Undirected Replacement Paths: Dual Fault Reduces to Single Source

Jakob Nogler
Massachusetts Institute of Technology, Cambridge, MA, USA

Virginia Vassilevska Williams
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

Given a graph and two vertices s and t, the *Replacement Path* Problem (RP) is to compute for every edge e, the distance between s and t when e is removed. There are two natural extensions to RP:

- **Single Source Replacement Paths (SSRP):** Given a graph $\mathbf{G}$ and a source node s, compute for every vertex v and every edge e the s-v distance in $\mathbf{G} \setminus e$. That is, we do not fix the target anymore.
- **2-Fault Replacement Paths (2-FRP):** Given a graph $\mathbf{G}$ and two nodes s and t, compute for every pair of edges e, e' the s-t distance in $\mathbf{G} \setminus e, e'$. That is, there are two failures instead of one.

Previously, there was no known formal reduction between SSRP and 2-FRP. It seemed plausible that 2-FRP would be computationally harder because there are no settings where 2-FRP admits a faster algorithm than SSRP. In directed unweighted graphs there is a provable gap in complexity, and in undirected graphs many of the known 2-FRP algorithms in a variety of settings are much slower than those for SSRP in the same setting.

The main contribution of this paper is a tight reduction from undirected 2-FRP to undirected SSRP, showing that contrary to prior intuition, 2-FRP is not harder than SSRP. As our reduction is weight-preserving, we get new algorithms for 2-FRP that match the best-known runtimes for SSRP:

(a) $\tilde{\mathcal{O}}(Mn^{\omega})$ for weights in $[1..M]$ [Grandoni and Vassilevska Williams, FOCS 2012 & TALG 2019], improving upon $\mathcal{O}(Mn^{2.87})$ [Chechik, Zhang, ICALP 2024];

(b) $n^3/2^{\Omega(\sqrt{\log n})}$ for weights in $[1..\text{poly}(n)]$ [Grandoni and Vassilevska Williams, FOCS 2012 & TALG 2019], improving over the previous $n^3 \,\text{polylog}(n)$ running time [Vassilevska W., Woldeghebriel and Xu, FOCS 2022];

(c) $\tilde{\mathcal{O}}(mn^{1/2} + n^2)$ combinatorial time for unweighted graphs [Chechik and Cohen, SODA 2019], and more generally for rational weights in $[1, 2]$ [Chechik and Magen, ICALP 2020], improving upon $\tilde{\mathcal{O}}(n^{3-1/18})$ [Chechik, Zhang ICALP 2024].

We complement these upper bounds with tight lower bounds under fine-grained hypotheses.

2012 ACM Subject Classification Theory of computation → Shortest paths; Theory of computation → Divide and conquer

Keywords and phrases Single Source Replacement Paths, Dualt Fault Replacement Paths, Fine-Grained Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.144

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2605.02114 [34]

Funding *Jakob Nogler*: Supported by the Akamai Presidential Fellowship.
Virginia Vassilevska Williams: Supported by NSF Grant CCF-2330048, BSF Grant 2024233 and a Simons Investigator Award.

53rd International Colloquium on Automata, Languages, and Programming (ICALP 2026).
Editors: Sayan Bhattacharya, Danupon Nanongkai, Michael Benedikt, and Gabriele Puppis;
Article No. 144; pp. 144:1–144:16

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

1 Introduction

In the *Replacement Path* problem (RP) one is given a graph $\mathbf{G}$ and two fixed nodes s and t and is asked to compute for every possible failed edge e of $\mathbf{G}$, the distance between s and t in the subgraph $\mathbf{G} \setminus e$. Beyond its clear application for distance computation in error-prone networks, the problem serves as a critical algorithmic primitive in diverse domains: it can be used to identify segments in biological sequence alignments [14], to calculate Vickrey Prices in an auction on a distributed network [33, 27], and as a subproblem in an algorithm for computing the kth shortest simple s-t paths in both directed and undirected graphs [30, 45, 35, 37].

For undirected graphs, a near-linear, $\tilde{\mathcal{O}}(m)$[1] time algorithm was first proposed by Malik et al. [31] (with corrections in [38]), and later improved by Nardelli, Proietti, and Widmayer to $\mathcal{O}(m\alpha(n,m))$ [32]. Interestingly, RP is computationally harder in directed graphs: the general weighted version is equivalent to the All-Pairs Shortest Paths (APSP) problem, with reductions existing in both directions [21, 41].

The popular APSP Hypothesis from Fine-Grained Complexity (see e.g. [40]) states that APSP in n-node graphs requires $n^{3-o(1)}$ time in the word-RAM model of computation. Due to the equivalence, the APSP Hypothesis implies an $n^{3-o(1)}$ time lower bound for RP in directed graphs as well. Truly subcubic[2] time algorithms exist for graphs with small integer weights [37, 39, 43] and for the approximate version of the problem [8].

The two most studied generalizations of RP involve either increasing the number of failures or relaxing the target constraint. The first considers the failure of *two* edges simultaneously, while the second generalizes the target by requiring the computation of the distances from a fixed source s to all other vertices $v \in V(\mathbf{G})$ after the failure of each edge e. More formally, the two obtained problems read as follows.

Single Source Replacement Paths Problem (SSRP)
Input: A graph $\mathbf{G}$ and a node $s \in V(\mathbf{G})$.
Output: The distance $d_{\mathbf{G}\setminus e}(s,v)$ for each edge $e \in E(\mathbf{G})$ and node $v \in V(\mathbf{G})$.

Dual-Fault Replacement Path Problem (2-FRP)
Input: A graph $\mathbf{G}$ and nodes $s,t \in V(\mathbf{G})$.
Output: The distance $d_{\mathbf{G}\setminus e,e'}(s,t)$ for all $e,e' \in E$.

We remark that for both problems the output size is actually quadratic. In SSRP this is because for each $v \in V(\mathbf{G})$ we only care about $d_{\mathbf{G}\setminus e}(s,v)$ for e on the shortest path $\pi_{\mathbf{G}}(s,v)$ between s and v. Similarly, for 2-FRP we only need to consider e,e' on the shortest s-t path in $\mathbf{G}$ and $\mathbf{G} \setminus e$, respectively.

Next, we summarize our algorithmic understanding of SSRP and 2-FRP (see also Table 1).

Previous results on SSRP. In a general graph, SSRP admits a simple $\tilde{\mathcal{O}}(mn)$ time algorithm by solving SSSP in $\mathbf{G} \setminus e$ for each e in the shortest path tree of s in $\mathbf{G}$. This is optimal even in undirected graphs under the APSP hypothesis [15] (a previous $\Omega(nm)$ lower bound was already established in the path comparison model [28, 29]).

The first to study SSRP in depth were Grandoni and Vassilevska Williams [22, 23], who positioned the problem not only as a generalization of RP but also as a fundamental component for constructing faster *distance sensitivity oracles* (DSOs). They provided

[1] The $\tilde{\mathcal{O}}(\cdot)$ notation suppresses factors in $\mathcal{O}(\text{polylog}(n))$.
[2] A running time is truly subcubic if it is of the form $\mathcal{O}(n^{3-\varepsilon})$ for constant $\varepsilon > 0$.

Table 1 Upper and lower bounds for SSRP and 2-FRP under different graph settings before this work. Under the current bounds APSP can be solved in time $n^3/\exp(\Omega(\sqrt{\log n}))$ [44]. The last two rows refer to combinatorial algorithms. The cells colored in yellow are improved by this paper.

Setting	Weights	SSRP		2-FRP	
		Upper Bound	Lower Bound	Upper Bound	Lower Bound
undir.	$[1..n^c]$	APSP-time [23]	APSP-hard [16]	$\tilde{\mathcal{O}}(n^3)$ [42]	APSP-hard [26]
dir.	$[1..n^c]$	APSP-time [23]	APSP-hard [16]	$\tilde{\mathcal{O}}(n^3)$ [42]	APSP-hard [41]
dir.	$[-M..M]$	$\mathcal{O}(M^{0.81}n^{2.49})$[24]	?	$\mathcal{O}(Mn^{2.87})$ [17]	?
undir.	$[1..M]$	$\tilde{\mathcal{O}}(Mn^{\omega})$ [23]	?	$\mathcal{O}(Mn^{2.87})$ [17]	?
dir.	$[1..M]$	$\tilde{\mathcal{O}}(Mn^{\omega})$ [23]	?	$\mathcal{O}(Mn^{2.87})$ [17]	?
undir.	$\mathbb{Q}\cap[1,2]$	$\tilde{\mathcal{O}}(mn^{1/2}+n^2)$ [16]	$mn^{1/2-o(1)}$ [16]	$\tilde{\mathcal{O}}(n^3)$ [42]	?
dir.	$\mathbb{Q}\cap[1,2]$	$\tilde{\mathcal{O}}(mn^{1/2}+n^2)$ [16]	$mn^{1/2-o(1)}$ [16]	$\tilde{\mathcal{O}}(n^3)$ [42]	?
undir.	unweigh.	$\tilde{\mathcal{O}}(mn^{1/2}+n^2)$ [15]	$mn^{1/2-o(1)}$ [15]	$\mathcal{O}(n^{3-1/18})$[17]	?
dir.	unweigh.	$\tilde{\mathcal{O}}(mn^{1/2}+n^2)$ [16]	$mn^{1/2-o(1)}$ [16]	$\mathcal{O}(n^{3-1/18})$[17]	$n^{8/3-o(1)}$ [42]

algorithms running in $\tilde{\mathcal{O}}(M^{1/(4-\omega)}n^{2+1/(4-\omega)})$ time for directed graphs with weights in $[-M..M]$ and in $\tilde{\mathcal{O}}(Mn^{\omega})$ time for weights in $[1..M]$; the latter matches the runtime for the RP problem itself. Notably, [22, 23] demonstrates that SSRP in general graphs also reduces to APSP, establishing a computational equivalence between the two problems and, by extension, directed RP. Subsequent improvements for the $[-M..M]$ weight domain were later provided by [24] and [11].

Combinatorial algorithms for RP and SSRP have also been intensively studied. For RP in m-edge, n-node directed unweighted graphs, Roditty and Zwick [37] gave an $\tilde{\mathcal{O}}(m\sqrt{n})$ time algorithm and Vassilevska Williams and Williams [41] showed that this is optimal for "combinatorial"[3] algorithms unless the so called BMM Hypothesis is violated; the latter asserts that Boolean Matrix Multiplication (BMM) does not admit an $\mathcal{O}(n^{3-\varepsilon})$ time combinatorial algorithm for any $\varepsilon > 0$. Chechik and Cohen [15] showed that roughly the same running time and conditional lower bounds hold for undirected unweighted SSRP by giving an $\tilde{\mathcal{O}}(\sqrt{n}m+n^2)$ time combinatorial algorithm and a matching combinatorial lower bound under the BMM Hypothesis. Then [16] obtained the same running time for directed unweighted graphs and even for directed graphs with rational weights upper bounded by a constant (derandomized in [11]) and showed that the latter version requires $mn^{0.5-o(1)}$ time even under the APSP hypothesis. Lastly, SSRP has also been studied in specialized graph classes, such as planar graphs [4], and in the approximate setting [3, 10, 2, 11, 25]. (In the latter, to bypass the quadratic output lower bound, SSRP is often framed as an oracle problem.)

Previous results on 2-FRP. Early on, it was conjectured that 2-FRP might admit an $\tilde{\mathcal{O}}(n^3)$ time algorithm [9], matching the complexity of the single-failure case in directed graphs. The existence of such an algorithm was eventually established nearly twenty years later by Vassilevska Williams, Woldeghebriel, and Xu [42]. Interestingly, in undirected graphs, 2-FRP reduces to APSP [26, 19]. Consequently, the two-failure case in both undirected and directed graphs shares the same cubic-time complexity. This is in stark contrast to the single-failure setting, where the problems exhibit significantly different computational profiles.

Similar to the developments in SSRP, research has also focused on small-weight and approximate algorithms for 2-FRP. In [42], an (algebraic) algorithm with a running time of $\mathcal{O}(M^{2/3}n^{2.9153})$ was introduced for weights in $[-M..M]$. The authors also obtained a

[3] Not a well-defined notion but used in the literature anyway.

conditional lower bound for combinatorial algorithms for unweighted directed graphs under the BMM Hypothesis. However, this lower bound of $n^{8/3-o(1)}$, was subcubic and thus left open the possibility that a subcubic combinatorial algorithm is possible. Subsequently, Chechik and Zhang [17] were able to obtain the first truly subcubic combinatorial algorithm for unweighted 2-FRP, running in $\mathcal{O}(n^{3-1/18})$ time. They also obtained a slightly improved $\mathcal{O}(Mn^{2.8716})$ (algebraic) algorithm for weights in $[-M\,..\,M]$.

Finally, regarding approximation schemes, Chechik and Zhang [18] also demonstrated that a $(1+\varepsilon)$-approximation for 2-FRP can be achieved in near-optimal $\tilde{\mathcal{O}}_\varepsilon(n^2)$ time.

SSRP vs 2-FRP. The preceding literature suggests that 2-FRP is inherently more difficult than SSRP. There are no settings where the former admits a faster algorithm than the latter; even in the undirected case – specifically for small edge weights and combinatorial algorithms in unweighted graphs – a clear complexity gap exists. In unweighted directed graphs, this gap is even provable for combinatorial algorithms: directed SSRP admits an $\tilde{\mathcal{O}}(mn^{1/2}+n^2)$-time combinatorial algorithm [16], whereas directed 2-FRP is subject to a conditional combinatorial lower bound of $n^{8/3-o(1)}$ [42].

We consider the following tantalizing question:

Is 2-FRP computationally harder than SSRP in undirected graphs?
Is it harder than APSP in arbitrary integer weighted graphs?

Our Results

In this paper, we demonstrate that for undirected graphs the opposite of our intuition holds: we show that the seemingly more difficult 2-FRP problem can, in fact, be reduced to the seemingly easier SSRP problem.

Our reduction considers two cases: whether both failed edges e, e' lie on the s-t shortest path or if only one does (note that at least one must always lie on this path). For the former case, we show that it can be handled in the same time as SSRP on graphs with the same edge weight set.

▶ **Theorem 1.** *Let $\mathsf{T}_{\mathsf{SSRP}}(n,m,W)$ be the time needed to compute SSRP in an undirected graph with n nodes, m edges, and positive weight set W. If there is a constant $0<\rho\leq 1$ such that*

$$\mathsf{T}_{\mathsf{SSRP}}(n_1,m_1,W)+\mathsf{T}_{\mathsf{SSRP}}(n_2,m_2,W)\leq\rho\cdot\mathsf{T}_{\mathsf{SSRP}}(n,m,W)$$

for all $n_1+n_2\leq n+1$, $m_1+m_2\leq m$ and $n_1,n_2\leq 2n/3$, then we can compute all values $d_{\mathbf{G}\setminus e,e'}(s,t)$ such that $e,e'\in\pi_{\mathbf{G}}(s,t)$ in time $\mathcal{O}(\mathsf{T}_{\mathsf{SSRP}}(n,m,W)+n^2)$ if $\rho<1$, and $\mathcal{O}(\mathsf{T}_{\mathsf{SSRP}}(n,m,W)\log^2 n+n^2)$ otherwise.

Note that in Theorem 1, the assumptions on $\mathsf{T}_{\mathsf{SSRP}}$ are more than reasonable and align with standard algorithmic running times. For instance, any function of the form $n^\alpha m$ for $\alpha>0$, or $n^\alpha m^\beta$ for $\alpha\geq 1$ and $\beta>0$, satisfies these conditions with $\rho<1$; similarly, any function of the form m^α for $\alpha\geq 1$ satisfies it with $\rho=1$.

For the latter case when $e'\notin\pi_{\mathbf{G}}(s,t)$, we obtain a reduction in a stronger sense. Specifically, we show that it suffices to compute SSRP on the *same graph* from both s and t, followed by $\tilde{\mathcal{O}}(n^2)$ additional work.

▶ **Theorem 2.** *Let $\mathbf{G}$ be an undirected graph with n nodes and positive weight set, and let $s,t\in\mathbf{G}$ be two nodes. Further, let T be the time needed to solve SSRP from s and t. Then, we can compute all values $d_{\mathbf{G}\setminus e,e'}(s,t)$ such that $e\in\pi_{\mathbf{G}}(s,t)$ and $e'\notin\pi_{\mathbf{G}}(s,t)$ in time $\mathsf{T}+\mathcal{O}(n^2\log^2 n)$.*

Our reduction also yields faster algorithms; specifically, it allows us to take the upper bounds for SSRP from each row for undirected graphs in Table 1 and apply them directly to 2-FRP. More specifically, combining Theorem 2, Theorem 1 with [22, 23] and [15, 16][4] (and the fastest APSP algorithm to date [44]), we obtain the following faster algorithms for 2-FRP in undirected graphs.

▶ **Corollary 3.** *In undirected graphs there are the following algorithms for 2-FRP:*
(a) *An $\tilde{\mathcal{O}}(Mn^{\omega})$-time algorithm for weights in $[1..M]$;*
(b) *An $n^3/2^{\Omega(\sqrt{\log n})}$-time algorithm for weights in $[1..\text{poly}(n)]$; and*
(c) *An $\tilde{\mathcal{O}}(mn^{1/2}+n^2)$-time (combinatorial) algorithm for rational weights in $[1,2]$;*

As a side-result of our reduction, we establish that undirected 2-FRP is *computationally equivalent* to the following other problems on general weighted graphs: directed RP, directed or undirected SSRP, and directed or undirected APSP. In other words, tolerating a single failure is computationally equivalent to tolerating two, provided we make the underlying network undirected.

Lower bounds. We complement the algorithmic results presented in Corollary 3 with corresponding lower bounds. The tightness of Corollary 3(b) is immediate; by combining the results of [26, 19] with our reduction, we establish the APSP-equivalence of 2-FRP for weights in $[1..\text{poly}(n)]$.

Regarding Corollary 3(a) and Corollary 3(c), we provide lower bounds based on the BMM and APSP hypotheses for the regime $m=\Omega(n^{3/2})$, where the output-size lower bound is non-dominant.

▶ **Lemma 12.** *Consider the problem of solving 2-FRP in a graph with n nodes and m edges. If $m=\Omega(n^{3/2})$, then for any $\varepsilon>0$, there is no*
(i) *$\mathcal{O}(mn^{1/2-\varepsilon})$ combinatorial algorithm for unweighted graphs, unless the BMM hypothesis is false;*
(ii) *$\mathcal{O}(mn^{1/2-\varepsilon})$ algorithm for graphs with rational weights in $[1,2]$, unless the APSP hypothesis is false;*

Interestingly, our reduction proceeds via a version of triangle detection in sparse graphs with unbalanced partition sizes; for certain values of m, this could imply a non-combinatorial lower bound even for the unweighted case. (See Section 2.1 for a more detailed discussion.)

A note on the structural connection between SSRP and 2-FRP. From a *structural* perspective, existing results already demonstrate a relationship between dual-failure fixed-endpoints and single-source single-failure instances in undirected graphs. To see this, we recall the Restoration Lemma, a classic result by Afek et al. [13] (recently improved by [12]).

▶ **Theorem 4** (Restoration Lemma [13])**.** *Given an undirected graph $\mathbf{G}$, any shortest path $\pi_{\mathbf{G}\setminus F}(s,t)$ where $F\subseteq E(\mathbf{G})$ and $|F|=f$ can be decomposed into $f+1$ shortest paths in $\mathbf{G}$ and f edges in-between.*

This lemma implies that any shortest path $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ can be decomposed into $\pi_{\mathbf{G}\setminus e}(s,u)\circ\{u,v\}\circ\pi_{\mathbf{G}\setminus e}(v,t)$ for some $u,v\in V(\mathbf{G})$. This establishes a clear structural link between dual-failure scenarios with source/target s,t and single-failure instances with source s and t and

[4] Works [22, 23, 16] are formulated on directed graphs, but it is possible to verify that the algorithms work on undirected graphs as well.

arbitrary target. However, it remains highly unclear how to use this lemma *algorithmically*; the search for the intermediate vertices u, v ranges over n possible vertices each, and both must ensure that the subpaths $\pi_{\mathbf{G}\setminus e}(s,u)$ and $\pi_{\mathbf{G}\setminus e}(v,t)$ do not contain the second failed edge e'. In fact, our reduction does not rely on this lemma; instead, for most cases, we manage to use a simpler decomposition consisting of a shortest path in $\mathbf{G}$, an individual edge, and a shortest path in $\mathbf{G}\setminus e$.

Open questions. We pose the following two questions connected to this work:

- Is a reduction from 2-FRP to SSRP possible for directed graphs, at least in a weaker form?
- In the directed case, if no reduction from 2-FRP to SSRP can be found, can we at least demonstrate that 2-FRP reduces to APSP, thereby establishing that one and two failures are computationally equivalent in general graphs?

2 Overview

In this section, we provide a concise overview of our results; complete proofs are available in the full version of this work [34].

This overview is organized into three parts. First, we present our reduction across two (sub)sections: the first addresses the case where both failed edges, e and e', lie on the shortest path $\pi_{\mathbf{G}}(s,t)$ (Theorem 1), while the second covers the case where only e belongs to $\pi_{\mathbf{G}}(s,t)$ (Theorem 2). Finally, we conclude by presenting our lower bounds.

Notation is intended to be clear from the context; however, the reader may refer to [34] for further clarification.

We begin with an overview of the reduction. Throughout this discussion, for simplicity, we assume $\mathbf{G}$ has unique shortest paths (with the important property that subpaths of shortest paths are themselves shortest paths). Furthermore, we assume the availability of an algorithm for computing SSRP on any graph using the same weight set as $\mathbf{G}$.

Case $e', e \in \pi_{\mathbf{G}}(s,t)$

To compute $d_{\mathbf{G}}(s,t)$ for all $e, e' \in \pi_{\mathbf{G}}(s,t)$, say we adopt a divide-et-impera approach. Then, the most natural thing is to attempt the following strategy: first select a midpoint $c \in \pi_{\mathbf{G}}(s,t)$ (see Figure 1a) and compute $d_{\mathbf{G}\setminus e,e'}(s,t)$ for all $e \in \pi_{\mathbf{G}}(s,c)$ and $e' \in \pi_{\mathbf{G}}(c,t)$ (here our SSRP subroutine should prove helpful). We then construct two auxiliary graphs, $\mathbf{A}$ and $\mathbf{B}$ (sharing the same weight set as $\mathbf{G}$), and recurse to compute $d_{\mathbf{G}\setminus e,e'}(s,t)$ for all $e, e' \in \pi_{\mathbf{G}}(s,c)$ and $e, e' \in \pi_{\mathbf{G}}(c,t)$, recursively.

Computing $d_{\mathbf{G}\setminus e,e'}(s,t)$ for all $e \in \pi_{\mathbf{G}}(s,c)$ and $e' \in \pi_{\mathbf{G}}(c,t)$. This turns out not to be the hard part. Let $\mathbf{L}$ and $\mathbf{R}$ be $\mathbf{G}$ where we remove all edges of $\pi_{\mathbf{G}}(c,t)$ and $\pi_{\mathbf{G}}(s,c)$, respectively (see Figure 1b and Figure 1c). Suppose we compute SSRP in $\mathbf{L}$ and $\mathbf{R}$ from s and t, respectively. Order the nodes on $\pi_{\mathbf{G}}(s,t)$ from the closest to s to the farthest, letting $\preceq$ be the preceding relation. For edges $e = \{u,v\} \in \pi_{\mathbf{G}}(s,c)$ and $e' = \{u',v'\} \in \pi_{\mathbf{G}}(c,t)$ such that u and u' are closer to s than v and v', respectively, we claim that the desired value $d_{\mathbf{G}\setminus e,e'}(s,t)$ equals:

$$d_{\mathbf{G}\setminus e,e'}(s,t) = \min \left\{ \begin{array}{l} \min\limits_{v' \preceq z \preceq t} \{d_{\mathbf{L}\setminus e}(s,z) + d_{\mathbf{G}}(z,t)\}, \\ \min\limits_{s \preceq z \preceq u} \{d_{\mathbf{G}}(s,z) + d_{\mathbf{R}\setminus e'}(z,t)\}, \\ d_{\mathbf{G}\setminus e}(s,c) + d_{\mathbf{G}\setminus e'}(c,t) \end{array} \right\}. \tag{1}$$

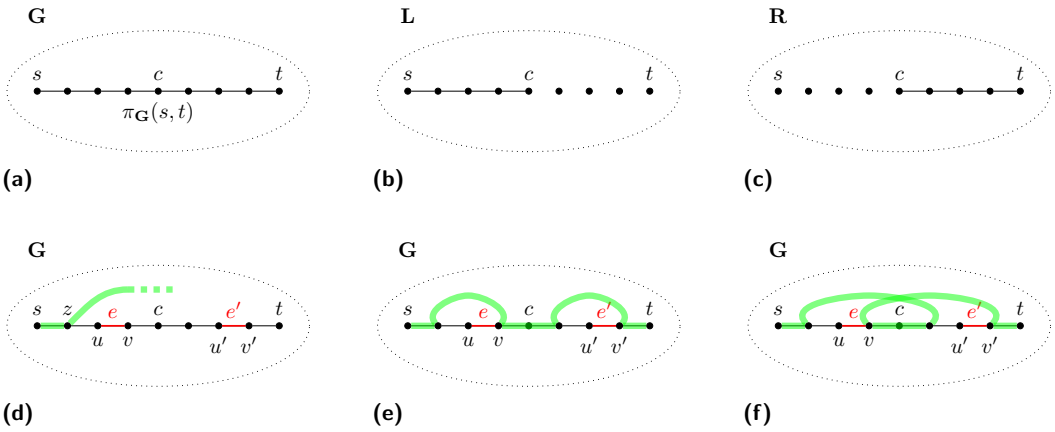

Figure 1 Some visualizations of the introduced notation when computing $d_{\mathbf{G}\setminus e,e'}(s,t)$.

It turns out that by using some simple tricks, Equation (1) can be computed in $\mathcal{O}(n^2)$ overall time for each $e \in \pi_{\mathbf{G}}(s,c)$ and $e' \in \pi_{\mathbf{G}}(c,t)$. We defer the discussion to the full proof.

To provide intuition for the correctness, we explain why Equation (1) is upper bounded by $d_{\mathbf{G}\setminus e,e'}(s,t)$. To see this, we perform a case distinction on $\pi_{\mathbf{G}\setminus e,e'}(s,t)$:

- $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ does not use any vertex $x \in \pi_{\mathbf{G}}(s,t)$ such that $v \preceq x \preceq c$. Then, the last node $z \in \pi_{\mathbf{G}}(s,t)$ visited by $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ must satisfy $z \preceq u$ (see Figure 1d). In this case, $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ between s and z equals $\pi_{\mathbf{G}}(s,z)$, and between z and t, where no other edge of $\pi_{\mathbf{G}}(s,c)$ is visited, it must coincide with $\pi_{\mathbf{R}\setminus e'}(z,t)$. We conclude that $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ is captured by the second term of Equation (1).
- $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ does not use any vertex $y \in \pi_{\mathbf{G}}(s,t)$ such that $c \preceq y \preceq u'$. Using a symmetric argument to the one above, this case is captured by the first term of Equation (1).
- $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ uses a vertex $x \in \pi_{\mathbf{G}}(s,t)$ such that $v \preceq x \preceq c$ and a vertex $y \in \pi_{\mathbf{G}}(s,t)$ such that $c \preceq y \preceq u'$ (refer to Figure 1e and Figure 1f to see the two possible ways $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ might look in this case). Since no edge on $\pi_{\mathbf{G}}(s,t)$ between x and y equals e or e', the path $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ between x and y equals $\pi_{\mathbf{G}}(x,y)$ and, in particular, passes through c. Since $d_{\mathbf{G}\setminus e}(s,c)$ and $d_{\mathbf{G}\setminus e'}(c,t)$ are always less than or equal to $d_{\mathbf{G}\setminus e,e'}(s,c)$ and $d_{\mathbf{G}\setminus e,e'}(c,t)$, the third term of Equation (1) is upper bounded by $d_{\mathbf{G}\setminus e,e'}(s,t)$.

It remains to show why Equation (1) is lower bounded by $d_{\mathbf{G}\setminus e,e'}(s,t)$. It is not difficult to see that the first and second terms are lower bounded by $d_{\mathbf{G}\setminus e,e'}(s,t)$ because the described paths never use e or e'. For the third term, the complete argument for which we omit, we essentially need to argue that if $\pi_{\mathbf{G}\setminus e}(s,c)$ or $\pi_{\mathbf{G}\setminus e'}(c,t)$ uses e' or e, then there exists a shorter option for $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ different from the path described by those terms and captured by the other first two terms.

Constructing A and B. Ideally, we would like to pick c and construct $\mathbf{A}$ and $\mathbf{B}$ such that each contains roughly half of the nodes of $\mathbf{G}$. If we pick c to be roughly in the middle of $\pi_{\mathbf{G}}(s,t)$ (the exact definition of which we leave unspecified for now), a natural candidate for $\mathbf{A}$ would be the subtree rooted at c within the shortest path tree rooted at t. By picking $\mathbf{A}$ in this manner, we obtain the useful property that for any $e, e' \in \pi_{\mathbf{G}}(s,c)$, as soon as $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ visits the first vertex $z \notin \mathbf{A}$, the path $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ from z to t coincides with $\pi_{\mathbf{G}}(z,t)$, as this path contains neither e nor e'. By further adding to $\mathbf{A}$ an edge from c to every node $x \in \mathbf{A}$ with weight $\min_{z \notin \mathbf{A}}\{w(x,z) + d_{\mathbf{G}}(z,t)\} - d_{\mathbf{G}}(c,t)$, we have $d_{\mathbf{A}\setminus e,e'}(s,c) + d_{\mathbf{G}}(c,t) = d_{\mathbf{G}\setminus e,e'}(s,t)$ because such new edge captures the segment $\pi_{\mathbf{G}}(z,t)$ and the edge $\{x,z\}$ before z on $\pi_{\mathbf{G}\setminus e,e'}(s,t)$.

This last construction is nice because it lets us condense $\mathbf{G}$ into $\mathbf{A}$ that acts well for the purpose of computing $d_{\mathbf{G}\setminus e,e'}(s,t)$ for $e,e' \in \pi_{\mathbf{G}}(s,c)$. However, the construction works against the objective of creating a universal reduction that preserves edge weights: The newly added edge might be of increased edge weight and $\mathbf{A}$ would not have the same edge set as $\mathbf{G}$! Generally, when condensing $\mathbf{G}$ into $\mathbf{A}$, we should always expect to encounter this behavior, regardless of how we choose the smaller subgraph $\mathbf{A}$.

A technical contribution to SSRP as workaround. As a workaround, we establish that the added weights do not, in fact, impact our capability to compute SSRP on $\mathbf{A}$. Formally, let $\mathbf{H}$ be a graph with a positive weight set W and let s denote a vertex. We define *short-cut weights* w_s as weights that satisfy $d_{\mathbf{H}}(s,x) \leq w_s(x)$ for every x, and the *augmented graph* as the result of adding edges $\{s,x\}$ with weights $w_s(x)$ to $\mathbf{H}$. (Clearly, $\mathbf{A}$ is captured by this construction because $\min_{z\notin\mathbf{A}}\{w_{\mathbf{G}}(x,z)+d_{\mathbf{G}}(z,t)\} \geq d_{\mathbf{G}}(x,c)+d_{\mathbf{G}}(c,t)$ for each $x \in \mathbf{A}$.) We show that SSRP can be solved in the same time complexity even with such weights.

▶ **Theorem 5.** *Let $\mathbf{G}$ be an undirected (resp. directed) n-node graph with a set of positive weights W. Let $s \in \mathbf{G}$ be a node, and let w_s denote shortcut weights w.r.t. s (where $w_s(x)$ is not necessarily in W). Furthermore, let $\tilde{\mathbf{G}}$ be the graph $\mathbf{G}$ augmented by w_s. Then, for $\mathsf{T}_{SSRP}(n,m,W)$ and ρ as in Theorem 1, we can solve SSRP in $\tilde{\mathbf{G}}$ with source s in time $\mathcal{O}(\mathsf{T}_{SSRP}(n,m,W)+n^2)$ if $\rho<1$, and $\mathcal{O}(\mathsf{T}_{SSRP}(n,m,W)\log n+n^2)$ otherwise.*

Theorem 5 serves not only as an essential component of our reduction but also as a mean to simplify existing SSRP results. Notably, these weights have appeared (either implicitly or explicitly) in some prior algorithms [22, 23, 16]. Consequently, we consider Theorem 5 to be a technical contribution of our work.

Getting back to the construction of A and B. Theorem 5 is the key to devise our divide-et-impera approach, yet the implementation remains non-trivial. Several technical details must still be addressed:

- The recursive scheme must operate on graphs $\mathbf{G}$ augmented by short-cut weights w_s and w_t (relative to s and t in $\mathbf{G}$), which are accumulated as the recursion descends.
- Defining $\mathbf{A}$ as previously established is insufficient; we must ensure that the shortest path from s to any $x \in \mathbf{A}$ is contained within $\mathbf{A}$ itself. Otherwise, shortcut edges might become shorter than the true distances from s. The current definition of $\mathbf{A}$ does not guarantee this property, but some similar variant of it will.
- We must select both $\mathbf{A}$ and $\mathbf{B}$ such that they are disjoint and partition the nodes roughly in half.
- We must prove we can compute SSRP with shortcut weights as fast as without shortcut weights not only within $\mathbf{A}$ (resp. $\mathbf{B}$) but also within $\mathbf{R}$ (resp. $\mathbf{L}$). Indeed, in $\mathbf{R}$ (resp. $\mathbf{L}$), upon removing a specific set of edges in $\pi_{\mathbf{G}}(c,t)$ (resp. $\pi_{\mathbf{G}}(s,c)$), the short-cut property $d_{\mathbf{H}}(s,x) \leq w_s(x)$ might not hold anymore; however, the removed edges are sufficiently structured that we can show that this is possible.

Case $e' \notin \pi_{\mathbf{G}}(s,t)$

While the case where $e' \in \pi_{\mathbf{G}}(s,t)$ was handled via a divide-et-impera approach, the case $e' \notin \pi_{\mathbf{G}}(s,t)$ requires extensive structural analysis and several case distinctions.

The underlying strategy is to characterize the path $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ as the concatenation of a shortest path $\pi_{\mathbf{G}}(s,x)$ in $\mathbf{G}$, a single edge $\{x,y\}$, and a path of the form $\pi_{\mathbf{G}\setminus e}(y,t)$ or $\pi_{\mathbf{G}\setminus e'}(y,t)$. While it is challenging to provide a holistic treatment of every case, we limit our discussion to describing the primary tools used to simplify $\pi_{\mathbf{G}\setminus e,e'}(s,t)$ to the desired form and provide one example of such simplification.

To this end, let us introduce some notation. Let $T_{\mathbf{G}}(s)$ denote the shortest path tree rooted at s in $\mathbf{G}$. For any node $x \in V(\mathbf{G})$, let $T_{\mathbf{G}}(s,x)$ denote the subtree of $T_{\mathbf{G}}(s)$ rooted at x. Lastly, given an edge $e \in T_{\mathbf{G}}(s)$, we let $T_{\mathbf{G}}(s,e)$ be the subtree of $T_{\mathbf{G}}(s)$ below e.

The first (well-known) observation (which also highlights one of the primary reasons our reduction is restricted to undirected graphs) that allows us to simplify ways paths are written is the following:

▶ **Proposition 6.** *Let $\mathbf{G}$ be a graph, and let $s,t \in V$ be nodes. Then, for any edge $e = \{u,v\} \in \pi_{\mathbf{G}}(s,t)$ such that u is closer to s than v we have $T_{\mathbf{G}}(s,v) \cap T_{\mathbf{G}}(t,u) = \varnothing$, which implies $\pi_{\mathbf{G}\setminus e}(x,t) = \pi_{\mathbf{G}}(x,t)$ for any $x \in T_{\mathbf{G}}(s,v)$.*

Proof. For the sake of contradiction, assume that there is $x \in T_{\mathbf{G}}(s,v) \cap T_{\mathbf{G}}(t,u)$. From $x \in T_{\mathbf{G}}(s,v)$, it follows that $e \notin \pi_{\mathbf{G}}(v,x)$. Moreover, from $x \in T_{\mathbf{G}}(t,u)$ follows $v \in \pi_{\mathbf{G}}(t,x)$ and $e \in \pi_{\mathbf{G}}(t,x)$, meaning $\pi_{\mathbf{G}}(t,x) = \pi_{\mathbf{G}}(t,v) \circ \pi_{\mathbf{G}}(v,x)$. But neither $\pi_{\mathbf{G}}(t,v)$ nor $\pi_{\mathbf{G}}(v,x)$ uses e, a contradiction. ◀

Defining $E_{\mathbf{G}}(A,B) \coloneqq E(\mathbf{G}) \cap (A \times B)$ for $A,B \subseteq V(\mathbf{G})$, the second observation is as follows:

▶ **Proposition 7.** *Let $\mathbf{G}$ be a graph, and consider two vertices $s,t \in \mathbf{G}$ and a set of edges $F \subseteq E(T_{\mathbf{G}}(s))$ such that all $T_{\mathbf{G}}(s,e)$ for different $e \in F$ are pairwise disjoint. Let $S \coloneqq \bigcup_{e\in F} V(T_{\mathbf{G}}(s,e))$ be the set of vertices below any edge of F.*

Then, provided that $t \in S$, there is $\{x,y\} \in E_{\mathbf{G}}(V(\mathbf{G}) \setminus S, S)$ with $\{x,y\} \notin F$ such that we can write

$$\pi_{\mathbf{G}\setminus F}(s,t) = \pi_{\mathbf{G}}(s,x) \circ \{x,y\} \circ \pi_{\mathbf{G}\setminus F}(y,t).$$

Moreover, for such x,y we have that $\pi_{\mathbf{G}\setminus F}(y,t)$ only contains vertices in S.

Proof. Let x be the last node visited by $\pi_{\mathbf{G}\setminus F}(s,t)$ that is contained in $V(\mathbf{G}) \setminus S$, and let y be the node immediately after. This means that $\pi_{\mathbf{G}\setminus F}(s,t) = \pi_{\mathbf{G}\setminus F}(s,x) \circ \{x,y\} \circ \pi_{\mathbf{G}\setminus F}(y,t)$, where $\pi_{\mathbf{G}\setminus F}(y,t)$ only contains vertices in S. From $x \in V(\mathbf{G}) \setminus S$ and from the assumption on $V(\mathbf{G}) \setminus S$ we get that $\pi_{\mathbf{G}\setminus F}(s,x) = \pi_{\mathbf{G}}(s,x)$, which concludes the proof. ◀

A simple showcase. As a (not too involved) example of how these observations are applied, we showcase the computations performed to determine $d_{\mathbf{G}\setminus e,e'}(s,t)$ for all e,e' such that $e \in \pi_{\mathbf{G}}(s,t)$, $e' \notin T_{\mathbf{G}}(s)$, and $e' \notin E_{\mathbf{G}}(T_{\mathbf{G}}(s,e), V(\mathbf{G}) \setminus T_{\mathbf{G}}(s,e))$.

Consider such arbitrary e,e'. Note that since $e' \notin T_{\mathbf{G}}(s)$, we have $T_{\mathbf{G}\setminus e'}(s) = T_{\mathbf{G}}(s)$. In particular, $T_{\mathbf{G}\setminus e'}(s,e) = T_{\mathbf{G}}(s,e)$. Thus, we can use Proposition 7 on $\mathbf{G} \setminus e', s, t$ and $F = \{e\}$ to get that there is $\{x,y\} \in E(\mathbf{G}) \setminus e'$ with $x \notin T_{\mathbf{G}}(s,e)$ and $y \in T_{\mathbf{G}}(s,e)$ such that:

$$\pi_{\mathbf{G}\setminus e,e'}(s,t) = \pi_{\mathbf{G}\setminus e'}(s,x) \circ \{x,y\} \circ \pi_{\mathbf{G}\setminus e,e'}(y,t). \tag{2}$$

Since $e' \notin T_{\mathbf{G}}(s) = T_{\mathbf{G}\setminus e}(s)$, we can simplify $\pi_{\mathbf{G}\setminus e'}(s,x) = \pi_{\mathbf{G}}(s,x)$. Moreover, from Proposition 6 follows that $\pi_{\mathbf{G}\setminus e,e'}(y,t) = \pi_{\mathbf{G}\setminus e'}(y,t)$. Altogether, Equation (2) simplifies to

$$\pi_{\mathbf{G}\setminus e,e'}(s,t) = \pi_{\mathbf{G}}(s,x) \circ \{x,y\} \circ \pi_{\mathbf{G}\setminus e'}(y,t). \tag{3}$$

Now that the three components of Equation (3) are simpler, we can focus on computing $d_{\mathbf{G}\setminus e,e'}(s,t)$ as

$$d_{\mathbf{G}\setminus e,e'}(s,t) = \min_{x\notin T_{\mathbf{G}}(s,e), y\in T_{\mathbf{G}}(s,e)} \Big\{ d_{\mathbf{G}}(s,x) + w_{\mathbf{G}}(x,y) + d_{\mathbf{G}\setminus e'}(y,t) \Big\}. \tag{4}$$

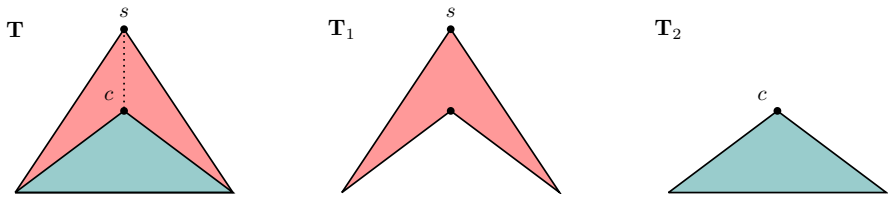

Figure 2 This figure shows how $\mathbf{T}$ is split into $\mathbf{T}_1$ and $\mathbf{T}_2$.

Some additional (algorithmic) tools. To this end, similarly to many works on replacement paths, we employ what is commonly known as *centroid decomposition*.

▶ **Lemma 8** (Folklore, Separator Lemma). *Given a tree $\mathbf{T}$ with n nodes rooted at node s, one can find in time a node c (called* centroid*) that separates $\mathbf{T}$ into 2 edge disjoint subtrees $\mathbf{T}_1, \mathbf{T}_2$ such that $E(\mathbf{T}_1) \cup E(\mathbf{T}_2) = E(\mathbf{T})$, $V(\mathbf{T}_1) \cap V(\mathbf{T}_2) = \{c\}$ and $n/3 \leq |V(\mathbf{T}_1)|, |V(\mathbf{T}_2)| \leq 2n/3$. (Refer to Figure 2 for a visualization.)*

Another tool we need are common range query data structures.

▶ **Lemma 9** (See [20, 7, 5, 6], Range Minimum Query). *Let $q_0, \ldots, q_{n-1}$ be n values. Then, we can construct in $\mathcal{O}(n)$ time a data structure that answers queries of the type "return $\min_{i \in [a\,..\,b)} q_{x_i}$" in $\mathcal{O}(1)$ time, for any $a, b \in [0\,..\,n]$ with $a < b$.*

▶ **Corollary 10** (Range Minimum Query on Trees). *Suppose for each vertex x of a n-node tree we are given a value q_x. Then, there is a data structure that can be constructed in time $\mathcal{O}(n)$ and answers in time $\mathcal{O}(1)$ the following queries: return $\min_{x \in X} q_x$, where X is the union and/or intersection of constantly many subtrees of $\mathbf{T}$.*

Proof. We order the nodes of $\mathbf{T}$ according to a pre-order traversal, obtaining the sequence $x_0, \ldots, x_{n-1}$. The nodes of any subtree in $\mathbf{T}$ correspond precisely to a set $\{x_i : i \in [a\,..\,b)\}$ for some interval $[a\,..\,b)$. This implies that any set formed by the union and intersection of a constant number of subtrees in $\mathbf{T}$ also corresponds to the union of a constant number of such intervals. Thus, any query for the data structure of Corollary 10 can be converted into a constant number of queries to the aforementioned data structure. The final result is then the minimum of the values returned by these queries. ◀

Getting back to Equation (4). We want to compute $d_{\mathbf{G}\setminus e,e'}(s,t)$ for all e, e' as described above via Equation (4). In this presentation, we will use one fact that we prove later in the paper: there exists an (efficiently computable) set $F \subseteq E(\mathbf{G})$ of size $|F| = \mathcal{O}(n)$ that collects all e' (across also different e) falling into this case.

We solve the problem recursively. In the beginning, $\mathbf{T} \coloneqq T_{\mathbf{G}}(s)$. Our goal is to compute the following expression for every $e \in E(\mathbf{T}) \cap \pi_{\mathbf{G}}(s,t)$ and $e' \in F$:

$$\min_{x \in \mathbf{T} \setminus T_{\mathbf{G}}(s,e), y \in T_{\mathbf{G}}(s,e)} \left\{ d_{\mathbf{G}}(s,x) + w_{\mathbf{G}}(x,y) + d_{\mathbf{G}\setminus e'}(y,t) \right\} \tag{5}$$

In each level of the recursion, we split $\mathbf{T}$ into $\mathbf{T}_1$ and $\mathbf{T}_2$ using Lemma 8, and find the corresponding centroid c. Let $n_{\mathbf{T}}, n_{\mathbf{T}_1}, n_{\mathbf{T}_2}$ be number of nodes in the corresponding tree.

For $e \in E(\mathbf{T}_1) \cap \pi_{\mathbf{G}}(s,t)$ and $e' \in F$, we rewrite Equation (5) as

$$\min \left\{ \begin{array}{l} \min\limits_{\substack{x \in \mathbf{T}_1 \setminus T_{\mathbf{G}}(s,e) \\ y \in T_{\mathbf{G}}(s,e)}} \left\{ d_{\mathbf{G}}(s,x) + w_{\mathbf{G}}(x,y) + d_{\mathbf{G}\setminus e'}(y,t) \right\}, \\ \min\limits_{\substack{x \in \mathbf{T}_2 \\ y \in T_{\mathbf{G}}(s,e)}} \left\{ d_{\mathbf{G}}(s,x) + w_{\mathbf{G}}(x,y) + d_{\mathbf{G}\setminus e'}(y,t) \right\} \end{array} \right\},$$

where in the second term we use that if $x \in \mathbf{T}_2$ and $e \in \mathbf{T}_1$, then $x \notin T_{\mathbf{G}}(s,e)$. (notice that this is only true when e is not on the path from the root of $\mathbf{T}$ to c, however if e is on such path then the second term is non-existent, and we can ignore it.)

For the first term, we can recurse on $\mathbf{T}_1$. To compute the second term, we do the following:

- for each node $y \in \mathbf{G}$ we compute $h_y \coloneqq \min_{x \in \mathbf{T}_2}\{d_{\mathbf{G}}(s,x) + w_{\mathbf{G}}(x,y)\}$ in time $\mathcal{O}(n_{\mathbf{T}} \cdot n)$;
- we setup for each $e' \in F$ a data structure from Corollary 10 where y has the associated value $h_y + d_{\mathbf{G}\setminus e'}(y,t)$ in time $\mathcal{O}(|F| \cdot n) = \mathcal{O}(n^2)$.
- for each $e \in \mathbf{T}_1$ and $e' \in F$ we query $\max_{y \in T_{\mathbf{G}}(s,e)}\{h_y + d_{\mathbf{G}\setminus e'}(y,t)\}$ in time $\mathcal{O}(|F| \cdot n_{\mathbf{T}}) = \mathcal{O}(nn_{\mathbf{T}})$.

Similarly, when $e \in E(\mathbf{T}_2) \cap \pi_{\mathbf{G}}(s,t)$ we rewrite Equation (4) as

$$\min \left\{ \begin{array}{l} \min\limits_{\substack{x \in \mathbf{T}_2 \setminus T_{\mathbf{G}}(s,e) \\ y \in T_{\mathbf{G}}(s,e)}} \left\{ d_{\mathbf{G}}(s,x) + w_{\mathbf{G}}(x,y) + d_{\mathbf{G}\setminus e'}(y,t) \right\}, \\ \min\limits_{\substack{x \in \mathbf{T}_1 \\ y \in T_{\mathbf{G}}(s,e)}} \left\{ d_{\mathbf{G}}(s,x) + w_{\mathbf{G}}(x,y) + d_{\mathbf{G}\setminus e'}(y,t) \right\} \end{array} \right\},$$

where we use again $x \in \mathbf{T}_1$ and $e \in \mathbf{T}_2$ implies $x \notin T_{\mathbf{G}}(s,e)$. As before, we can compute the first term by recursing in $\mathbf{T}_2$ and the first by performing similar computations to above.

For the running time notice that first we need to compute SSRP from t in $\mathbf{G}$. Then, when the recursion arrives to a certain tree $\mathbf{T}$, we use time $\mathcal{O}(n_{\mathbf{T}} \cdot n)$ and we recurse on $\mathbf{T}_1$ and $\mathbf{T}_2$ that satisfy $(n_{\mathbf{T}_1} - 1) + (n_{\mathbf{T}_2} - 1) = (n_{\mathbf{T}} - 1)$. Since with every level of recursion $n_{\mathbf{T}}$ shrinks by a constant fraction, we get that the overall running time is $\mathcal{O}(n^2 \log n)$.

2.1 Lower Bounds

As a first step, we demonstrate how to encode a single 2-FRP instance into another instance of a more sparse triangle detection problem, formulated as follows:

Sparse $(\sqrt{n}, n, n)$-Triangle Detection
Input: A tripartite graph $\mathbf{G}$ with m edges s.t. $V(\mathbf{G}) = A \cup B \cup C$ with $|A| = \mathcal{O}(\sqrt{n})$, $|B|, |C| = \mathcal{O}(n)$.
Output: YES if there are $a, b, c \in A \times B \times C$ such that $(a,b), (b,c), (c,a) \in E(\mathbf{G})$, and NO otherwise.

In the Sparse $(\sqrt{n}, n, n)$-Minimum Weight Triangle Problem, the graph is additionally weighted and we want to find a triangle of minimal weight.

The Sparse $(\sqrt{n}, n, n)$-Triangle Detection for $m = n^{1.5}$ was considered in [36] who related its complexity to girth approximation in undirected graphs and hypothesized that $n^{2-o(1)}$ time is required even using fast matrix multiplication. It is easy to observe that for any m, one can combinatorially reduce BMM to Sparse $(\sqrt{n}, n, n)$-Triangle Detection, so that under the BMM Hypothesis $mn^{0.5-o(1)}$ time is required for any m that is a polynomial function of n.

In the All-Edge version of Sparse $(\sqrt{n}, n, n)$-Triangle detection, one needs to decide for every edge in $B \times C$ whether it is in some triangle. Abboud et al. [1] showed that this version of the problem (for $m = n^{1.5}$) is equivalent to the so-called Fully Sparse Boolean Matrix multiplication problem.

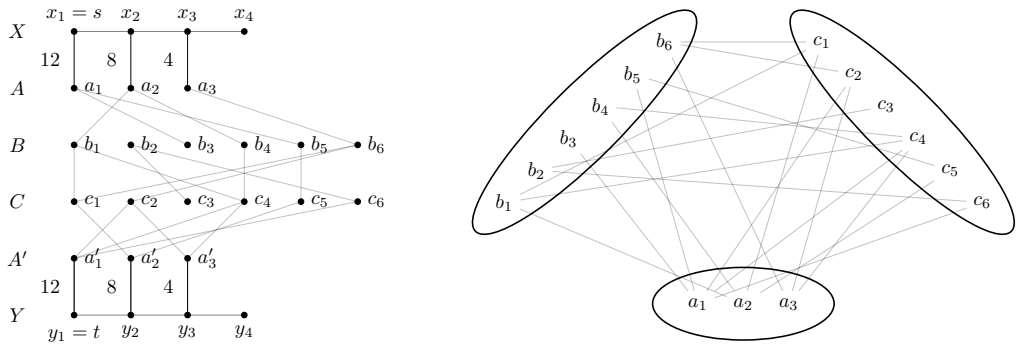

Figure 3 An example of the construction of $\mathbf{G}$ for the Sparse $(\sqrt{n}, n, n)$-Triangle Detection Problem.

▶ **Lemma 11.** *Given an instance of Sparse $(\sqrt{n}, n, n)$-Triangle Detection, we can construct an undirected, unweighted graph $\mathbf{G}$ with $\mathcal{O}(n)$ vertices and $\mathcal{O}(m)$ edges in time $\mathcal{O}(n+m)$ such that the triangle problem can be solved by reading $\mathcal{O}(\sqrt{n})$ outputs of a 2-FRP computation on $\mathbf{G}$. Furthermore, if the objective is the minimum weight version rather than detection, a similar construction holds where $\mathbf{G}$ is assigned rational weights in the range $[1, 2]$.*

We note that our construction actually reduces the potentially harder All-Nodes version of Sparse $(\sqrt{n}, n, n)$-Triangle to 2-FRP.

Proof. Let $(A \cup B \cup C, E)$ be an instance of sparse $(\sqrt{n}, n, n)$-triangle detection, where $B = \{b_1, \ldots, b_n\}$, $C = \{c_1, \ldots, c_n\}$ and $A = \{a_1, \ldots, a_\ell\}$ for some $\ell = \mathcal{O}(\sqrt{n})$.

We construct the graph $\mathbf{G}$ as follows (see Figure 3 for an illustration). The vertex set of $\mathbf{G}$ is partitioned into A, B, C, A', X, Y, where A, B, C are inherited from the input instance. The sets A', X, Y contain copies of the vertices in A; specifically, A' (resp. X and Y) contains a copy a'_i (resp. x_i and y_i) for each $i \in [1..\ell]$. In X and Y we additionally add two dummy vertices $x_{\ell+1}$ and $y_{\ell+1}$.

The edge set of $\mathbf{G}$ contains the same edges as the triangle detection instance, with the modification that edges of the type $\{c_i, a_j\}$ in the original instance become edges of the form $\{c_i, a'_j\}$ in $\mathbf{G}$. For each $i \in [1..\ell]$, we insert a path of length $4(n-i+1)$ from x_i to a_i and from a'_i to y_i. Lastly, for each $i \in [1..\ell]$, we connect x_i with x_{i+1} and y_i with y_{i+1}.

It is not difficult to see that the constructed graph $\mathbf{G}$ contains $\mathcal{O}(n + \sum_{i=1}^{\ell} 4(n-i+1)) = \mathcal{O}(n)$ nodes and $\mathcal{O}(n + \sum_{i=1}^{\ell} 4(n-i+1) + m) = \mathcal{O}(m)$ edges.

For $i \in [1..\ell]$, let $e_i \coloneqq \{x_i, x_{i+1}\}$ and $e'_i \coloneqq \{y_i, y_{i+1}\}$. Further, set $s = x_1$ and $t = y_1$. We claim that for each $i \in [1..\ell]$, $d_{\mathbf{G}\setminus\{e_i,e'_i\}}(s,t) = 11 + 8n - 6i$ if and only if there exist b_j and c_k such that $\{a_i, b_j, c_k\}$ forms a triangle in the input instance (so we can get the final answer from $d_{\mathbf{G}\setminus\{e_i,e'_i\}}(s,t)$ for all $i \in [1..\ell]$). To this end, we analyze the cost of a path in $\mathbf{G} \setminus \{e_i, e'_i\}$ that travels from s to x_i, then to a_i, takes a shortest path to a'_i, and finally proceeds to y_i and t. It is straightforward to verify that such a path has length $3+4(n-i+1)+4(n-i+1)+(i-1)+(i-1) = 11+8n-6i$ if a triangle exists (traversing b_j and c_k); otherwise, the cost is at least one unit higher. To complete the proof, we show that this is the only path that can achieve a cost of $11+8n-6i$. Indeed, in $\mathbf{G}\setminus\{e_i, e'_i\}$, any path between $a_{\hat{\imath}}$ and $a'_{\hat{\jmath}}$ has cost at least $3+4(n-\hat{\imath}+1)+4(n-\hat{\jmath}+1)+(\hat{\imath}-1)+(\hat{\jmath}-1) = 11+8n-3\hat{\imath}-3\hat{\jmath}$ for any $\hat{\imath}, \hat{\jmath} \in [1..i]$. In particular, if $\hat{\imath} \neq i$ or $\hat{\jmath} \neq i$, then $11 + 8n - 3\hat{\imath} - 3\hat{\jmath} \geq (11 + 8n - 6i) + 3$.

When handling rational weights, we make $\mathbf{G}$ weighted as follows. Let M be the largest weight in the input graph; we then assign to each edge $\{a_i, b_j\}$ of weight $w(a_i, b_j)$ in the input graph a weight of $1 + w(a_i, b_j)/(M+1)$ in $\mathbf{G}$. We do the same for edges $\{b_j, c_k\}$

and $\{c_k, a_i\}$ (which are $\{c_k, a'_i\}$ in $\mathbf{G}$). All other edges in $\mathbf{G}$ retain their original weight of one. The proof of correctness is very similar to the unweighted case. It is easy to see that for each $i \in [1..\ell]$, the same path we were considering before traversing a_i, a'_i has cost $5 + 4n - 2i + (w(a_i, b_j) + w(b_j, c_k) + w(c_k, a_i))/(M + 1)$ for $j, k \in [1..n]$ that minimize the sum $w(a_i, b_j) + w(b_j, c_k) + w(c_k, a_i)$. Using the same analysis as before, all paths going through other a_i and a'_j have cost at least $(5 + 4n - 2i) + 1$, which is strictly greater than $5 + 4n - 2i + (w(a_i, b_j) + w(b_j, c_k) + w(c_k, a_i))/(M + 1)$ because the fractional part is less than one. ◀

With Lemma 11 at our disposal, Lemma 12 follows almost immediately.

▶ **Lemma 12.** *Consider the problem of solving 2-FRP in a graph with n nodes and m edges. If $m = \Omega(n^{3/2})$, then for any $\varepsilon > 0$, there is no*

- **(i)** *$\mathcal{O}(mn^{1/2-\varepsilon})$ combinatorial algorithm for unweighted graphs, unless the BMM hypothesis is false;*
- **(ii)** *$\mathcal{O}(mn^{1/2-\varepsilon})$ algorithm for graphs with rational weights in $[1, 2]$, unless the APSP hypothesis is false;*

Proof. For (i), we show that any combinatorial algorithm running in $\mathcal{O}(mn^{1/2-\varepsilon})$ time implies a subcubic algorithm for triangle detection in a tripartite graph $\mathbf{G} = (A \cup B \cup C, E)$ with n nodes per partition (which, in turn, implies a subcubic algorithm for Boolean Matrix Multiplication).

To this end, we partition A into $\Theta(\sqrt{n})$ disjoint sets, each of size $\mathcal{O}(\sqrt{n})$, and we partition $E \cap (B \times C)$ into $\Theta(|E|/m)$ disjoint sets, each of size $\mathcal{O}(m)$. For each partition A' of A and E' of $E \cap (B \times C)$, we apply Lemma 11 to the subgraph induced by $(A' \cup B \cup C, E' \cup (E \setminus (B \times C)))$ and execute 2-FRP on the resulting instance. Note that, crucially, $m = \Omega(n^{3/2})$; since $|A'| = \mathcal{O}(\sqrt{n})$, we always have $|E \cap (A' \times B)| = \mathcal{O}(m)$ and $|E \cap (A' \times C)| = \mathcal{O}(m)$. Clearly, this reduction determines whether $\mathbf{G}$ contains a triangle. The overall runtime is $\mathcal{O}(\frac{n^2}{m} \cdot \sqrt{n} \cdot mn^{1/2-\varepsilon}) = \mathcal{O}(n^{3-\varepsilon})$, which contradicts the BMM hypothesis.

For (ii), we follow a similar approach, but instead of reducing from triangle detection, we reduce from the negative triangle detection problem. ◀

References

1. Amir Abboud, Karl Bringmann, Nick Fischer, and Marvin Künnemann. The time complexity of fully sparse matrix multiplication. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4670–4703. SIAM, 2024. doi:10.1137/1.9781611977912.167.
2. Surender Baswana, Keerti Choudhary, Moazzam Hussain, and Liam Roditty. Approximate single-source fault tolerant shortest path. *ACM Trans. Algorithms*, 16(4):44:1–44:22, 2020. doi:10.1145/3397532.
3. Surender Baswana and Neelesh Khanna. Approximate shortest paths avoiding a failed vertex: Near optimal data structures for undirected unweighted graphs. *Algorithmica*, 66(1):18–50, 2013. doi:10.1007/S00453-012-9621-Y.
4. Surender Baswana, Utkarsh Lath, and Anuradha S. Mehta. Single source distance oracle for planar digraphs avoiding a failed node or link. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 223–232, USA, 2012. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973099.20.
5. Michael A. Bender and Martin Farach-Colton. The LCA problem revisited. In Gaston H. Gonnet, Daniel Panario, and Alfredo Viola, editors, *LATIN 2000: Theoretical Informatics, 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000, Proceedings*, volume 1776 of *Lecture Notes in Computer Science*, pages 88–94. Springer, 2000. doi:10.1007/10719839_9.

6 Michael A. Bender and Martín Farach-Colton. The level ancestor problem simplified. *Theor. Comput. Sci.*, 321(1):5–12, June 2004. doi:10.1016/j.tcs.2003.05.002.

7 Omer Berkman and Uzi Vishkin. Finding level-ancestors in trees. *J. Comput. Syst. Sci.*, 48(2):214–230, April 1994. doi:10.1016/S0022-0000(05)80002-9.

8 Aaron Bernstein. A nearly optimal algorithm for approximating replacement paths and k shortest simple paths in general graphs. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 742–755. SIAM, 2010. doi:10.1137/1.9781611973075.61.

9 Amit Bhosle and Teofilo Gonzalez. Replacement paths for pairs of shortest path edges in directed graphs. In *Proc. 16th IASTED International Conference on Parallel and Distributed Computing and System (PDCS)*, September 2004.

10 Davide Bilò, Keerti Choudhary, Luciano Gualà, Stefano Leucci, Merav Parter, and Guido Proietti. Efficient oracles and routing schemes for replacement paths. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, Caen, France, February 28 - March 3, 2018*, volume 96 of *LIPIcs*, pages 13:1–13:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.STACS.2018.13.

11 Davide Bilò, Sarel Cohen, Tobias Friedrich, and Martin Schirneck. Near-optimal deterministic single-source distance sensitivity oracles. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, Lisbon, Portugal (Virtual Conference), September 6-8, 2021*, volume 204 of *LIPIcs*, pages 18:1–18:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.18.

12 Greg Bodwin and Lily Wang. Improved shortest path restoration lemmas for multiple edge failures: Trade-offs between fault-tolerance and subpaths. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 5245–5262. SIAM, 2025. doi:10.1137/1.9781611978322.179.

13 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.

14 Thomas H. Byers and Michael S. Waterman. Determining all optimal and near-optimal solutions when solving shortest path problems by dynamic programming. *Operations Research*, 32(6):1381–1384, 1984. URL: http://www.jstor.org/stable/170956.

15 Shiri Chechik and Sarel Cohen. Near optimal algorithms for the single source replacement paths problem. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2090–2109. SIAM, 2019. doi:10.1137/1.9781611975482.126.

16 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, Saarbrücken, Germany (Virtual Conference), July 8-11, 2020*, volume 168 of *LIPIcs*, pages 81:1–81:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.81.

17 Shiri Chechik and Tianyi Zhang. Faster algorithms for dual-failure replacement paths. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, Tallinn, Estonia, July 8-12, 2024*, volume 297 of *LIPIcs*, pages 41:1–41:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ICALP.2024.41.

18 Shiri Chechik and Tianyi Zhang. Nearly optimal approximate dual-failure replacement paths. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 2568–2596. SIAM, 2024. doi:10.1137/1.9781611977912.91.

19 Shucheng Chi, Ran Duan, Benyu Wang, and Tianle Xie. Undirected 3-fault replacement path in nearly cubic time. In Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis, editors, *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, July 8-11, 2025, Aarhus, Denmark*, volume 334 of *LIPIcs*, pages 57:1–57:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ICALP.2025.57.

20 Paul F. Dietz. Finding level-ancestors in dynamic trees. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Nicola Santoro, editors, *Algorithms and Data Structures, 2nd Workshop WADS '91, Ottawa, Canada, August 14-16, 1991, Proceedings*, volume 519 of *Lecture Notes in Computer Science*, pages 32–40. Springer, 1991. doi:10.1007/BFB0028247.

21 Zvi Gotthilf and Moshe Lewenstein. Improved algorithms for the k simple shortest paths and the replacement paths problems. *Inf. Process. Lett.*, 109(7):352–355, March 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 *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 748–757. IEEE Computer Society, 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 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, Glasgow, Scotland (Virtual Conference), July 12-16, 2021*, volume 198 of *LIPIcs*, pages 75:1–75:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.75.

25 Kaito Harada, Naoki Kitamura, Taisuke Izumi, and Toshimitsu Masuzawa. A nearly linear time construction of approximate single-source distance sensitivity oracles. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms, ESA 2024, Royal Holloway, London, United Kingdom, September 2-4, 2024*, volume 308 of *LIPIcs*, pages 65:1–65:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.65.

26 Monika Henzinger, Andrea Lincoln, Stefan Neumann, and Virginia Vassilevska Williams. Conditional Hardness for Sensitivity Problems. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:31, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2017.26.

27 J. Hershberger and S. Suri. Vickrey prices and shortest paths: What is an edge worth? In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, page 252, USA, 2001. IEEE Computer Society.

28 John Hershberger, Subhash Suri, and Amit M. Bhosle. On the difficulty of some shortest path problems. *ACM Trans. Algorithms*, 3(1):5:1–5:15, 2007. doi:10.1145/1219944.1219951.

29 David R. Karger, Daphne Koller, and Steven J. Phillips. Finding the hidden path: Time bounds for all-pairs shortest paths. *SIAM J. Comput.*, 22(6):1199–1217, 1993. doi:10.1137/0222071.

30 Eugene L. Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18(7):401–405, 1972. URL: http://www.jstor.org/stable/2629357.

31 K. Malik, A. K. Mittal, and S. K. Gupta. The k most vital arcs in the shortest path problem. *Oper. Res. Lett.*, 8(4):223–227, August 1989. doi:10.1016/0167-6377(89)90065-5.

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

33 Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1):166–196, 2001. doi:10.1006/game.1999.0790.

34 Jakob Nogler and Virginia Vassilevska Williams. Undirected replacement paths: Dual fault reduces to single source, 2026. arXiv:2605.02114.

35 Liam Roditty. On the k-simple shortest paths problem in weighted directed graphs. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 920–928, USA, 2007. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1283383.1283482.

36 Liam Roditty and Virginia Vassilevska Williams. Subquadratic time approximation algorithms for the girth. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 833–845. SIAM, 2012. doi:10.1137/1.9781611973099.67.

37 Liam Roditty and Uri Zwick. Replacement paths and k simple shortest paths in unweighted directed graphs. *ACM Trans. Algorithms*, 8(4):33:1–33:11, 2012. doi:10.1145/2344422.2344423.

38 Baruch Schieber, Amotz Bar-Noy, and Samir Khuller. The complexity of finding most vital arcs and nodes. Technical report, University of Maryland at College Park, USA, 1995.

39 Virginia Vassilevska Williams. Faster replacement paths. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1337–1346, USA, 2011. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973082.102.

40 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, volume 3, pages 3431–3472. World Scientific, 2018.

41 Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5):27:1–27:38, 2018. doi:10.1145/3186893.

42 Virginia Vassilevska Williams, Eyob Woldeghebriel, and Yinzhan 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, 2022. doi:10.1109/FOCS54457.2022.00090.

43 Oren Weimann and Raphael Yuster. Replacement paths and distance sensitivity oracles via fast matrix multiplication. *ACM Trans. Algorithms*, 9(2):14:1–14:13, 2013. doi:10.1145/2438645.2438646.

44 R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018. doi:10.1137/15M1024524.

45 Jin Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11):712–716, 1971. URL: http://www.jstor.org/stable/2629312.