-017-0396-z.

9 Jan van den Brand and Thatchaphol Saranurak. Sensitive Distance and Reachability Oracles for Large Batch Updates. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS, 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 424–435. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00034.

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

11 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.

12 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. $f$-Sensitivity Distance Oracles and Routing Schemes. *Algorithmica*, 63:861–882, 2012. doi:10.1007/s00453-011-9543-0.

13 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.

14 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.

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

17 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.

18 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.

19 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)*, 2021. To appear.

20 Yuzhou Gu, Adam Polak, Virginia Vassilevska Williams, and Yinzhan Xu. Faster Monotone Min-Plus Product, Range Mode, and Single Source Replacement Paths. In *Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2021. To appear.

21 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.

22 Manoj Gupta and Aditi Singh. Generic Single Edge Fault Tolerant Exact Distance Oracle. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 72:1–72:15, 2018. doi:10.4230/LIPIcs.ICALP.2018.72.

23 John Hershberger and Subhash Suri. Vickrey Prices and Shortest Paths: What is an edge worth? In *Proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS)*, pages 252–259, 2001. doi:10.1109/SFCS.2001.959899.

24 John Hershberger and Subhash Suri. Erratum to "Vickrey Pricing and Shortest Paths: What is an edge worth?". In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS)*, page 809, 2002. doi:10.1109/SFCS.2002.1182006.

**25** François Le Gall. Powers of Tensors and Fast Matrix Multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

**26** Kavindra Malik, A. K. Mittal, and Sumit K. Gupta. The $k$ Most Vital Arcs in the Shortest Path Problem. *Operations Research Letters*, 8:223–227, 1989. doi:10.1016/0167-6377(89)90065-5.

**27** Enrico Nardelli, Guido Proietti, and Peter Widmayer. A Faster Computation of the Most Vital Edge of a Shortest Path. *Information Processing Letters*, 79:81–85, 2001. doi:10.1016/S0020-0190(00)00175-7.

**28** Enrico Nardelli, Guido Proietti, and Peter Widmayer. Finding the Most Vital Node of a Shortest Path. *Theoretical Computer Science*, 296:167–177, 2003. doi:10.1016/S0304-3975(02)00438-3.

**29** Merav Parter and David Peleg. Sparse Fault-Tolerant BFS Structures. *ACM Transactions on Algorithms*, 13:11:1–11:24, 2016. doi:10.1145/2976741.

**30** 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.

**31** Hanlin Ren. Improved Distance Sensitivity Oracles with Subcubic Preprocessing Time. *CoRR*, abs/2007.11495, 2020. ArXiv preprint. Full version of [30]. arXiv:2007.11495.

**32** 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.

**33** Mikkel Thorup. Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time. *Journal of the ACM*, 46:362–394, 1999. doi:10.1145/316542.316548.

**34** Mikkel Thorup and Uri Zwick. Approximate Distance Oracles. *Journal of the ACM*, 52:1–24, 2005. doi:10.1145/1044731.1044732.

**35** Virginia Vassilevska Williams. Multiplying Matrices Faster Than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing (STOC)*, pages 887–898, 2012. doi:10.1145/2213977.2214056.

**36** Virginia Vassilevska Williams and R. Ryan Williams. Subcubic Equivalences Between Path, Matrix, and Triangle Problems. *Journal of the ACM*, 65:27:1–27:38, 2018. doi:10.1145/3186893.

**37** 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.