Undirected 3-Fault Replacement Path in Nearly Cubic Time

Shucheng Chi
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Ran Duan
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Benyu Wang
University of Michigan, Ann Arbor, MI, USA

Tianle Xie
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Abstract

Given a graph $G = (V, E)$ ($n = |V|$, $m = |E|$) and two vertices $s, t \in V$, the f-fault replacement path (fFRP) problem computes for every set F of at most f edges, the distance from s to t when edges in F fail. A recent result shows that 2FRP in directed graphs can be solved in $\tilde{O}(n^3)$ time [Vassilevska Williams, Woldeghebriel, Xu 2022]. In this paper, we show a 3FRP algorithm in deterministic $\tilde{O}(n^3)$ time for undirected weighted graphs, which almost matches the size of the output. This implies that fFRP in undirected graphs can be solved in nearly optimal $\tilde{O}(n^f)$ time for all $f \geq 3$.

To construct our 3FRP algorithm, we introduce an incremental distance sensitivity oracle (DSO) for undirected graphs with $\tilde{O}(n^2)$ worst-case update time, while preprocessing time, space, and query time are still $\tilde{O}(n^3)$, $\tilde{O}(n^2)$ and $\tilde{O}(1)$, respectively, which match the static DSO [Bernstein and Karger 2009]. Here in a DSO, we can preprocess a graph so that the distance between any pair of vertices given any failed edge can be answered efficiently. From the recent result in [Peng and Rubinstein 2023], we can obtain an offline dynamic DSO from the incremental worst-case DSO, which makes the construction of our 3FRP algorithm more convenient. By the offline dynamic DSO, we can also construct a 2-fault single-source replacement path (2-fault SSRP) algorithm in $\tilde{O}(n^3)$ time, that is, from a given vertex s, we want to find the distance to any vertex t when any pair of edges fail. Thus the $\tilde{O}(n^3)$ time complexity for 2-fault SSRP is also nearly optimal.

Now we know that in undirected graphs 1FRP can be solved in $\tilde{O}(m)$ time [Nardelli, Proietti, Widmayer 2001], and 2FRP and 3FRP in undirected graphs can be solved in $\tilde{O}(n^3)$ time. In this paper, we also show that a truly subcubic algorithm for 2FRP in undirected weighted graphs does not exist under APSP hypothesis.

2012 ACM Subject Classification Theory of computation → Shortest paths; Theory of computation → Graph algorithms analysis

Keywords and phrases Graph Algorithm, Shortest Path, Replacement Path

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.57

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2411.18312` [9]

1 Introduction

The shortest path problem is one of the most fundamental problems in computer science, and the single-source shortest path problem is known to have an almost linear $\tilde{O}(m)$ time algorithm [14, 15, 17]. In a failure-prone graph $G = (V, E)$ ($n = |V|$, $m = |E|$), some edges may fail on the shortest path, so we want to find a *replacement path*. In the classical replacement path (RP) problem, given source s and destination t, we want to find the s-t

© Shucheng Chi, Ran Duan, Benyu Wang, and Tianle Xie;
licensed under Creative Commons License CC-BY 4.0
52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025).
Editors: Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis
Article No. 57; pp. 57:1–57:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

shortest path when edge e fails for every edge $e \in E$. Of course, if e is not on the original s-t shortest path in G, the replacement path is still the original shortest path, so in fact we just need to find an s-t replacement path for every edge e on the s-t shortest path in G, which counts for $O(n)$ replacement paths.

We can generalize the RP problem to any number of failed edges f, that is, finding s-t shortest paths for every failed edge set F where $|F| \leq f$. This problem is called *f-fault replacement path* (fFRP) [30], therefore 1FRP is just the original RP problem. As before, given the first $(f-1)$ failed edges and the corresponding s-t replacement shortest path, we still only need to consider the cases when the f-th failed edge is on it. Because there are $O(n)$ edges on each shortest path, the total number of replacement paths we need to find for fFRP problem is $O(n^f)$. The current well-known results of fFRP algorithms for *real-weighted* directed and undirected graphs are summarized as follows.

- RP problem in directed graphs can be solved in $O(mn + n^2 \log\log n) = O(n^3)$ time [21].
- Under APSP hypothesis[1], RP in directed graphs does not have $O(n^{3-\epsilon})$ time algorithm for any constant $\epsilon > 0$ [31]. So the $O(n^3)$-time algorithm is (conditionally) nearly optimal.
- RP problem in undirected graphs can be solved in $\tilde{O}(m)$ time[2] [26], thus also almost optimal.
- Recently, Vassilevska Williams, Woldeghebriel and Xu [30] gave a 2FRP algorithm with $\tilde{O}(n^3)$ running time for directed graphs, almost matching the 1FRP case. (Although not formulated, it also works for undirected graphs with slight modifications.)

Thus, 1FRP and 2FRP problems both have $\tilde{O}(n^3)$-time algorithms, so it is natural to ask whether 3FRP still has a $\tilde{O}(n^3)$-time algorithm. In this paper, we give such a 3FRP algorithm for undirected graphs: (Note that all algorithms in this paper are deterministic.)

▶ **Theorem 1.** *The 3FRP problem in undirected real-weighted graphs can be solved in $\tilde{O}(n^3)$ time.*

Denote the shortest path between s and t in graph G by $\pi_G(s,t)$. Then in $\tilde{O}(n^3)$ time, for every edge $d_1 \in \pi_G(s,t)$, we can find all edges of $\pi_{G-\{d_1\}}(s,t)$, then for every edge $d_2 \in \pi_{G-\{d_1\}}(s,t)$, we can find all edges of $\pi_{G-\{d_1,d_2\}}(s,t)$, then for every edge $d_3 \in \pi_{G-\{d_1,d_2\}}(s,t)$, we can find $|\pi_{G-\{d_1,d_2,d_3\}}(s,t)|$, so the algorithm outputs $O(n^3)$ distances in total. One can see the difficulty of this problem since every answer only takes $\tilde{O}(1)$ time on average.

Extend to fFRP for $f \geq 3$. For any $f \geq 3$, by Theorem 1, the fFRP problem can be solved in $\tilde{O}(n^f)$ time in undirected graphs (see the reduction in [30]). Since the size of the output of fFRP can be $\Theta(n^f)$, the running time is almost optimal. As in [30], we also take the 1-failure distance sensitivity oracle as an important subroutine, since it only takes $\tilde{O}(1)$ query time. Here an f-failure distance sensitivity oracle (DSO) is a data structure that supports distance queries between any pair of $u, v \in V$ given any f failed edges. It is widely known that 1-failure DSO takes $\tilde{O}(n^2)$ space, $O(1)$ query time, and $\tilde{O}(mn)$ construction time [11, 4]. Since current 2-failure DSOs are still hard to construct efficiently [18], we instead construct an incremental 1-failure DSO: (In the following DSO means 1-failure DSO for convenience.)

[1] APSP is an abbreviation for all-pair shortest path problem, and APSP hypothesis [31] suggests that APSP cannot be solved in truly subcubic $O(n^{3-\epsilon})$ time for any constant $\epsilon > 0$.

[2] Here $\tilde{O}(\cdot)$ hides polylogarithmic factors.

▶ **Theorem 2.** *For a given undirected graph G, there is an incremental DSO that can be constructed in $\tilde{O}(n^3)$ time, so that when we insert an edge e into G, the DSO can be maintained in worst-case $\tilde{O}(n^2)$ time. The DSO also has $\tilde{O}(n^2)$ space and $\tilde{O}(1)$ query time.*

Recently Peng and Rubinstein [27] gave a reduction from offline fully dynamic structure to worst-case incremental structure, where "offline" means all updates are known ahead. So we can obtain an offline dynamic efficient DSO. (We also give a simple proof of the reduction we need by a different method as [27], see Theorem 11.)

▶ **Theorem 3** ([27])**.** *Let $T \geq 1$ be the total number of updates. If there exists an incremental dynamic algorithm with query time Γ_q and worst-case update time Γ_u, then there is an offline dynamic algorithm with query time Γ_q and worst-case update time $O(\Gamma_u \cdot \log^2(T))$.*

▶ **Corollary 4.** *Given an undirected graph G and a sequence of $O(n)$ edge insertions and deletions, there is an offline dynamic DSO which can be constructed in $\tilde{O}(n^3)$ time, and the total update time is also $\tilde{O}(n^3)$. It can answer the distance between any pair of vertices when any edge fails in $\tilde{O}(1)$ time.*

We have known 1FRP is as hard as APSP in directed graphs [31], but in undirected graphs, 1FRP can be solved in $\tilde{O}(m)$ time [26], and 2FRP in undirected graphs can be solved in $\tilde{O}(n^3)$ time. One may wonder whether 2FRP in undirected graphs has a truly subcubic time algorithm. However, we show that it is not possible under the APSP hypothesis.

▶ **Theorem 5.** *Assuming the APSP hypothesis that APSP cannot be solved in truly subcubic $O(n^{3-\epsilon})$ time for any constant $\epsilon > 0$, then 2FRP problem in undirected weighted graphs also cannot be solved in truly subcubic time.*

We can also apply the offline dynamic DSO to get a 2-fault single-source replacement path (2-fault SSRP) algorithm for undirected graphs, that is, given a source s, for every other vertex $t \in V$, we can find $\pi_{G-\{d_1,d_2\}}(s,t)$ for all edges d_1, d_2. The reduction is very simple: we can remove an edge d_1 from the shortest path tree from s each time, and then put it back. By the offline dynamic DSO, we can answer the distance between s and t avoiding d_2 in the current graph without d_1.

▶ **Theorem 6.** *The 2-fault single-source replacement path problem can be solved in $\tilde{O}(n^3)$ time for undirected graphs.*

Note that Theorem 6 can be extended to undirected f-fault SSRP algorithm in $\tilde{O}(n^{f+1})$ time for all $f \geq 2$, which is almost optimal. Previously there are 1-fault single-source replacement path algorithms of running time $\tilde{O}(m\sqrt{n} + n^2)$ for unweighted undirected graphs [7, 12] and unweighted directed graphs [8].

Note that the 2FRP, 3FRP, and single-source 2FRP algorithms in this paper obtain every distance from DSOs, which are essentially similar to the one in [4], or the APSP table. By the properties of the DSOs and the construction of our algorithms, in these algorithms in fact we can obtain an oracle of size $\tilde{O}(n^3)$ in which we can retrieve a shortest path under failures in $O(1)$ time per edge.

Other related work. The current fastest running time for the directed all-pair shortest path (APSP) problem is $n^3/2^{\Omega(\sqrt{\log n})}$ [29]. Finding truly subcubic time algorithms for APSP with arbitrary weights is considered as a major open problem in algorithm research, and its hardness is one of the major assumptions in proving conditional lower bounds.

For replacement path (RP) and single-source replacement path (SSRP) problems, there are also many subcubic time results for graphs with small integer edge weights. In unweighted directed graphs, the RP problem can be solved in $\tilde{O}(m\sqrt{n})$ time [2]. If the edge weights are integers in $[-M, M]$, Vassilevska Williams gave a $\tilde{O}(Mn^{\omega})$ time RP algorithm [29], where $\omega < 2.371339$ is the exponent of the complexity of matrix multiplication [1, 32, 16]. Moreover, [30] gave a 2FRP algorithm in small edge weight directed graphs in $\tilde{O}(M^{2/3}n^{2.9153})$ time. For SSRP problem, there is also a $\tilde{O}(Mn^{\omega})$ time algorithm for graphs with integer weights in $[1, M]$ and algorithms with running time $\tilde{O}(M^{0.7519}n^{2.5286})$ and $\tilde{O}(M^{0.8043}n^{2.4957})$ for graphs with integer weights in $[-M, M]$. [23, 22, 25].

After the breakthrough result of efficient 1-failure DSO by Demetrescu, Thorup, Chowdhury and Ramachandran [11], there are many efforts to improve the preprocessing time [3, 4, 22, 23, 7, 28, 24], space [20], and extend to more failures. However, keeping query time $\tilde{O}(1)$ and space $\tilde{O}(n^2)$ is difficult when generalized to any constant number of f failures. The 2-failure DSO of $\tilde{O}(n^2)$ space and $\tilde{O}(1)$ query time [18] is much more complicated than the 1-failure case, so it seems impossible to generalize it to f-failure DSO in arbitrary graphs. For undirected graphs, recently Dey and Gupta gave an f-failure DSO with $O(f^4n^2\log^2(nM))$ space and $O((f\log(nM))^{O(f^2)})$ query time [13], improving the result in [19], but their construction time is still large, thus still not suitable for efficient fFRP algorithms.

Organization. In Section 2 we introduce basic notations, assumptions and concepts. In Section 3 we give a brief description of the 2FRP algorithm for undirected graphs, which is a little simpler than the one for directed graphs in [30], then give an overview on the ideas for our incremental DSO and 3FRP algorithm. We will also discuss the hardness of 2FRP and single-source replacement path under multiple edge failures in Section 4 and 5, respectively. Due to page limit, many details are omitted and can be found in the full version of this paper [9].

2 Preliminaries

2.1 Notations and Assumptions

We consider an undirected weighted graph $G = (V, E)$ and $n = |V|, m = |E|$. For any two vertices $u, v \in V$, we use $\pi_G(u, v)$ to denote the shortest path from u to v in G. If the graph is clear in the context, then we use uv to be short for $\pi_G(u, v)$. For a path P, we use $|P|, \|P\|$ to denote the length of P and the number of edges in P, respectively.

If the context is clear that a vertex z is on the shortest path $\pi_G(u, v)$ from u to v in a graph G, we use $z \oplus i$ to represent the vertex that is i vertices after z in the direction from u to v. Similarly, we define $z \ominus i$ to represent the vertex that is i vertices before z. When vertices u and v and the shortest path uv are clear in the context, for two vertices x and y on uv, we use the notation $x > y$ to say that x is farther than y from u, or $\|ux\| > \|uy\|$. Similarly $x < y$ means $\|ux\| < \|uy\|$, $x \leq y$ means $\|ux\| \leq \|uy\|$, $x \geq y$ means $\|ux\| \geq \|uy\|$.

If $x, y \in uv$, $\pi_G(x, y)$ will be the subpath from x to y on uv, and if $x \leq y$, we use the interval $[x, y]$ to denote the subpath $\pi_G(x, y)$ of uv in graph G. Thus the interval between i-th vertex and j-th vertex after u on uv is denoted as $R = [u \oplus i, u \oplus j]$.

Let P_1, P_2 be two paths. If they share a same endpoint, we use $P_1 \circ P_2$ to denote the concatenation of them. If they do not share any edge, we say P_1 avoids P_2. Sometimes for brevity, we use $\min\{P_1, P_2\}$ to denote the shorter path of P_1, P_2, which means if $|P_1| \leq |P_2|$, $\min\{P_1, P_2\} = P_1$ and otherwise $\min\{P_1, P_2\} = P_2$.

In this paper, for a path P in G, $G - P$ denotes the graph obtained by removing all **edges** of P from G. (We do not need to remove vertices in this paper.)

As in many distance oracle papers, we also use the **unique shortest path assumption**, which means the shortest path between any two vertices in any subgraph of G is unique. For the incremental DSO, we also make the unique shortest path assumption at any time. So for example, we can check whether an edge $e = (x, y)$ is on st or not by checking whether $|sx \circ (x, y) \circ yt| = |st|$ or $|sy \circ (y, x) \circ xt| = |st|$. To assume unique shortest path, we can break the ties by adding small perturbations to the edge weights. (See [10].) One way to assume unique shortest path deterministically is to compare two paths of the same length by their number of edges, then by their vertices from the end node to start node.

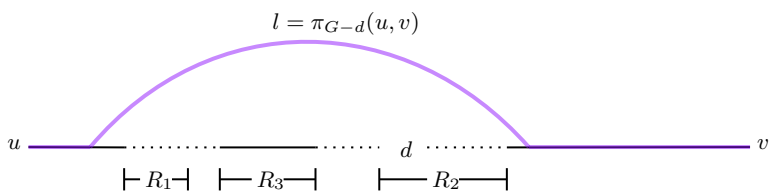
2.2 Replacement Paths

We first consider the divergence and convergence points of two paths: (As before, although our graph is undirected, we may assume a path P starts from a vertex u and ends at v.)

- For two paths P, Q both starting from u, we say they diverge at vertex $a \in P \cap Q$ if Q first gets into an edge out of P from a, that is, the subpaths of P and Q before a coincide, and we denote this divergence point a by $\Delta(P, Q)$. If P, Q are both shortest paths, their divergence point is unique.
- Symmetrically, for P, Q both ends at v, we say they converges at vertex $b \in P \cap Q$ if Q gets into an edge in P from b, and the subpaths of P and Q coincide after b. The convergence point b is denoted by $\nabla(P, Q)$.
- For two shortest paths P, Q which do not share endpoints, if they intersect, the intersection part is also a shortest path, that is, they must first converge and then diverge.

The following theorem is well known for replacement paths.

▶ **Theorem 7** (Equivalent to Claim 1 in [11]). *Let l be a 1-fault replacement path in G, i.e. $l = \pi_{G-d}(u, v)$ for an edge d. Let R_1, R_2 be two subpaths in uv with no intersection, and R_3 be a subpath between them. If l avoids both R_1 and R_2, then it avoids R_3.*



From this theorem, we can see for any f-fault replacement path $\pi_{G-F}(u, v)$ where only one failed edge in F is on the original shortest path uv, $\pi_{G-F}(u, v)$ only diverges and converges once on uv. And if two failed edges d_1, d_2 in F are on uv, they will cut uv into three intervals D_1, D_2, D_3. Because D_1, D_2, D_3 are all shortest paths in $G - F$, $\pi_{G-F}(u, v)$ may:

- diverge at D_1 and converge at D_3.
- diverge at D_1, converge at some point in D_2, diverge from D_2, then converge at D_3.

In undirected weighted graphs, a notable theorem on the structure of replacement paths says that a 1-fault replacement path $\pi_{G-f}(u, v)$ is a concatenation of a shortest path ux, an edge $e = (x, y)$, and a shortest path yv, for some vertices x and y. (Any part can degenerate to a point.) More generally,

▶ **Theorem 8** ([6]). *For any undirected weighted graph G and any set of failed edges F with $|F| = f$, an f-fault replacement path can be partitioned into a concatenation interleaving at most $f + 1$ subpaths and f edges, where each subpath is a shortest path in the graph G.*

From this theorem, we have:

▶ **Lemma 9** ([5]). *For a given pair of vertices u, v, the union of all 1-fault u-v replacement paths has $O(n)$ edges in total in undirected graphs.*

Proof. From Theorem 8, a 1-fault u-v replacement path in undirected graphs is composed of a shortest path ux, an edge (x, y), and a shortest path yv. The total number of the middle edges (x, y) is bounded by $O(n)$ since there are $O(n)$ 1-fault replacement paths, and other edges are on the shortest path trees from u and v, thus there are also at most $O(n)$ edges. ◀

2.3 DSOs

The paper uses the 1-fault DSO given by Bernstein and Karger [4] to initialize the incremental DSO. While [4] gives a 1-fault DSO in directed weighted graphs, it also works for undirected weighted graphs by a simple reduction. Suppose G is an undirected weighted graph, we introduce a directed graph G' by replacing each edge in G by a pair of directed edges with the same weight. Suppose that in graph G edge (x, y) on the shortest path st is removed (x is closer to s), then we remove the directed edge (x, y) on the shortest path from s to t in G'. We can see that the 1-fault replacement path $\pi_{G'-(x,y)}(s, t)$ cannot go through edge (y, x). This is because that no edge on $\pi_{G'}(s, x)$ is removed, so $\pi_{G'-(x,y)}(s, x) = \pi_{G'}(s, x)$ does not go through (y, x). Therefore, $\pi_{G-(x,y)}(s, t) = \pi_{G'-(x,y)}(s, t)$.

There is a result in [27] to view worst-case incremental structures as offline fully dynamic structures.

▶ **Theorem 10** ([27]). *Let $T \geq 1$ be the total number of updates. Suppose there exists an incremental dynamic algorithm with query time Γ_q and worst-case update time Γ_u, then there is a dynamic algorithm for deletions-look-ahead setting with query time Γ_q and worst-case update time $O(\Gamma_u \cdot \log^2(T))$.*

This theorem can be used to utilize our incremental DSO in Section 3.2 as an offline fully dynamic DSO, and here we also prove the reduction we need in a self-contained way:

▶ **Theorem 11.** *Starting from a graph of $O(n)$ vertices, if we have an incremental DSO of $\tilde{O}(n^2)$ space with preprocessing time $\tilde{O}(n^3)$, worst-case update time $\tilde{O}(n^2)$ and query time $\tilde{O}(1)$, then it can be transformed into an offline fully dynamic DSO, such that if the total number of updates is $T = O(n)$, the total preprocessing and update time is $\tilde{O}(n^3)$ and the query time is still $\tilde{O}(1)$.*

Proof. In the offline fully dynamic DSO, let the initial graph be G_0 and the graph after the i-th update be G_i. We first list all graphs $G_0, G_1, \cdots, G_T$. For an interval of integers $[i, j]$, define the graph $G_{[i,j]} = G_i \cap G_{i+1} \cap \cdots \cap G_j$. Then for an interval $[i', j'] \subseteq [i, j]$, we know that $G_{[i,j]} \subseteq G_{[i',j']}$ and $|G_{[i',j']} - G_{[i,j]}| \leq j - i$.

Then we build a binary range tree on $[0, T]$, in which the root contains the graph $G_{[0,T]}$, the children of the root contain the graphs $G_{[0,\lfloor T/2 \rfloor]}$ and $G_{[\lfloor T/2 \rfloor, T]}$, respectively, and each leaf contains a graph G_i. So we can build the DSOs on this tree from the root using the incremental structure, since $G_R \subseteq G_{R'}$ if R' is a child of R. Then we can see the total construction time is $\tilde{O}(n^3)$, and we can use the DSOs on leaves to answer queries. ◀

3 Technical Overview

In this section, we first give a brief description of 2FRP algorithm for undirected graphs, which can list all edges of all 2-fault replacement paths in $\tilde{O}(n^3)$ time, so that we can find the third failed edge for our 3FRP algorithm. Then the high-level ideas for the incremental DSO and 3FRP algorithm are discussed.

3.1 2FRP Algorithm between s, t

All 1-fault replacement paths are easy to find in $\tilde{O}(mn)$ time since we can delete each edge on st and then run Dijkstra's algorithm [14] from s. Then for edges $d_1 \in st$ and $d_2 \in \pi_{G-d_1}(s,t)$, consider whether d_2 is on st or not.

3.1.1 Only d_1 is on st

By the methods in [30], we create a graph H from $G - st$ by adding new vertices as follows. Let N be a number that is large enough. (Namely, N can be the sum of all edge weights in G.)

- First let H be a copy of $G - st$, that is, remove all edges on $\pi_G(s,t)$ from G.
- For every edge $d = (x, y) \in st$ and x is before y on st, introduce two new vertices d^- and d^+.
  - For every vertex x' on sx, the node d^- is connected with x' by an edge with weight $|sx'| + N$.
  - For every vertex y' on yt, the node d^+ is connected with y' by an edge with weight $|y't| + N$.

The number of vertices in H is still $O(n)$, so we construct a 1-failure DSO on H with $\tilde{O}(n^3)$ construction time, $\tilde{O}(n^2)$ space and $O(1)$ query time by [4]. We say the edge (d^-, x') corresponds to the path sx' since their length differs by a fixed number N, similarly (y', d^+) corresponds to the path $y't$. (Here N guarantees that any shortest path between new vertices will not travel through other new vertices.) Since d_2 is not on st, we can obtain $\pi_{G-\{d_1,d_2\}}(s,t)$ by querying the DSO. We have:

▶ **Lemma 12.** *Given G and $d_1 \in st$, $d_2 \notin st$, in the graph H we defined,*

$$|\pi_{G-\{d_1,d_2\}}(s,t)| = |\pi_{H-d_2}(d_1^-, d_1^+)| - 2N$$

Proof. The proof is easy to see since the first edge (d_1^-, x') and last edge (y', d_1^+) on the path $\pi_{H-d_2}(d_1^-, d_1^+)$ have:

$$|(d_1^-, x')| = |sx'| + N,\ |(y', d_1^+)| = |y't| + N$$

Also x' is before d_1 and y' is after d_1, so on $\pi_{H-d_2}(d_1^-, d_1^+)$ the first edge and last edge can be replaced by the original shortest paths $s'x$ and $y't$, respectively, so it is a path in $G - \{d_1, d_2\}$, and $|\pi_{G-\{d_1,d_2\}}(s,t)| \le |\pi_{H-d_2}(d_1^-, d_1^+)| - 2N$. Similarly the subpaths of $\pi_{G-\{d_1,d_2\}}(s,t)$ before divergence point x' and after convergence point y' on st can be replaced by edges (d_1^-, x') and (y', d_1^+) in H, respectively (with value $2N$ added), so $|\pi_{G-\{d_1,d_2\}}(s,t)| \ge |\pi_{H-d_2}(d_1^-, d_1^+)| - 2N$. ◀

Since the 1-failure DSO supports path retrieving [11, 4] in $O(1)$ time per edge, we can restore all paths $\pi_{G-\{d_1,d_2\}}(s,t)$ in $\tilde{O}(n^3)$ time.

3.1.2 Both d_1, d_2 are on st

Our structure in this case is also similar to [30], but is simpler since G is undirected here. If we use the graph H before, the distance $\pi_H(d_1^-, d_2^+)$ cannot capture $\pi_{G-\{d_1,d_2\}}(s,t)$ since it only formulates the subpaths before divergence point and after convergence point on st, but $\pi_{G-\{d_1,d_2\}}(s,t)$ may go through some edges between d_1 and d_2 on st, which are not included in H.

W.l.o.g., assume $d_1 = (x_1, y_1)$ is before $d_2 = (x_2, y_2)$ on st, so $s \le x_1 < y_1 \le x_2 < y_2 \le t$ on the shortest path st. For all pairs of d_1, d_2, we want to find the paths: (w, a, b, z are all on st.)

$$P(d_1, d_2) = \min_{w \le x_1, y_1 \le a \le b \le x_2, z \ge y_2} \{sw \circ \pi_{G-st}(w, a) \circ ab \circ \pi_{G-st}(b, z) \circ zt\}$$

and also:

$$P'(d_1, d_2) = \min_{w \le x_1, y_1 \le b \le a \le x_2, z \ge y_2} \{sw \circ \pi_{G-st}(w, a) \circ ab \circ \pi_{G-st}(b, z) \circ zt\}$$

First we run the APSP algorithm on the graph $G - st$, then find the paths $U(d_1, a) = \min_{w \le x_1}\{sw \circ \pi_{G-st}(w, a)\}$ for all edge $d_1 = (x_1, y_1)$ and vertex $a \ge y_1$ on st in $O(n^3)$ time, since we need $O(n)$ time to enumerate w for every pair of d_1 and a. Symmetrically also find $U'(d_2, b) = \min_{z \ge y_2}\{\pi_{G-st}(b, z) \circ zt\}$ for all edge $d_2 = (x_2, y_2)$ and vertex $b \le x_2$ on st in $O(n^3)$ time. Then for every pair of $d_1 = (x_1, y_1), d_2 = (x_2, y_2)$ and $y_1 \le b \le x_2$, we can find:

$$W(d_1, d_2, b) = \min_{w \le x_1, b \le a \le x_2} \{sw \circ \pi_{G-st}(w, a) \circ ab\}$$

That is, the path diverges before d_1, converges with st at some point a between b and x_2, then goes through subpath of st to b. If $b = x_2$, it is just $U(d_1, x_2)$. Then we can compute $W(d_1, d_2, b)$ for b from x_2 backward to y_1. If (b, b') with $b < b'$ is an edge on st, the path $W(d_1, d_2, b)$ can either first go through $W(d_1, d_2, b')$ or directly go to b, so $W(d_1, d_2, b) = \min\{U(d_1, b), W(d_1, d_2, b') \circ (b', b)\}$. So the total time to get all of $W(d_1, d_2, b)$ is still $O(n^3)$.

Thus, given d_1, d_2, we can get $P'(d_1, d_2)$ from $W(d_1, d_2, b) \circ U'(d_2, b)$ by enumerating all possible b between d_1, d_2 in $O(n)$ time. $P(d_1, d_2)$ can be obtained similarly. So the total time is $O(n^3)$, and it is also easy to list all edges on every $\pi_{G-\{d_1,d_2\}}(s,t)$.

3.2 Ideas of Incremental DSO

In 3FRP problem, consider the case that only d_1 is on st but d_2, d_3 are not, the 2-failure DSO of $\tilde{O}(n^2)$ space and $\tilde{O}(1)$ query time [18] cannot be directly applied here since its construction time is too large. However, since the number of possible "second failures" d_2 is $O(n)$ by Lemma 9, a dynamic 1-failure DSO with $\tilde{O}(n^2)$ edge update time will help. We can delete every d_2, query for every d_1 and d_3 as the method in Section 3.1.1, then add d_2 back.

Thus below we first introduce ideas toward Theorem 2 to build an incremental DSO with construction time $\tilde{O}(n^3)$, worst-case update time $\tilde{O}(n^2)$ and query time $\tilde{O}(1)$. By the reduction in Theorem 11, we can construct an offline dynamic DSO with $\tilde{O}(n^2)$ update time. In the incremental DSO, when inserting an edge $e = (x, y)$, let G' be the new graph $G' = G \cup e$.

Inspired by APSP update. It is not hard to maintain APSP incrementally in $O(n^2)$ time. When inserting an edge $e = (x, y)$, we determine whether e provides a shorter path for any pair of vertices u, v. That is, for each pair (u, v), we compute the updated shortest path:

$$\pi_{G'}(u, v) = \min\{uv, ux \circ e \circ yv, uy \circ e \circ xv\}.$$

Following previous works [11], our goal is to construct a DSO that maintains the following information:

- A shortest path tree rooted at each vertex u.
- For all vertex pair (u, v), the detour $\pi_{G-(u\oplus i)(v\ominus j)}(u, v)$, where i, j are either 0 or integer powers of 2, satisfying $i + j < \|uv\|$.

Here, $\|uv\|$ denotes the number of edges in the shortest path uv. Note that in our definition, $G - (u \oplus i)(v \ominus j)$ represents the graph obtained by removing the edges in $(u \oplus i)(v \ominus j)$ rather than removing vertices, which slightly differs from previous works [11, 4]. However, under the **weak subpath** constraint (as will be formally defined later), we indeed only consider the detours avoiding one edge in the range which also avoids all the edges and vertices in the whole range $[u \oplus i, v \ominus j]$. We want to maintain such detours after inserting a new edge $e = (x, y)$ in $\tilde{O}(1)$ time per detour.

Weak subpath and proper form. As in [4], to have an efficient construction time, instead of maintaining $\pi_{G-(u\oplus i)(v\ominus j)}(u, v)$ avoiding the whole range, we can alternatively maintain the longest shortest detour avoiding one edge in the range $\max_{f\in(u\oplus i)(v\ominus j)} \pi_{G-f}(u, v)$. Moreover, it is easy to observe that we only need to maintain the detour $\max_{f\in(u\oplus i)(v\ominus j)} \pi_{G-f}(u, v)$ when the detour $\pi_{G-f}(u, v)$ avoids the entire range $[u \oplus i, v \ominus j]$.

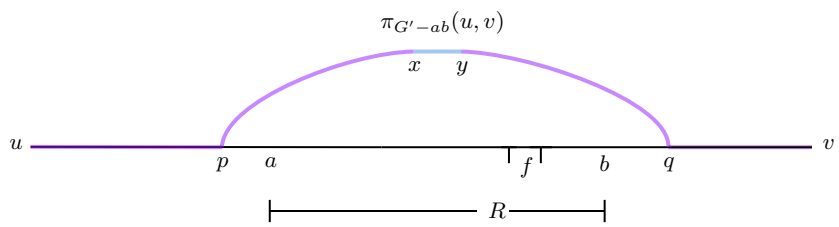
So we define a subpath $ab \subseteq uv$ to be **weak** under uv, if there exists an edge $f \in ab$, $\pi_{G-f}(u, v) = \pi_{G-ab}(u, v)$. Here f is called a **weak point** of ab (under uv). Note that by Theorem 8, the detour $\pi_{G-f}(u, v)$ is a concatenation of a shortest path ux, an edge (x, y) and a shortest path yv in G. Therefore, we can store the replacement path for weak subpaths $ab \subseteq uv$ explicitly. We call such concatenation of any shortest path ux, any edge (x, y) and a shortest path yv a **proper form**. (Any part can degenerate to a point.)

In the incremental DSO, for every pair of vertices u, v and $a = u \oplus i, b = v \ominus j$ where i, j are either 0 or integer powers of 2, we maintain $\varpi_{G-ab}(u, v)$ to be a null path, or of the proper form, such that:

- If ab is **weak** under uv, we store the correct replacement path $\varpi_{G-ab}(u, v) = \pi_{G-ab}(u, v)$ in **proper form**.
- If ab is not weak under uv, $\varpi_{G-ab}(u, v)$ can be either some path from u to v that avoids ab in proper form, or a null path of length $+\infty$.

▶ Remark 13. Note that in this structure we generally can hardly say whether a given ab is weak or not under uv, even if we know the value of $\varpi_{G-ab}(u, v)$. We also point out that even if ab is weak under uv we do not record the position of the weak point.

Consider the case that the shortest $u - v$ path does not change after inserting an edge $e = (x, y)$, that is, $\pi_G(u, v) = \pi_{G'}(u, v)$. If ux and yv do not intersect with the range $ab \subseteq uv$, then $ux \circ e \circ yv$ is a path between u and v in $G' - ab$.



However, if $|ux \circ e \circ yv|$ is shorter than the original $|\varpi_{G-ab}(u,v)|$, we can hardly check whether ab is weak under $\pi_{G'}(u,v)$ in G', and even if it is weak, we cannot efficiently find the weak point f such that $\pi_{G'-f}(u,v) = \pi_{G'-ab}(u,v)$. So in this case, we just store $\varpi_{G-ab}(u,v) = ux \circ e \circ yv$, then it is the correct replacement path $\pi_{G'-ab}(u,v)$ when ab is weak under $\pi_{G'}(u,v)$.

Also note that the proper form for all stored $\varpi_{G-ab}(u,v)$ makes it easier to locate vertices and compute intersections between paths.

3.2.1 Ideas of DSO Query

The correctness of DSO query is guaranteed by Theorem 15, showing that we can answer the shortest path between u, v avoiding any subpath ab of uv if ab is weak under uv. Note that in the query stage, we in fact only care about the case where ab is a single edge (because it is an 1-DSO), and an interval consisting of a single edge is, of course, weak.

We sometimes need to check whether a path is in proper form, and if so, transform it to its proper form representation.

▶ **Definition 14.** *Let P be a path from u to v in G and let R be a subpath of uv in G. We define a transform $T_{G,R}(P)$ on P to be*

- *the path P of the proper form, if P can be transformed to the proper form and it avoids edges of R.*
- *a null path with weight $+\infty$, otherwise.*

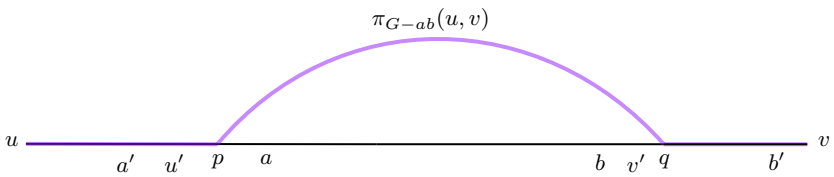
When this transform is used in this section, P is given by a concatenation of several shortest paths and edges in G, and we want to transform it into the proper form in G or the new graph G'. The transform $T_{G,R}$ can be done in $\tilde{O}(1)$ time, while the proof refers to the full version of this paper [9].

Now we state the DSO query theorem formally:

▶ **Theorem 15.** *Let ab be a subpath of uv. By the information we maintain in the DSO, we can calculate the length of $\varpi_{G-ab}(u,v)$ in $\tilde{O}(1)$ time. W.l.o.g., suppose a is closer to u than b is. Let i be the largest integer power of 2 s.t. $u \oplus i$ falls in ua. Similarly j is the largest integer power of 2 s.t. $v \ominus j$ falls in bv. Let $a' = a \ominus i, u' = u \oplus i, v' = v \ominus j, b' = b \oplus j$. (Note that if $a = u$, $a' = u' = u$ and if $b = v$, $b' = v' = v$.) Then*

$$\begin{aligned}\varpi_{G-ab}(u,v) = \min\Big\{ & T_{G,ab}(ua' \circ \varpi_{G-ab}(a',b') \circ b'v),\ \varpi_{G-u'v'}(u,v), \\ & T_{G,ab}(ua' \circ \varpi_{G-av'}(a',v)), T_{G,ab}(\varpi_{G-u'b}(u,b') \circ b'v)\Big\}.\end{aligned} \tag{1}$$

Proof refers to the full version of this paper [9].



3.2.2 Ideas of Incremental Update

Now we see how the DSO update is implemented. For the detours on $\varpi_{G-(u\oplus i)(v\ominus j)}(u,v)$, where i,j are zero or integer powers of 2, we consider two cases separately as below: either $\pi_{G'}(u,v) \neq uv$, or $\pi_{G'}(u,v) = uv$. In both cases, the algorithm further studies many subcases based on the relative positions between $u \oplus i, v \ominus j$, the divergent and convergent points of uv and $\pi_{G'}(u,v)$, and e.

Recall that in our incremental structure, the graph before introducing edge e is called G, and the graph $G \cup \{e\}$ is called G'. Let $p = \Delta(ux, uv)$ be the last point that $\pi_G(u,x)$ and $\pi_G(u,v)$ shares, and similarly $q = \nabla(yv, uv)$ be the first point that $\pi_G(y,v)$ and $\pi_G(u,v)$ shares. Let $a = u \oplus i, b = v \ominus j$. The incremental update requires handling multiple distinct scenarios, determined by the relative positions between nodes a, b, p, q, and edge $e = (x,y)$. We present selected representative cases with corresponding update mechanisms from the full version of this paper [9].

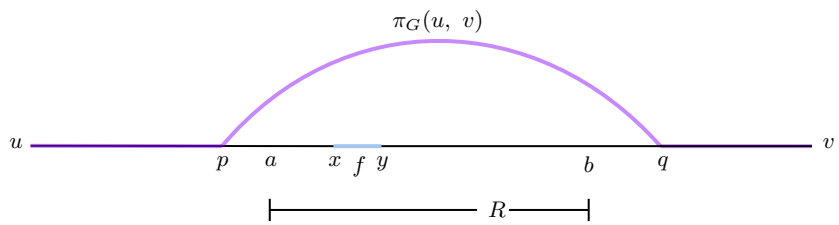
In the case discussions, by default we assume that $R = \pi_{G'}(u \oplus i, v \ominus j)$ is weak under $\pi_{G'}(u,v)$. This is because we only care about the path $\pi_{G'-R}(u,v)$ when R is weak. If R is not weak, because of the transform of $T_{G',R}$, we can always make sure that the path given by the algorithm is a proper form or a null path.

3.2.2.1 $\pi_G(u,v) \neq \pi_{G'}(u,v)$

We know in this case $e = (x,y)$ is on $\pi_{G'}(u,v)$. W.l.o.g., suppose that x is nearer to u than y. We can see that uv is a shortest $u-v$ path in $G = G' - e$, i.e. $uv = \pi_{G'-e}(u,v)$. Recall that $p = \Delta(ux, uv)$, $q = \nabla(yv, uv)$, $a = u \oplus i, b = v \ominus j$, then $R = \pi_{G'}(a,b)$ and we want to find $\varpi_{G'-R}(u,v)$. So the points a, b, x, y and p, q are all on the new shortest path $\pi_{G'}(u,v)$.

CASE 1. $p \leq a \leq x < y \leq b \leq q$,

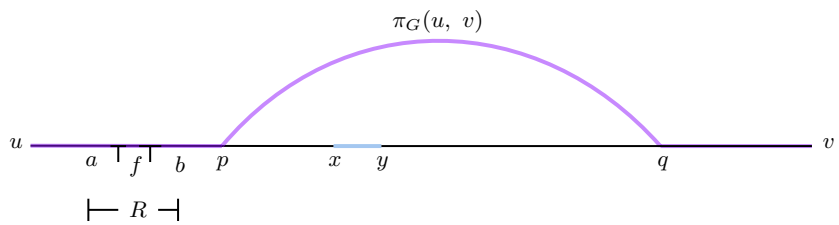
$$\varpi_{G'-R}(u,v) = T_{G',R}(uv).$$



In this case, since $uv = \pi_{G'-e}(u,v)$, $e \in R$, $R = \pi_{G'}(a,b)$, and uv avoids R, so R is weak with a weak point $f = e$, and $\pi_{G'-e}(u,v) = uv$, which must pass the transform $T_{G',R}$.

CASE 2. $a \leq b \leq p \leq x < y \leq q$,

$$\varpi_{G'-R}(u,v) = \min\{T_{G',R}(\varpi_{G-R}(u,v)), T_{G',R}(\varpi_{G-R}(u,x) \circ e \circ yv)\}.$$

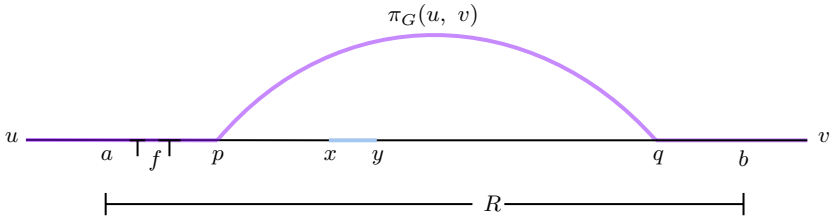


- If $\pi_{G'-R}(u,v)$ does not go through e, similar as **CASE 1**, it is in $G'-e$, which equals G. Since R is weak under $\pi_{G'}(u,v)$, it is also weak under uv, so $\pi_{G'-R}(u,v) = \varpi_{G-R}(u,v)$.
- If $\pi_{G'-R}(u,v)$ goes through e, by Theorem 7 and 8, it takes a shortest $u-x$ path that avoids R, then goes along $\pi_{G'}(x,v)$, i.e. it equals $\pi_{G-R}(u,x) \circ e \circ yv$. Suppose $f \in ab$ is a weak point of R under $\pi_{G'}(u,v)$, $\pi_{G-f}(u,x) \circ e \circ yv = \pi_{G'-f}(u,v) = \pi_{G'-R}(u,v) = \pi_{G-R}(u,x) \circ e \circ yv$, so R is also weak under ux with a weak point f, so $\varpi_{G-R}(u,x) = \pi_{G-R}(u,x)$.

This means if $\varpi_{G-R}(u,x) \circ e \circ yv$ is not a proper form, $\pi_{G'-R}(u,v)$ does not go through e, so $\varpi_{G'-R}(u,v) = uv$. Otherwise, we should choose the minimum of both. Note that if $\varpi_{G-R}(u,x)$ intersects with yv, $\varpi_{G-R}(u,x) \circ e \circ yv$ cannot be the shortest one and also cannot pass the transform $T_{G',R}$.

CASE 6. $a \le p \le x < y \le q \le b$,[3]

$\varpi_{G'-R}(u,v) = T_{G',R}(\varpi_{G-ab}(u,v))$.



Assume R is weak under $\pi_{G'}(u,v)$ and $f \in R$ is a weak point of R. $\pi_{G'-f}(u,v)$ avoids the whole interval R, so it avoids e. As a result, $\pi_{G'-f}(u,v)$ is a path in G, i.e. $\pi_{G'-f}(u,v) = \pi_{G-f}(u,v)$. If f is not in uv, $\pi_{G'-f}(u,v) = \pi_{G-f}(u,v) = uv$. Since uv does not avoid the whole interval R, this is contradictory with the fact that f is a weak point of R, so $f \in uv$ and $\pi_{G-f}(u,v) = \pi_{G'-f}(u,v) = \pi_{G'-R}(u,v)$. As a result, $f \in ap \cup qb$, and $\pi_{G-f}(u,v)$ avoids R.

If $\pi_{G-f}(u,v)$ intersects $ab = \pi_G(a,b)$, it can only intersects $ab \setminus R = pq = \pi_G(p,q)$. By Theorem 7, it is impossible that $\pi_{G-f}(u,v)$ intersects pq but avoids ap and qb, so $\pi_{G-f}(u,v)$ avoids ab, and ab is weak under uv with weak point f, thus $\varpi_{G'-R}(u,v) = \pi_{G-f}(u,v) = \varpi_{G-ab}(u,v)$.

Also, if $\varpi_{G-ab}(u,v)$ does not pass the transform $T_{G',R}$, then R is not weak under $\pi_{G'}(u,v)$, so the null path is legal for $\varpi_{G'-R}(u,v)$.

3.2.2.2 $\pi_G(u,v) = \pi_{G'}(u,v)$

As before, let $a = u \oplus i$, $b = v \ominus j$, $R = \pi_{G'}(a,b)$, and we want to find $\varpi_{G'-R}(u,v)$. Also suppose that R is weak under $\pi_{G'}(u,v)$ in the new graph G', and f is a weak point of it. The update algorithm goes as follows.

- Set $\varpi_{G'-R}(u,v) \leftarrow T_{G',R}(\varpi_{G-R}(u,v))$
- Execute the following algorithm, then swap x, y and repeat again
- Let $p = \Delta(ux, uv)$ be the last point that $\pi_G(u,x)$ and $\pi_G(u,v)$ shares, and similarly $q = \nabla(yv, uv)$ be the first point that $\pi_G(y,v)$ and $\pi_G(u,v)$ shares, consider the following cases

[3] We keep the case numbers same as those in the full version of this paper [9] for consistency.

- If $p \le a < b \le q$, set $\varpi_{G'-R}(u,v) \leftarrow \min\{\varpi_{G'-R}(u,v), T_{G',R}(ux \circ e \circ yv)\}$
- If $a < p \le b \le q$, set $\varpi_{G'-R}(u,v) \leftarrow \min\{\varpi_{G'-R}(u,v), T_{G',R}(\varpi_{G-ap}(u,x) \circ e \circ yv)\}$
- If $p \le a \le q < b$, set $\varpi_{G'-R}(u,v) \leftarrow \min\{\varpi_{G'-R}(u,v), T_{G',R}(ux \circ e \circ \varpi_{G-qb}(y,v))\}$
- If $a < p \le q < b$, do nothing
- If $a < b \le p < q$, set $\varpi_{G'-R}(u,v) \leftarrow \min\{\varpi_{G'-R}(u,v), T_{G',R}(\varpi_{G-R}(u,x) \circ e \circ yv)\}$
- if $p < q \le a < b$, set $\varpi_{G'-R}(u,v) \leftarrow \min\{\varpi_{G'-R}(u,v), T_{G',R}(ux \circ e \circ \varpi_{G-R}(y,v))\}$

We list some lemmas we need. The proofs refer to the full version of this paper [9].

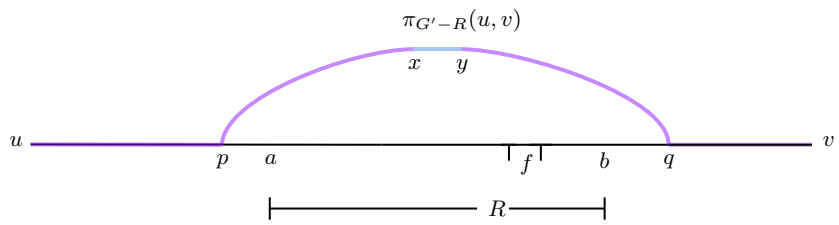
▶ **Lemma 16.** *For any edge f, if $\pi_{G'-f}(u,v)$ does not go through $e = (x,y)$, then $\pi_{G'-f}(u,v) = \pi_{G-f}(u,v)$.*

▶ **Lemma 17.** *For any edge $f \in pq$, if $\pi_{G'-f}(u,v)$ goes through $e = (x,y)$ and u is closer to x than to y in G', then $\pi_{G'-f}(u,v) = ux \circ e \circ yv$.*

▶ **Lemma 18.** *If R is weak under $\pi_{G'}(u,v)$ in G' and $\pi_{G'-R}(u,v)$ does not go through e, then $\varpi_{G'-R}(u,v) = \varpi_{G-R}(u,v)$.*

In Lemma 18 we can see that if $\varpi_{G-R}(u,v)$ does not pass the transform $T_{G',R}$, R is not weak or $\varpi_{G'-R}(u,v)$ must pass e. So in the algorithm we first set $\varpi_{G'-R}(u,v)$ to be $T_{G',R}(\varpi_{G-R}(u,v))$. Then we consider the cases that $\varpi_{G'-R}(u,v)$ goes through $e = (x,y)$ (W.l.o.g., assume u is closer to x than to y):

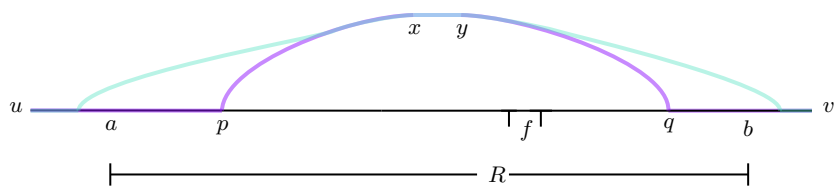
CASE 1. $p \le a < b \le q$,



By Lemma 17, we set $\varpi_{G'-R}(u,v) = T_{G',R}(ux \circ e \circ yv)$. (See Remark 13 for discussions.)

CASE 4. $a < p \le q < b$,

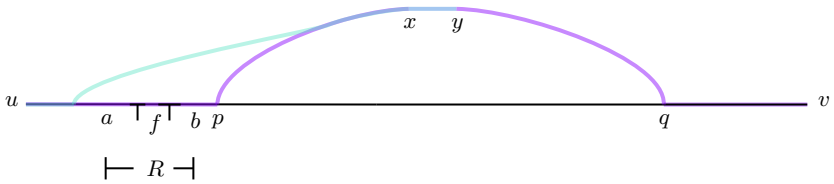
We can prove that in this case if R is weak then $\pi_{G'-R}(u,v)$ cannot go through e.



Prove by contradiction. Suppose R is weak under $\pi_{G'}(u,v)$ with weak point f and $\pi_{G'-R}(u,v)$ goes through e (first x then y), then $\pi_{G'}(u,x) = ux$ and $\pi_{G'}(y,v) = yv$ since they cannot go through e. We know $\pi_{G'-f}(u,v)$ is a 1-replacement-path in G', so by Theorem 8, it equals a concatenation of a shortest path, an edge, and a shortest path in G'. However, $a < p$ means that $\pi_{G'-R}(u,x)$ is different from ux, so it is a concatenation of at least two shortest paths in G'. Same for $\pi_{G'-R}(y,v)$. Therefore, $\pi_{G'-R}(u,v)$ equals the shortest $u-x$ path avoiding R plus e plus the shortest $y-v$ path avoiding R, which will be a concatenation of at least three shortest paths in G, contradicting to R is weak.

CASE 5. $a < b \leq p < q$,

$$\varpi_{G'-R}(u,v) = T_{G',R}(\varpi_{G-R}(u,x) \circ e \circ yv).$$



Suppose that R is weak under uv with a weak point f. By assumption, $\pi_{G'-f}(u,v)$ goes through e, so it equals $\pi_{G'-f}(u,x) \circ e \circ \pi_{G'-f}(y,v)$. By Lemma 16, $\pi_{G'-f}(y,v) = \pi_{G-f}(y,v) = yv$ and $\pi_{G'-f}(u,x) = \pi_{G-f}(u,x)$. We can see that R is also weak under ux with weak point f in G', because $\pi_{G'-f}(u,x)$ is a part of $\pi_{G'-f}(u,v)$, which avoids R. Therefore, by Lemma 18, $\pi_{G'-f}(u,x) = \pi_{G'-R}(u,x) = \varpi_{G-R}(u,x)$.

3.3 Ideas of 3FRP Algorithm

As in the 2FRP algorithm in Section 3.1, we also consider how many of d_1, d_2, d_3 are on the original shortest path st, and use different methods to solve them.

3.3.1 Only d_1 is on st

To solve the case that only d_1 is on st but d_2, d_3 are not, we use a similar structure as Section 3.1.1, that also starts from H which introduces new vertices to $G - st$. We further note that from Lemma 9, for all possible d_1, the union of possible "second failures" d_2 such that $d_2 \in \pi_{G-d_1}(s,t)$ is $O(n)$.

By the incremental 1-failure DSO in Section 3.2 and the reduction from offline dynamic structure to worst-case incremental structure in Theorem 11, we can get an offline dynamic 1-failure DSO with $\tilde{O}(n^2)$ update time. Then for every possible d_2 we can delete it from H "temporarily" and query

$$|\pi_{(H-d_2)-d_3}(d_1^-, d_1^+)| - 2N$$

for every d_1 and d_3, which equals $|\pi_{G-\{d_1,d_2,d_3\}}(s,t)|$. After that, we add d_2 back to H.

3.3.2 Only d_1, d_2 are on st

To solve the case that d_1, d_2 are on st but d_3 is not, we apply the incremental DSO on the binary partition structure as in the 2-failure DSO of [18], that is, we build a binary range tree on the path st, and define two graphs $H_{i,0}$ and $H_{i,1}$ (based on the graph H in Section 3.1.1) in each level i of the range tree, where in $H_{i,0}$ we remove all odd ranges on level i and in $H_{i,1}$ we remove all even ranges. Given d_1 and d_2, we find the first level in which d_1 and d_2 are in different ranges, then d_1 is in an even range and d_2 is in an adjacent odd range, and let m be the point that separates these two ranges. Consider the following cases for the intersection of $\pi_{G-\{d_1,d_2,d_3\}}(s,t)$ and the edges between d_1 and d_2 on st:

- If $\pi_{G-\{d_1,d_2,d_3\}}(s,t)$ does not travel through edges between d_1 and d_2, we can use DSO on H to give the answer.
- If $\pi_{G-\{d_1,d_2,d_3\}}(s,t)$ only goes through edges between d_1 and m, we can use $H_{i,0}$ which does not contain edges between m and d_2. So the distance between d_1^- and d_2^+ in $H_{i,0} - d_1$ avoiding d_3 can give the answer. (Note that edges between s and d_1 in $H_{i,0}$ will not affect the answer, similarly for edges between d_2 and t.)

- If $\pi_{G-\{d_1,d_2,d_3\}}(s,t)$ only goes through edges between m and d_2, symmetrically we can use $H_{i,1} - d_2$.
- The only case left is that $\pi_{G-\{d_1,d_2,d_3\}}(s,t)$ travels through m. In this case, we can capture the replacement path using the midpoint m. Namely, we compute its length by:

  $$|\pi_{G-\{d_1,d_2,d_3\}}(s,m)| + |\pi_{G-\{d_1,d_2,d_3\}}(m,t)|.$$

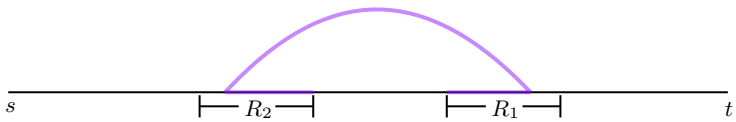
  Using the offline dynamic DSO on $H_{i,0} - d_1$ and $H_{i,1} - d_2$, we can answer these two subpaths separately.

So we need to build offline dynamic DSOs on $H_{i,0}$ and $H_{i,1}$ in each level i. Since there are $O(\log n)$ levels, the total time is still $\tilde{O}(n^3)$.

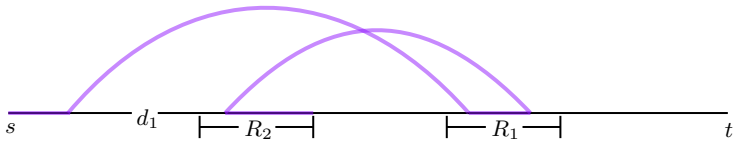
3.3.3 All of d_1, d_2, d_3 are on st

We assume the order of them on st is just d_1, d_2, d_3, then they will cut st into four ranges D_1, D_2, D_3, D_4 in order. The difficulty will come from the fact that the path $\pi_{G-\{d_1,d_2,d_3\}}(s,t)$ can go through both of D_2 and D_3, in any order. To solve this case, we first build a structure in $\tilde{O}(n^3)$ time which can answer the following queries in $\tilde{O}(1)$ time:

- Given two vertex-disjoint ranges R_1, R_2 of consecutive edges on the original shortest path st (R_1, R_2 can be single vertices), answer the shortest path that starts from the leftmost (rightmost) vertex of R_1, diverges from R_1, converges in R_2, and finally ends at the leftmost (rightmost) vertex of R_2.



- Given an edge d_1 on st and two vertex-disjoint ranges R_1, R_2 of consecutive edges on st after d_1, answer the shortest path that starts from s, diverges before d_1, converges in R_1, and diverges from R_1, then converges in R_2, and finally ends at the leftmost (rightmost) point of R_2.

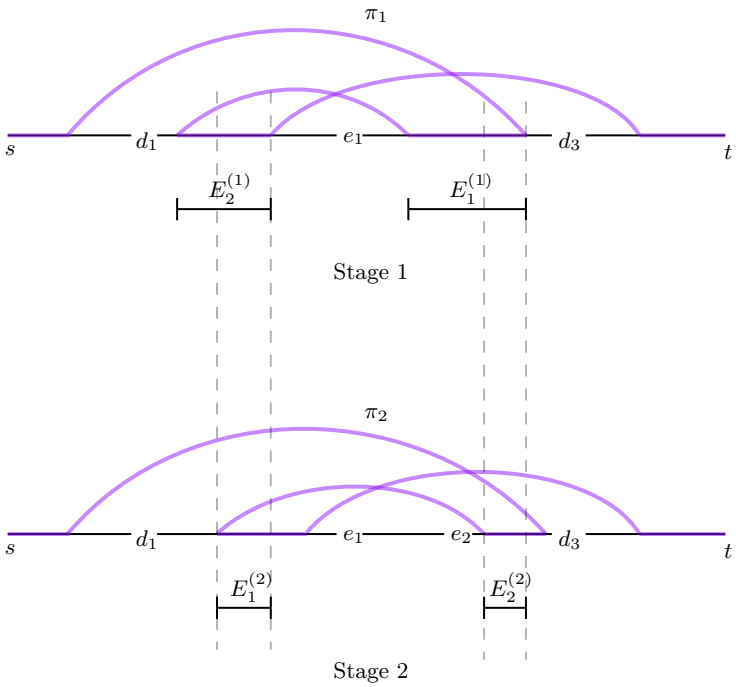


In the structure we store the paths for all ranges R_1, R_2 in the binary range tree on st, then given any R_1, R_2 in the query, they can be split into $O(\log n)$ ranges in the binary range tree, so the queries for R_1, R_2 can be answered in $\tilde{O}(1)$ time.

Similar to Section 3.1.2, by this structure, when given d_1, d_2, d_3 and a vertex b on st, it takes $\tilde{O}(1)$ time to find the shortest path goes through b avoiding d_1, d_2, d_3. So it will take $\tilde{O}(n)$ time to find the shortest path goes through both of D_2 and D_3 avoiding d_1, d_2, d_3, but we need the time bound to be $\tilde{O}(1)$ on average. For every pair of d_1 and d_3, the main ideas are as follows.

- First find the middle edge e_1 on the range between d_1 and d_3, and use $\tilde{O}(n)$ time to find the shortest path $\pi_{G-\{d_1,d_3,e_1\}}(s,t)$.
- For other edges d_2 between d_1 and d_3, if d_2 is not on $\pi_{G-\{d_1,d_3,e_1\}}(s,t)$, then either $\pi_{G-\{d_1,d_2,d_3\}}(s,t)$ is equal to $\pi_{G-\{d_1,d_3,e_1\}}(s,t)$ or it goes through e_1, so this case can be solved in $\tilde{O}(1)$ time.

- The intersection of $\pi_{G-\{d_1,d_3,e_1\}}(s,t)$ and st between d_1 and d_3 consists of at most two ranges, then find the middle edge e_2 of the larger range, and find the shortest path avoiding d_1, d_3, e_1, e_2 and goes through two ranges between them. This can also be done in $\tilde{O}(n)$ time.
- As before, for d_2 not on the new path we can solve it by querying the paths that go through e_1 or e_2, so next we only need to consider the edges in the intersection of all paths so far, and we can argue that the size of its intersection with the range between d_1 and d_3 will shrink by a constant factor in every two iterations, so only $O(\log n)$ iterations are needed. Thus the time for every pair of d_1 and d_3 is only $\tilde{O}(n)$.



4 Hardness of 2FRP

In this section we show that the 2FRP problem in undirected graphs is hard, assuming the APSP hypothesis. We note that in directed graphs, 1FRP is as hard as APSP [31]. However, in undirected graphs, 1FRP can be solved in almost-optimal $\tilde{O}(m)$ time [26].

▶ **Theorem 19.** *If 2FRP in undirected weighted graphs can be solved in $O(n^{3-\epsilon})$ time for any $\epsilon > 0$, then in undirected weighted graphs, the all-pairs shortest path problem can also be solved in $O(n^{3-\epsilon})$ time.*

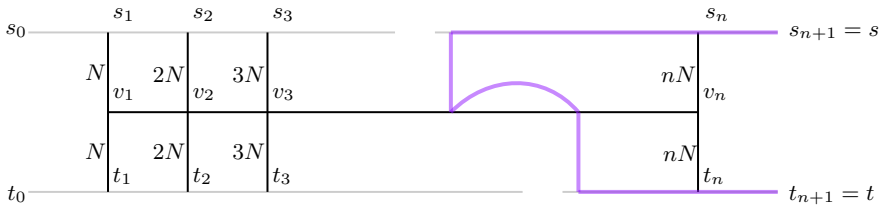
We remark that by [31], the hardness of the all-pairs shortest path problem in undirected weighted graphs is equivalent to the hardness of APSP. Therefore, any algorithm in $O(n^{3-\epsilon})$ time for 2FRP will refute the APSP hypothesis. So we immediately get the corollary:

▶ **Corollary 20.** *Assuming the APSP hypothesis that APSP cannot be solved in $O(n^{3-\epsilon})$ time for any $\epsilon > 0$, 2FRP problem in undirected weighted graphs cannot be solved in $O(n^{3-\epsilon})$ time for any $\epsilon > 0$.*

Suppose toward contradiction that we can solve 2FRP in $O(n^{3-\epsilon})$ time. Consider any undirected weighted all-pairs shortest path instance G with n nodes $v_1, v_2, ..., v_n$, we will construct an undirected weighted 2FRP instance H by adding vertices and edges on G, solving which will result in solving all-pairs shortest path in G.

Let N be a number that is larger than the sum of all weights in G. Let S be a path with $(n+2)$ nodes $(s_0, s_1, s_2, ..., s_n, s_{n+1} = s)$ with weight 0 edges (s_i, s_{i+1}) for all $0 \le i \le n$. Similarly, let T be a path with $(n+2)$ nodes $(t_0, t_1, t_2, ..., t_n, t_{n+1} = t)$ with weight 0 edges (t_i, t_{i+1}) for all $0 \le i \le n$. Moreover, we construct two matchings $E_1 = \{(s_i, v_i)\}$ with weight $w(s_i, v_i) = iN$ for all $1 \le i \le n$, and $E_2 = \{(v_i, t_i)\}$ with weight $w(v_i, t_i) = iN$ for all $1 \le i \le n$.

Let $H = S \cup E_1 \cup G \cup E_2 \cup T$. By inspection, the s to t shortest path in H is precisely the path goes from s to s_1 in S, from s_1 to v_1 to t_1, and then from t_1 to t in T. Consider any pair of failed edges, one on $\pi_H(s, s_1)$ and the other on $\pi_H(t_1, t)$. We will show the following relationship for 2FRP in H and all-pairs shortest paths in G:



▶ **Lemma 21.** *For any $1 \le i, j \le n$, let $D = \{(s_{i-1}, s_i), (t_{j-1}, t_j)\}$ be a set of two edge failures with one in S and one in T. Then regarding the failure set D:*

$$|\pi_{H-D}(s,t)| = |\pi_G(v_i, v_j)| + (i+j)N.$$

The proof refers to the full version of this paper [9]. With this lemma, we can prove Theorem 19. Considering any undirected weighted all-pairs shortest path instance G, we construct an undirected weighted graph H as above. Now we check all failure set $D = \{(s_{i-1}, s_i), (t_{j-1}, t_j)\}$ for all pairs (i, j), $1 \le i, j \le n$. By Lemma 21, we can obtain $|\pi_G(v_i, v_j)|$ from $|\pi_{H-D}(s,t)|$, which means by solving 2FRP in H we can solve all-pairs shortest paths in G.

Since we know $n' = 2(n+2) + n = O(n)$ nodes in H, if we have a truly subcubic time algorithm that solving 2FRP in time $O((n')^{3-\epsilon}) = O(n^{3-\epsilon})$ for H, then using this algorithm we can solve all-pairs shortest paths in $O(n^{3-\epsilon})$ time for G. Therefore, Theorem 19 is true.

5 Almost Optimal Single-Source Replacement Path under Multiple Edge Failures

For a graph G and a vertex s, the 2-fault single-source replacement path problem is that for every tuple of two edges d_1, d_2 and a vertex t, answer the distance between s and t in G after removing edges d_1 and d_2. We show how to solve it in almost optimal time $\tilde{O}(n^3)$ for undirected weighted graphs, via the incremental DSO.

Let T be the shortest path tree of s in G, with edges $e_1, e_2, \cdots, e_{n-1}$. Our algorithm goes as follows: we maintain an offline dynamic DSO on G. For each $1 \le i < n$, we remove e_i in G, maintain the dynamic DSO. For every vertex t_j in the subforest of e_i in T and every edge d_k on the replacement path $\pi_{G-e_i}(s, t_j)$, save the distance $\pi_{(G-e_i)-d_k}(s, t_j)$ in $G - e_i$ via querying the dynamic DSO. At the end of each step, add e_i back into G.

Correctness. For each tuple (d_1, d_2, t), if none of d_1, d_2 is in T, we know the shortest path st is the same after removing d_1, d_2. W.l.o.g., suppose $d_1 \in T$, and t is in the subtree of d_1 in T. If d_2 is not in $\pi_{G-d_1}(s,t)$, we know that $\pi_{G-d_1-d_2}(s,t) = \pi_{G-d_1}(s,t)$, and this value can be queried in the static DSO. If $d_2 \in \pi_{G-d_1}(s,t)$, we know the value equals $\pi_{(G-d_1)-d_2}(s,t)$ from the dynamic DSO in graph $G - d_1$, which is obtained in our algorithm.

Time analysis. The incremental DSO can be viewed as an offline fully dynamic DSO by Theorem 10. Therefore, the update time is $\tilde{O}(n^2)$ for each edge e_i. The number of edges on a shortest path tree is $O(n)$, so the total update time is $\tilde{O}(n^3)$. The number of distances we query is $O(n^3)$, each with query time $\tilde{O}(1)$, so the total query time is also $\tilde{O}(n^3)$. We also retrieve all 1-fault single-source replacement paths $\pi_{G-e_i}(s, t_j)$ in the static DSO, and the time is also bounded by $\tilde{O}(n^3)$.

Extend to f-fault SSRP for $f \geq 2$. As the fFRP reduction in [30], when we have an f-fault SSRP algorithm in $O(n^s)$ time, we can construct an $(f+1)$-fault SSRP algorithm in $O(n^{s+1})$ time, by computing the f-fault SSRP in graph $G - e$ for every e in the shortest path tree from s. Thus, the undirected 2-fault SSRP algorithm in $\tilde{O}(n^3)$ time can be extended to undirected f-fault SSRP algorithm in $\tilde{O}(n^{f+1})$ time for all $f \geq 2$, which is almost optimal.

References

1. Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2039, 2025. doi:10.1137/1.9781611978322.63.
2. Noga Alon, Shiri Chechik, and Sarel Cohen. Deterministic combinatorial replacement paths and distance sensitivity oracles. In *International Colloquium on Automata, Languages and Programming*, 2019. URL: https://api.semanticscholar.org/CorpusID:159041702.
3. Aaron Bernstein and David Karger. Improved distance sensitivity oracles via random sampling. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 34–43, USA, 2008. Society for Industrial and Applied Mathematics. URL: https://dl.acm.org/doi/10.5555/1347082.1347087.
4. Aaron Bernstein and David Karger. A nearly optimal oracle for avoiding failed vertices and edges. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, pages 101–110, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1536414.1536431.
5. Greg Bodwin, Fabrizio Grandoni, Merav Parter, and Virginia Vassilevska Williams. Preserving Distances in Very Faulty Graphs. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 73:1–73:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2017.73.
6. Anat Bremler-Barr, Yehuda Afek, Haim Kaplan, Edith Cohen, and Michael Merritt. Restoration by path concatenation: fast recovery of MPLS paths. In *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*, PODC '01, pages 43–52, New York, NY, USA, 2001. Association for Computing Machinery. doi:10.1145/383962.383980.
7. Shiri Chechik and Sarel Cohen. Near optimal algorithms for the single source replacement paths problem. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2090–2109. SIAM, 2019. doi:10.1137/1.9781611975482.126.
8. Shiri Chechik and Ofer Magen. Near Optimal Algorithm for the Directed Single Source Replacement Paths Problem. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 81:1–81:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2020.81.
9. Shucheng Chi, Ran Duan, Benyu Wang, and Tianle Xie. Undirected 3-fault replacement path in nearly cubic time, 2024. doi:10.48550/arXiv.2411.18312.

10 Camil Demetrescu and Giuseppe F. Italiano. A new approach to dynamic all pairs shortest paths. *J. ACM*, 51(6):968–992, November 2004. doi:10.1145/1039488.1039492.

11 Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37:1299–1318, 2008. doi:10.1137/S0097539705429847.

12 Dipan Dey and Manoj Gupta. Near Optimal Algorithm for Fault Tolerant Distance Oracle and Single Source Replacement Path Problem. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms (ESA 2022)*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2022.42.

13 Dipan Dey and Manoj Gupta. Nearly optimal fault tolerant distance oracle. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, pages 944–955, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3618260.3649697.

14 E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. doi:10.1007/BF01386390.

15 R. Duan, J. Mao, X. Shu, and L. Yin. A randomized algorithm for single-source shortest path on undirected real-weighted graphs. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 484–492, Los Alamitos, CA, USA, November 2023. IEEE Computer Society. doi:10.1109/FOCS57990.2023.00035.

16 R. Duan, H. Wu, and R. Zhou. Faster matrix multiplication via asymmetric hashing. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2129–2138, Los Alamitos, CA, USA, November 2023. IEEE Computer Society. doi:10.1109/FOCS57990.2023.00130.

17 Ran Duan, Jiayi Mao, Xiao Mao, Xinkai Shu, and Longhui Yin. Breaking the sorting barrier for directed single-source shortest paths, 2025. arXiv:2504.17033.

18 Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 506–515, USA, 2009. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973068.56.

19 Ran Duan and Hanlin Ren. Maintaining exact distances under multiple edge failures. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 1093–1101, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520002.

20 Ran Duan and Tianyi Zhang. Improved distance sensitivity oracles via tree partitioning. In Faith Ellen, Antonina Kolokolova, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures*, pages 349–360, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-62127-2_30.

21 Zvi Gotthilf and Moshe Lewenstein. Improved algorithms for the k simple shortest paths and the replacement paths problems. *Information Processing Letters*, 109(7):352–355, 2009. doi:10.1016/j.ipl.2008.12.015.

22 Fabrizio Grandoni and Virginia Vassilevska Williams. Improved distance sensitivity oracles via fast single-source replacement paths. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 748–757, 2012. doi:10.1109/FOCS.2012.17.

23 Fabrizio Grandoni and Virginia Vassilevska Williams. Faster replacement paths and distance sensitivity oracles. *ACM Trans. Algorithms*, 16(1), December 2019. doi:10.1145/3365835.

24 Yong Gu and Hanlin Ren. Constructing a Distance Sensitivity Oracle in $O(n^{2.5794}M)$ Time. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 76:1–76:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.76.

25 Yuzhou Gu, Adam Polak, Virginia Vassilevska Williams, and Yinzhan Xu. Faster Monotone Min-Plus Product, Range Mode, and Single Source Replacement Paths. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 75:1–75:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.75.

26 Enrico Nardelli, Guido Proietti, and Peter Widmayer. A faster computation of the most vital edge of a shortest path. *Information Processing Letters*, 79(2):81–85, 2001. doi:10.1016/S0020-0190(00)00175-7.

27 Binghui Peng and Aviad Rubinstein. Fully-dynamic-to-incremental reductions with known deletion order (eg sliding window). In *Symposium on Simplicity in Algorithms (SOSA)*, pages 261–271. SIAM, 2023. doi:10.1137/1.9781611977585.ch24.

28 Hanlin Ren. Improved distance sensitivity oracles with subcubic preprocessing time. *Journal of Computer and System Sciences*, 123:159–170, 2022. doi:10.1016/j.jcss.2021.08.005.

29 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 664–673, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591811.

30 V. Williams, E. Woldeghebriel, and Y. Xu. Algorithms and lower bounds for replacement paths under multiple edge failure. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 907–918, Los Alamitos, CA, USA, November 2022. IEEE Computer Society. doi:10.1109/FOCS54457.2022.00090.

31 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5), August 2018. doi:10.1145/3186893.

32 Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3792–3835. SIAM, 2024. doi:10.1137/1.9781611977912.134.