Near Optimal Algorithms For The Single Source Replacement Paths Problem

Shiri Chechik [*] Sarel Cohen[*]

Abstract
The Single Source Replacement Paths (SSRP) problem is as follows; Given a graph $G = (V, E)$, a source vertex s and a shortest paths tree T_s rooted in s, output for every vertex $t \in V$ and for every edge e in T_s the length of the shortest path from s to t avoiding e.

We present near optimal upper bounds, by providing $\tilde{O}(m\sqrt{n} + n^2)$ time randomized combinatorial algorithm [1] for unweighted undirected graphs, and matching conditional lower bounds for the SSRP problem.

1 Introduction

Let $G = (V, E)$ be a graph with n vertices and m edges. Given two vertices s and t and an edge e, a replacement path $P_{s,t,e}$ for the triple (s, t, e) is a shortest path from s to t that avoids the edge e. In the replacement paths (RP) problem, we are given a graph G and a shortest path P between two vertices s and t, and the goal is to return for every edge e on P, a replacement path $P_{s,t,e}$.

The motivation for the replacement paths problem stems from the fact that in real world networks links are prone to failures and having backup paths between important vertices is often desirable. Not only the replacement paths problem is well motivated by its own right but it is also used in other important applications. One application relates to auction theory, where the replacement paths problem is used to compute the Vickrey pricing of edges owned by selfish agents [25, 17]. Another application is that of computing the k shortest simple paths between a pair of vertices. The k shortest simple paths problem can be computed by k calls to the replacement paths algorithm (this reduction holds both in the directed weighted case and also in the undirected unweighed case) and has many applications by itself [14].

The replacement paths problem has attracted a lot of attention and many of the algorithms for different aspects of this problem admit by-now near optimal solutions (see e.g. [22, 24, 23, 27, 13, 19, 32, 29]). For a more extensive literature survey on the replacement paths problem see the full version.

In this paper we consider a natural and important generalization of the replacement paths problem, referred to as the single source replacement paths (SSRP) problem, which is defined as follows. Given a graph G and a fixed source vertex s the SSRP(s) problem is to compute the lengths of the replacement paths $P_{s,t,e}$ for every vertex $t \in V$ and for every edge $e \in P(s, t)$.

For every vertex $t \in V$, the number of edges on $P(s, t)$ is bounded by $O(n)$ and thus, there are $O(n^2)$ replacement paths $P_{s,t,e}$. It follows that the size of the SSRP(s) output (which is the lengths of these $O(n^2)$ replacement paths $P_{s,t,e}$) is $O(n^2)$.

Quite surprisingly, the SSRP problem, despite of its natural flavour, attracted much less attention. The first reference to the best of our knowledge is a paper by Hershberger *et al.* [18] that referred to the problem as edge-replacement shortest paths trees and showed that in the path-comparison model of computation of Karger *et al.* [18], SSRP on directed graphs with n nodes, m edges and arbitrary edge weights requires $\Omega(mn)$ comparisons. The reductions by Vassilevska Williams and Williams [31] imply that, for arbitrary weights, any $O(n^{3-\epsilon})$ algorithm for a constant $\epsilon > 0$ even for the simpler RP problem would imply a $O(n^{3-\delta})$ algorithm for some constant $\delta > 0$ for all-pairs shortest paths (APSP). Therefore, there seems to be little hope to obtain a subcubic time algorithm for the general case of SSRP of directed graphs with arbitrary edge weights.

To the best of our knowledge, the only non trivial upper bound for the SSRP problem was obtained by Grandoni and Vassilevska Williams [15], who were also the ones who named this problem the single source replacement paths problem. Grandoni and Vassilevska Williams [15] showed that by restricting the problem for small integer edge weights and by using fast matrix multiplications one can "break" these lower bounds. More precisely, they showed that for graphs with positive integer edge weights in the range $[1, M]$, SSRP can be computed in $\widetilde{O}(Mn^{\omega})$ time (here $\omega < 2.373$ is the matrix multiplication exponent [30, 20]). For integer edge

[*]Tel Aviv University, Israel. E-mails: `shiri.chechik@gmail.com, sarelcoh@post.tau.ac.il`. This work was partially supported by ISF grant No. 1528/15 and the Blavatnik Fund.

[1]As usual, n is the number of vertices, m is the number of edges and the $\tilde{O}$ notation suppresses polylogarithmic factors.

weights in the range $[-M, M]$, they presented an algorithm for SSRP in $\widetilde{O}(M^{\frac{1}{4-\omega}} n^{2+\frac{1}{4-\omega}})$ time. For comparison the current best known bound for the simpler problem of the RP for directed graph with weights $[-M, M]$ is by Vassilevska Williams [29] who gave an $\widetilde{O}(Mn^{\omega})$ time algorithm. Interestingly, the SSRP problem with positive weights of [15] matches the best known bound for the RP problem. However, the best known bound for the SSRP problem with negative weights [15] is larger than the bound for the RP problem with negative weights. This interesting gap between the best SSRP algorithm and the RP algorithm in the case of negative weights raises an interesting question of whether there is an existential gap between these two problems or whether one can close this gap. Grandoni and Vassilevska Williams [15] conjectured that the gap between these two problems is essential and in fact they conjectured that the SSRP problem with negative weights is as hard as the directed All Pairs Shortest Paths (APSP) problem.

The lower bounds for both the RP and the SSRP problems apply for directed graphs only. In fact, for the undirected case the RP problem admits a near linear time algorithm of $\tilde{O}(m)$ [22, 24, 23]. A natural question is whether the undirected version of the SSRP can also be solved in near linear time. In this paper we show that under some well believed assumptions this is impossible. We therefore present a separation gap between the SSRP problem and the RP problem. To the best of our knowledge this is the first separation gap introduced for these problems.

More precisely, for the undirected unweighted case we present matching upper and conditional lower bounds of $\tilde{\Theta}(m\sqrt{n} + n^2)$ time. Namely, we show that the SSRP problem for undirected unweighted graphs can be solved in $\widetilde{O}(m\sqrt{n} + n^2)$ time. In addition, we show that there is no combinatorial algorithm for solving undirected unweighted SSRP in $O(m(\sqrt{n})^{1-\epsilon} + n^2)$ for some constant ϵ unless there is a combinatorial algorithm for solving the Boolean Matrix Multiplication problem (BMM) on two $n \times n$ matrices A and B containing a total number of m 1's in $O(mn^{1-\epsilon})$ time. The n^2 term in the lower bound is trivial as we need $\Omega(n^2)$ time just to output the solution.

The term "combinatorial algorithm" is loosely defined, and it is often referred to as an algorithm that does not use any matrix multiplication tricks. The interest in combinatorial algorithms stems from the assumption that in practice combinatorial algorithms are much more efficient since the constants hidden in the matrix multiplication bounds are considered to be high.

Our main results are summarized in the following theorems.

Theorem 1.1. *There exists an $\tilde{O}(m\sqrt{n} + n^2)$ time combinatorial algorithm for the SSRP problem on unweighted undirected graphs. Our randomized algorithm is a monte carlo with a one-sided error, as we always output distances which are at least the exact distances, and with high probability (of at least $1 - n^{-q}$ for any constant $q > 0$) we output the exact distance.*

Our lower bound for undirected unweighted graphs (given in the next theorem) relies on the following conjecture.

Conjecture 1.1. (The BMM Conjecture) *In the Word RAM model with words of $O(\log n)$ bits, any combinatorial algorithm requires $\Omega(mn^{1-o(1)})$ time in expectation to compute the boolean product of two $n \times n$ matrices containing a total number of m 1's.*

Theorem 1.2. *Assuming the BMM conjecture any combinatorial algorithm for solving the SSRP problem on n-vertices m-edges undirected unweighted graphs requires $\Omega(mn^{0.5-\epsilon} + n^2)$ time.*

The SSRP problem considered in this paper is closely related to the distance sensitivity oracles problem. A distance sensitivity oracle is a preprocessed data structure on a given graph G that can answer distance queries of the following form. Given two vertices s and t and a set F of failed edges/vertices return the distance from s to t in $G \setminus F$. The distance sensitivity oracles problem has been extensively studied and the literature covers many different aspects of it (see e.g. [11, 3, 4, 12, 7, 5, 28, 15, 10, 6]).

In the full version, we show that for the undirected weighted case, no combinatorial SSRP algorithm exists in $O(mn^{1-\epsilon})$ time for any constant $\epsilon > 0$ unless APSP has a combinatorial algorithm with truly subcubic time. This matches (up to polylog factors) the upper bound of $\widetilde{O}(mn)$ (that can be derived easily from e.g. [4]). Therefore, the case of combinatorial SSRP for undirected weighted graphs is essentially resolved. Together with our new result, this completes the picture of SSRP for undirected graphs.

2 Preliminaries

Let $G = (V, E)$ be a weighted graph with integer edge weights in the range $[-M, M]$. Let P be a path in G. We denote by $w(P)$ the length of the path P which is defined as the sum of the weights of the edges along P, and by $|P|$ the number of edges of P. For unweighted graphs, $|P| = w(P)$.

Let $u, v \in V$ be two vertices, we denote by $P_G(u, v)$ a shortest path from u to v in G, and denote by $d_G(u, v)$ the distance from u to v in the graph G (i.e.,

$d_G(u,v) = w(P_G(u,v))$. When G is clear from the context, we abbreviate $P_G(u,v)$ by $P(u,v)$ and $d_G(u,v)$ by $d(u,v)$. Let $e = (x,y) \in E$, we define $d(s,e) = \min\{d(s,x), d(s,y)\}$.

We denote by $D = \max_{s,t\in V} d(s,t)$ the diameter of the graph G (which is the largest distance in the graph G).

Let $F \subset E$ be a set of edges, we denote by $G\setminus F$ the graph $(V, E\setminus F)$, that is the graph obtained by removing the set of edges F from G.

Let $s,t \in V$ be two vertices and $e \in P(s,t)$ be an edge on the shortest path from s to t. The replacement path associated with the triple (s,t,e), denoted by $P_{s,t,e}$, is the shortest path from the *source* vertex s to the *target* vertex t avoiding the edge e. We denote by $d_{s,t,e} = w(P_{s,t,e})$ the distance from s to t in the graph $G \setminus \{e\}$. Similarly, let $F \subset E$ be a set of edges, we denote by $P_{s,t,F}$ the shortest path from s to t in the graph $G\setminus F$ and by $d_{s,t,F} = w(P_{s,t,F})$ the distance from s to t in the graph $G \setminus F$.

For every vertex $v \in V$ we define by T_v the shortest paths tree rooted in v. We denote by T_v^e the shortest paths tree in the graph $G \setminus \{e\}$ rooted in v.

For a graph H we denote by $V(H)$ the set of its vertices, and by $E(H)$ the set of its edges. Let P be a path and u,v be two vertices on this path. We denote by $P[u..v]$ the subpath of P from u to v. When it is clear from the context, we sometimes abbreviate $e \in E(P)$ as $e \in P$ to denote an edge of the path P, and $v \in V(P)$ as $v \in P$ to denote a vertex of the path P.

We now define the path concatenation operator $\circ$. Let $P_1 = (x_1, x_2, \ldots, x_r)$ and $P_2 = (y_1, y_2, \ldots, y_t)$ be two paths. Then $P = P_1 \circ P_2$ is defined as the path $P = (x_1, x_2, \ldots, x_r, y_1, y_2, \ldots, y_t)$, and it is well defined if either $x_r = y_1$ or $(x_r, y_1) \in E$.

We will assume, without loss of generality, that every replacement path $P_{s,t,e}$ can be decomposed into a common prefix $\text{CommonPref}_{s,t,e}$ with the shortest path $P(s,t)$, a detour $\text{Detour}_{s,t,e}$ which is disjoint from the shortest path $P(s,t)$ (except from its first and last vertices), and finally a common suffix $\text{CommonSuff}_{s,t,e}$ which is common with the shortest path $P(s,t)$. Therefore, for every edge $e \in P(s,t)$ it holds that $P_{s,t,e} = \text{CommonPref}_{s,t,e} \circ \text{Detour}_{s,t,e} \circ \text{CommonSuff}_{s,t,e}$ (the prefix and/or suffix may be empty). See Figure 1 for illustration.

We remark that the common prefix $\text{CommonPref}_{s,t,e}$, the detour $\text{Detour}_{s,t,e}$ and the common suffix $\text{CommonSuff}_{s,t,e}$ are all defined given a specific replacement path $P_{s,t,e}$. Other replacement paths from s to t avoiding e may have a different common prefix, detour or common suffix. When the replacement path $P_{s,t,e}$ will be clear from the context, we will refer to $\text{CommonPref}_{s,t,e}$, $\text{Detour}_{s,t,e}$ and $\text{CommonSuff}_{s,t,e}$ as explained above relative to $P_{s,t,e}$. When the replacement path is not clear from the context, we will mention to which replacement path we associate $\text{CommonPref}_{s,t,e}$, $\text{Detour}_{s,t,e}$ and $\text{CommonSuff}_{s,t,e}$.

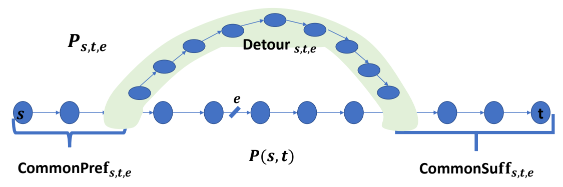

Figure 1: $P(s,t)$ is the shortest path from s to t, $P_{s,t,e}$ is the replacement path from s to t avoiding e, $\text{CommonPref}_{s,t,e}$ is the common prefix of $P_{s,t,e}$ and $P(s,t)$, $\text{CommonSuff}_{s,t,e}$ is the common suffix of $P_{s,t,e}$ and $P(s,t)$, and $\text{Detour}_{s,t,e}$ is the detour of $P_{s,t,e}$ which is edge-disjoint from $P(s,t)$. By definition, $P_{s,t,e} = \text{CommonPref}_{s,t,e} \circ \text{Detour}_{s,t,e} \circ \text{CommonSuff}_{s,t,e}$.

The following sampling Lemma is a folklore, we prove it in the full version.

LEMMA 2.1. *[Random Sampling] Consider n balls of which R are red and $n-R$ are blue and let $C, Q > 0$ be constants such that $R > 2CQ\ln n$. Let B be a random set of balls such that each ball is chosen to be in B independently uniformly at random with probability $\frac{(CQ\ln n)}{R}$. Then with high probability (with probability at least $1 - \frac{2C}{n^Q}$) at least C of the balls of B are red and the size of B is $\widetilde{O}(n/R)$.*

We note that previous related algorithms (e.g. [27, 15]) either used matrix multiplication (as in [15], and thus couldn't avoid the term n^ω in their running time) or handled the seemingly easier problem of replacement paths and heavily relied on the fact that there is a single target (as in [27]).

We next highlight several tools and ideas used in our construction. First, we define the notion of an edge e being (x,y)-replaceable as follows.

DEFINITION 2.1. *Let $x,y \in V, e \in E$, we say that e is (x,y)-replaceable iff $d_{x,y,e} = d(x,y)$ (i.e., e is (x,y)-replaceable iff there exists a shortest path from x to y which avoids e). Observe that in undirected graphs, e is (x,y)-replaceable iff e is (y,x)-replaceable.*

In our construction we develop the following useful tool, which we refer to as replaceability oracle. To build such replaceability oracles we utilize reachability oracles ([8]) and dominator trees ([21]).

Theorem 2.1. *[proof in Section 4] There exists a data-structure (single source replaceability oracle) such that given an undirected graph $G = (V, E)$ with n vertices and m edges, and a source vertex $v \in V$, preprocesses the graph in $\widetilde{O}(m)$ time, stores a data-structure R_v of size $\widetilde{O}(n)$. The data-structure R_v answers replaceability queries $(t, e) \in V \times E$ in constant time, that is, given a query (t, e) the data-structure answers if $d_{v,t,e} = d(v, t)$.*

Secondly, we introduce the notion of "double pivot". A common approach in the existing literature (see e.g., [27, 28, 15]) is to use sampling in order to hit all the long detours and then to compute distances exactly from the chosen vertices. The idea is that given a polynomial number of "long" detours, each containing at least L edges, then by sampling $\widetilde{O}(n/L)$ vertices, each detour contains w.h.p. at least one sampled vertex. Our goal in sampling two pivots on a detour is different. We aim to utilize a structural property of shortest paths from [1]. In [1] (Theorem 1) the authors prove that every s-to-t shortest path $P_{s,t,F}$ in an unweighed undirected graph $G \setminus F$ (with $|F| = f$) can be decomposed into a concatenation of $f + 1$ shortest path in the graph G. Specifically, for the case of a single failure ($f = 1$) it follows that there exists a vertex q on the path $P_{s,t,e}$ such that both its prefix from s to q and its suffix from q to t are shortest paths in the graph G. It would be helpful if we could sample the vertex q as a pivot, however the probability to sample q specifically may be very low. It turns out that it is sufficient to sample a vertex x before q (along $P_{s,t,e}$) and a vertex y after q (along $P_{s,t,e}$) such that the subpath of $P_{s,t,e}$ from x to y contains at most $\widetilde{O}(\sqrt{n})$ edges. The replacement path is then obtained by concatenating three shortest paths, from s to x, from x to y, and from y to t, from the original graph. In order for this to work, the algorithm rules out that these shortest paths contain the edge e that we want to avoid. See more details in the "double pivot" case in Section 7.

Another ides is our use of path intervals. The notion of splitting a path into segments was already known in the literature (*e.g.*, [27, 28, 4, 6]), however we define path intervals in a new way. Let $P(s, t) = (v_0, \dots, v_{d(s,t)})$, and $P(s, t, e)$ be a replacement path, and let x, y be two vertices on its detour part such that x appears before y along $P(s, t, e)$. The replacement path $P(s, t, e)$ avoids a subpath $(v_i, \dots, v_j)$ of $P(s, t)$. But this subpath $(v_i, \dots, v_j)$ may not be maximal, in the sense that there might exist another replacement path $P'(s, t, e)$ such that x appears before y on the detour part of $P'(s, t, e)$, and $P'(s, t, e)$ avoids a larger subpath $(v_{i'}, \dots, v_{j'}) \supset (v_i, \dots, v_j)$. Intuitively, we define the path interval $I_{x,y,t}$ as the maximum subpath of $P(s, t)$ which is avoided by at least one replacement path from s to t avoiding e which passes through x, y. We prove the existence and uniqueness of path intervals in our context in Section 7, and use it to efficiently find replacement paths in the "double pivot" case in Section 8.

3 An $\widetilde{O}(m\sqrt{n} + n^2)$ Algorithm for SSRP in Unweighted Undirected Graphs - An Overview

In this Section we present an overview of our $\widetilde{O}(m\sqrt{n} + n^2)$ algorithm for the SSRP problem in undirected unweighted graphs, which is near optimal according to our conditional lower bound in Section 10.

Let s be the source vertex, we compute T_s the shortest paths tree (BFS tree) rooted in s, and let $E(T_s)$ be the edges of the tree T_s. The output of our algorithm is the $O(n^2)$ distances: for every $t \in V, e \in E(T_s)$ we output $d_{s,t,e}$, the length of the shortest path from s to t avoiding e.

First, we sample every vertex $x \in V$ uniformly independently at random with probability $\frac{CQ \log n}{\sqrt{n}}$ for large enough constants $C, Q > 0$. Let B be the set of all the sampled vertices (also referred to as *pivots*). According to Lemma 2.1, it holds with high probability that $|B| = \widetilde{O}(\sqrt{n})$. For our analysis it is enough to assume that $|B| = \widetilde{O}(\sqrt{n})$. However, it is worth mentioning that if we sampled too many vertices and it does not hold that $|B| = \widetilde{O}(\sqrt{n})$ then we can resample B (by choosing every vertex $x \in V$ at random with probability $\frac{CQ \log n}{\sqrt{n}}$) until $|B| = \widetilde{O}(\sqrt{n})$.

Let $e = (u, v) \in E(T_s)$ be a tree-edge such that u is closer than v to the root s in the tree T_s (*i.e.*, $d(s, u) = d(s, v) - 1$), and let $t \in V$. In our algorithm and analysis we consider the following cases.

1. The replaceable edge case: Here we consider the case that e is (s, t)-replaceable, *i.e.*, $d_{s,t,e} = d(s, t)$. We construct a replaceability oracle R_s as in Theorem 2.1 that outputs in constant time for every $t \in V$ and $e \in E(T_s)$ whether or not $d_{s,t,e} = d(s, t)$. We use R_s to compute and store a table h_0 which is defined as follows for every $t \in V$ and $e \in E(T_s)$.

$$h_0[t, e] = \begin{cases} d_{s,t,e}, & \text{if } d_{s,t,e} = d(s,t) \\ \infty, & \text{otherwise} \end{cases} \tag{3.1}$$

We describe this case in detail in Section 4.

2. The small fall case: Here we consider the case that $d(s, t) < d_{s,t,e} \le d(s, u) + 4\sqrt{n}$. Recall that $e = (u, v)$ such that u is closer than v to s. We compute and store a table h_1 such that for every $t \in V$ and $e \in E(T_s)$ the following Equation holds:

(3.2)

$$h_1[t,e] = \begin{cases} d_{s,t,e}, & \text{if } d(s,t) < d_{s,t,e} \le d(s,u) + 4\sqrt{n} \\ \infty, & \text{otherwise} \end{cases}$$

We describe this case in detail in Section 5.

3. The single pivot case: In this case the algorithm computes and stores

(3.3)

$$h_2[t,e] = \min_{x \in B}\{\min\{h_0[x,e], h_1[x,e]\} + d(x,t) \mid e \text{ is among the last } 2\sqrt{n} \text{ edges of the path from } s \text{ to } x \text{ in } T_s\}.$$

If there is no vertex $x \in B$ such that e is among the last $2\sqrt{n}$ edges of the path from s to x in T_s, then we define $h_2[t,e] = \infty$. We describe this case in detail in Section 6.

4. The double pivot case: let $t \in V, e \in E(T_s)$. We say that (t,e) is of type "double pivot" if the following conditions hold: (a) There exists a replacement path $P_{s,t,e}$ and $x, y \in B$ such that x appears before y on the detour part of $P_{s,t,e}$. (b) e is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable.

Observe that we allow $x = y$ to be the same vertex of B. In the double pivot case the algorithm computes and stores a table h_3:

(3.4)

$$h_3[t,e] = \begin{cases} d_{s,t,e}, & \text{if } (t,e) \text{ is of type "double pivot"} \\ \ge d_{s,t,e}, & \text{otherwise} \end{cases}$$

We describe this case in detail in Sections 7 and 8.

For every pair (t,e) such that $t \in V, e \in E(T_s)$ we output $\hat{d}_{s,t,e} = \min\{h_0[t,e], h_1[t,e], h_2[t,e], h_3[t,e]\}$. According to Equations 3.1, 3.2 and 3.4 it is trivial that $h_0[t,e], h_1[t,e], h_3[t,e] \ge d_{s,t,e}$. In Section 6 we prove that $h_2[t,e] \ge d_{s,t,e}$. Therefore, it holds that $\hat{d}_{s,t,e} \ge d_{s,t,e}$. In Sections 4, 5, 6, and 7 we describe how to compute h_0, h_1, h_2, h_3 in time $\widetilde{O}(m\sqrt{n} + n^2)$. In Section 9 we prove the correctness of our algorithm by proving the following Theorem.

Lemma 3.1. *[proof in Section 9]. Let $t \in V, e \in E(T_s)$ and $\hat{d}_{s,t,e} = \min\{h_0[t,e], h_1[t,e], h_2[t,e], h_3[t,e]\}$.*
Then it holds w.h.p. that $\hat{d}_{s,t,e} = d_{s,t,e}$.

The most technical part of the paper is with handling the double pivot case in Sections 7 and 8.

4 Replaceable edges and replaceability oracles

In this Section we construct replaceability oracles, which are important components in our SSRP algorithm. In particular we can use them to compute in $\widetilde{O}(m + n^2)$ time, for every $t \in V$ and $e \in E(T_s)$ whether or not $d_{s,t,e} = d(s,t)$ and construct the table h_0.

Given a source vertex $v \in V$, we show how to construct a replaceability data-structure R_v in $\widetilde{O}(m)$ time that supports replaceable edge queries; Given a query (t,e) with $t \in V, e \in E$ the data-structure answers if e is a (v,t)-replaceable edge in constant time.

First we need the following tool, a reachability oracle of a directed acyclic graph. The following result follows from dominator trees [21], and it is also described in Section 3 of [8].

Theorem 4.1. *[8] There exists a data-structure (single source reachability oracle) such that given a directed acyclic graph $G = (V,E)$ with n vertices and m edges, and a source vertex $v \in V$, preprocesses the graph in $\widetilde{O}(m)$ time, stores a data-structure Q_v of size $\widetilde{O}(n)$. The data-structure Q_v answers reachability queries $(t,u) \in V \times V$ in constant time, that is, given a query (t,u) the data-structure reports if there is any path from v to t that avoids the vertex u.*

It is easy to generalize Lemma 4.1 to answer queries $(t,e) \in V \times E$. Create a graph G' obtained from G by adding for every edge $e = (x,y) \in E(T_v)$ a vertex $p(e)$, and replacing the edge (x,y) with the edges $(x,p(e)),(p(e),y)$. Then every path from v to t in G must pass through e iff every path from v to t must pass through the vertex $p(e)$ in G. In addition we only added $O(n)$ nodes to the graph and thus the new number of nodes remains $O(n)$. This proves the following Corollary.

Corollary 4.1. *There exists a data-structure (single source reachability oracle) such that given a directed acyclic graph $G = (V,E)$ with n vertices and m edges, and a source vertex $v \in V$, preprocesses the graph in $\widetilde{O}(m)$ time, stores a data-structure Q_v of size $\widetilde{O}(n)$. The data-structure Q_v answers reachability queries $(t,e) \in V \times E$ in constant time, that is, given a query (t,e) the data-structure reports if there is any path from v to t that avoids the edge e.*

Using the reachability oracle we construct a replaceability oracle, as per Theorem 2.1.

Proof. [proof of Theorem 2.1] Compute the BFS tree T_v rooted at v, and the distance $d(v,t)$ for all $t \in V$.
Using the BFS tree T_v we create a DAG $\vec{G}_v = (V, \vec{E}_v)$ whose vertex set is V. The set of edges $\vec{E}_v$ is obtained by removing from E all the edges $e = (a,b)$ such that $d(v,a) = d(v,b)$ (*i.e.*, remove all cross edges which are on the same level of T_v), and the remaining edges are directed away from the root v (*i.e.*, the edge $e = (a,b)$ is directed from a to b such that $d(v,a) < d(v,b)$, which means that a has a lower level than b in the tree T_v). Given the graph $\vec{G}_v$ we compute the single source reachability oracle specified in Theorem 4.1, denote it be R_v.
We claim that t is reachable from v in $\vec{G}_v \setminus \{e\}$ iff e is (v,t)-replaceable.
For the first direction, we assume that t is reachable from

v in $\vec{G}_v \setminus \{e\}$ and prove that $d_{v,t,e} = d(v,t)$ (*i.e.*, e is (v,t)-replaceable). Recall that the graph $\vec{G}_v \setminus \{e\}$ contains only edges (x,y) such that x is closer by 1 to v than y in the graph G. It follows that any directed path from v to t in $\vec{G}_v \setminus \{e\}$ is of length $d(v,t)$. Therefore, if t is reachable from v in $\vec{G}_v \setminus \{e\}$ then there exists a path $P \in \vec{G}_v \setminus \{e\}$ from v to t whose length is $|P| = d(v,t)$. The undirected version of P is a path in $G \setminus \{e\}$ and hence $|P| \geq d_{v,t,e}$. Since it always holds that the shortest path after the deletion of an edge may only be longer than the shortest path in the original graph, it follows that $d_{v,t,e} \geq d(v,t)$ and hence $d(v,t) = |P| \geq d_{v,t,e} \implies d(v,t) = d_{v,t,e}$.
For the other direction, we assume that $d_{v,t,e} = d(v,t)$ and prove that t is reachable from v in $\vec{G}_v \setminus \{e\}$. Let $P_{v,t,e}$ be the shortest path from v to t in the graph $G \setminus \{e\}$. Let $\vec{P}_{v,t,e}$ be a directed version of the path $P_{v,t,e}$, where the edges along the path are directed from v towards t. Since $d_{v,t,e} = d(v,t)$ it follows that $|\vec{P}_{v,t,e}| = d(v,t)$. Therefore, for every edge $(x,y) \in \vec{P}_{v,t,e}$ we must have $d(v,x) = d(v,y) - 1$, and hence $(x,y) \in \vec{G}_v \setminus \{e\}$. It follows that $\vec{P}_{v,t,e} \in \vec{G}_v \setminus \{e\}$ and hence t is reachable from v in $\vec{G}_v \setminus \{e\}$.

We describe how to construct h_0 in $\widetilde{O}(m + n^2)$ time. First, we construct R_s in $\widetilde{O}(m)$ time as in Lemma 2.1. Then, for every $t \in V, e \in E(T_s)$ (there are $O(n^2)$ such pairs (t,e)) we use R_s to check in constant time whether or not e is (s,t)-replaceable, *i.e.*, whether or not $d_{s,t,e} = d(s,t)$. Accordingly, we store the table h_0 such that if $d_{s,t,e} = d(s,t)$ then $h_0[t,e] = d_{s,t,e}$ and otherwise $h_0[t,e] = \infty$. We conclude the following Lemma.

Lemma 4.1. *One can compute the table h_0 in $\widetilde{O}(m + n^2)$ time such that Equation 3.1 holds.*

4.1 Some Properties Of Replaceable Edges

In this Section we prove that for every edge e on a shortest path $P(x,y)$ and for every vertex $p \in V$ we have that e is either (x,p)-replaceable or (p,y)-replaceable. This Lemma will be useful in the following sections.

Lemma 4.2. *Let $x, y \in V$, $e \in P(x,y)$ be an edge on a shortest path from x to y and $p \in V$ be an arbitrary vertex. Then either $e \notin P(x,p)$ or $e \notin P(p,y)$.*

Proof. Assume by contradiction that $e \in P(x,p)$ and $e \in P(p,y)$.
Let $e = (u,v)$ such that u is closer to x than y along $P(x,y)$. Since u is closer to x than v along $P(x,y)$ it follows that $d(x,u) < d(x,v)$ and hence u also appears before v on $P(x,p)$. Similarly, it must be that u appears before v in $P(p,y)$ (that is, v is closer to y than u).
Let P_1 be the subpath of $P(x,p)$ from x to u and let P_2 be the subpath of $P(x,p)$ from v to p. Let P_3 be the subpath of $P(p,y)$ from p to u and let P_4 be the subpath of $P(p,y)$ from v to y. Then $P(x,p) = P_1 \circ (u,v) \circ P_2$ and $P(p,y) = P_3 \circ (u,v) \circ P_4$. Let P'_2 be the path P_2 in reverse order (*i.e.*, from p to v) and let P'_3 be the path P_3 in reverse order (*i.e.*, from u to p).
Define $P' = P_1 \circ P'_3 \circ P'_2 \circ P_4$. Then P' is a path from x to y going through p and avoiding e and hence $|P'| \geq d_{x,p,e} + d_{p,y,e} \geq d(x,p) + d(p,y)$. On the other hand P' contains the edges of $P(x,p) \circ P(p,y)$ excluding the edge e from both $P(x,p)$ and $P(p,y)$ and thus $|P'| = |P(x,p) \setminus \{e\}| + |P(p,y) \setminus \{e\}| = |P(x,p)| - 1 + |P(p,y)| - 1 < d(x,p) + d(p,y)$. On the one hand $|P'| \geq d(x,p) + d(p,y)$ and on the other hand $|P'| < d(x,p) + d(p,y)$ so we get a contradiction.

According to Definition 2.1 and Lemma 4.2 we get the following corollary.

Corollary 4.2. *Let $x, y \in V$, $e \in P(x,y)$ be an edge on a shortest path from x to y and $p \in V$ be an arbitrary vertex. Then e is either (x,p)-replaceable or (p,y)-replaceable.*

We prove another claim regarding replaceable edges, which is similar to Lemma 4.2 but with some differences.

Lemma 4.3. *Let $x, y \in V$, $e \in E$ be an arbitrary edge and $p \in P_{x,y,e}$ be a vertex on the replacement path $P_{x,y,e}$. Then e is either not (x,p)-replaceable or not (p,y)-replaceable.*

To prove Lemma 4.3 we use the following Theorem from [1].

Theorem 4.2. (Theorem 1 in [1]) *After f failures in an unweighted undirected graph, each new shortest path is the concatenation of at most $f+1$ original shortest paths.*

Proof. [Proof of Lemma 4.3] According to Theorem 4.2 it follows that $P_{x,y,e}$ is a concatenation of two shortest paths in G. Let $q \in V(P_{x,y,e})$ be a vertex such that both the subpath of $P_{x,y,e}$ from x to q and the subpath of $P_{x,y,e}$ from q to y are shortest paths in G. Denote the subpath of $P_{x,y,e}$ from x to q by P_1 and the subpath of $P_{x,y,e}$ from q to y by P_2.
If $p \in P_1$ then p appears before q along the path $P_{x,y,e}$ or $p = q$. Since the subpath of $P_{x,y,e}$ from x to q is a shortest path in G, it also follows that the subpath of $P_{x,y,e}$ from x to p is a shortest path, and thus e is (x,p)-replaceable.
If $p \in P_2$ then q appears before p along the path $P_{x,y,e}$ (or $p = q$). Since the subpath of $P_{x,y,e}$ from q to y is a shortest path in G, it also follows that the subpath of $P_{x,y,e}$ from p to y is a shortest path, and thus e is not (p,y)-replaceable.

5 Handling The Small Fall Case

In this Section we describe how to compute h_1 in $\widetilde{O}(m\sqrt{n} + n^2)$ time. Let $t \in V$ and let $e = (u,v) \in E(T_s)$ be a tree-edge such that u is closer than v to the root s in the tree T_s. If $d(s,t) < d_{s,t,e} \leq d(s,u) + 4\sqrt{n}$ then we will have $h_1[t,e] = d_{s,t,e}$, and otherwise the algorithm symbolically sets $h_1[t,e] = \infty$.

Let $T_{s,v}$ be the subtree of T_s rooted in v. Let $\bar{T}_{s,v}$ be the tree $T_{s,v}$ truncated at depth $4\sqrt{n}$ (*i.e.*, we trim the subtree of v at depth $4\sqrt{n}$, and remove from it all the vertices whose distance from v is more than $4\sqrt{n}$). In the full version we prove the following lemma.

Lemma 5.1. *Let $t \in V$ and let $e = (u,v) \in E(T_s)$ be a tree-edge such that u is closer than v to the root s in the tree T_s. Assume $d(s,t) < d_{s,t,e} \leq d(s,u) + 4\sqrt{n}$ then $t \in \bar{T}_{s,v}$.*

The graph G_e: For every edge $e = (u, v) \in E(T_s)$ (such that u is closer than v to s) we construct a weighted graph $G_e = (V, E_e, w_e)$ which is defined as follows. Let $E_e^1 = \{(x, y) | x \in V(\bar{T}_{s,v}) \text{ OR } y \in V(\bar{T}_{s,v})\} \setminus \{e\}$ be the set of all edges incident to vertices in $\bar{T}_{s,v}$ (except the edge e itself), we set $w_e(x, y) = 1$ for every $(x, y) \in E_e^1$. Let $E_e^2 = \{(s, x) | x \in V \setminus V(T_{s,v})\}$ be additional edges from s to every vertex x not in the subtree $T_{s,v}$, we set $w_e(s, x) = d(s, x)$ for every $(s, x) \in E_e^2$. Let $E_e = E_e^1 \cup E_e^2$.

The algorithm for computing h_1 is as follows. For every edge $e = (u, v) \in E(T_s)$ we run Dijkstra(G_e, s) in the graph G_e from source vertex s. We denote the distances computed by Dijkstra(G_e, s) by $d_e(s, t)$, which is the distance from s to t in the graph G_e. For every $e = (u, v) \in E(T_s)$ and for every $t \in \bar{T}_{s,v}$ if $d(s, t) < d_e(s, t) \le d(s, u) + 4\sqrt{n}$ then we set $h_1[t, e] = d_e(s, t)$, otherwise if either $d(s, t) = d_e(s, t)$ or $d_e[t] > d(s, u) + 4\sqrt{n}$ we set $h_1[t, e] = \infty$. For every $e = (u, v) \in E(T_s)$ and $t \notin \bar{T}_{s,v}$ we set $h_1[t, e] = \infty$.

We prove in the full version that computing h_1 takes $\widetilde{O}(m\sqrt{n} + n^2)$ time. The main idea of the proof is that every graph G_e contains at most n edges for E_e^2 and that the number of edges in E_e^1 is the sum of the degrees of vertices u in $\bar{T}_{s,v}$. We show that every vertex z belongs to at most $\sqrt{n}$ trimmed trees $\bar{T}_{s,v}$ for some vertex $v \in V$ and thus every vertex contributes its degree to at most $\sqrt{n}$ graphs G_e and thus to at most $\sqrt{n}$ Dijkstra's computations. We show that this implies the desired running time.

LEMMA 5.2. *Let $e = (u, v) \in E(T_s)$ be a tree-edge such that u is closer than v to the root s in the tree T_s. Let $t \in \bar{T}_{s,v}$ and let $d_e(s, t)$ be the distance from s to t in the weighted graph G_e. If $d_{s,t,e} \le d(s, u) + 4\sqrt{n}$ then $d_e(s, t) = d_{s,t,e}$. Else (if $d_{s,t,e} > d(s, u) + 4\sqrt{n}$) then $d_e(s, t) > d(s, u) + 4\sqrt{n}$.*

Proof. We first prove that for every $t \in V$ it holds that $d_e(s, t) \ge d_{s,t,e}$. To show that notice that it is enough to show that distances in G_e are at least the distances in $G \setminus \{e\}$. More precisely, notice also that it is enough to show that for every edge $\tilde{e} \in E_e$ such that $\tilde{e} = (z_1, z_2)$, the weight of the edge $\tilde{e}$ is at least the distance $d_{z_1,z_2,e}$. Consider an $\tilde{e} \in E_e$ such that $\tilde{e} = (z_1, z_2)$. If $\tilde{e} \in E_e^2$ then recall that all edges in E_e^2 represents paths from s to some other vertex in $V \setminus V(T_{s,v})$ (that is $z_1 = s$ and $z_2 \in V \setminus V(T_{s,v})$). Note also that as $z_2 \in V \setminus V(T_{s,v})$ it follows that the shortest path from $s = z_1$ to z_2 in the tree T_s does not contain the edge e and hence the path $\tilde{e}$ represents also a path that exists in $G \setminus \{e\}$ and its weight is the distance of this path.

If $\tilde{e} \in E_e^1$ then notice that this edge also exists in $G \setminus \{e\}$ as required.

It follows that $d_e(s, t) \ge d_{s,t,e}$.

We get the following: (a) If $d_{s,t,e} > d(s, u) + 4\sqrt{n}$ then $d_e(s, t) > d(s, u) + 4\sqrt{n}$. (b) It is sufficient to prove that if $d_{s,t,e} \le d(s, u) + 4\sqrt{n}$ then $d_e(s, t) \le d_{s,t,e}$. In that case, since $d_e(s, t) \le d_{s,t,e}$ and $d_e(s, t) \ge d_{s,t,e}$ we have $d_e(s, t) = d_{s,t,e}$ as required.

Therefore, in the remaining of the proof we show that if $d_{s,t,e} \le d(s, u) + 4\sqrt{n}$ then $d_e(s, t) \le d_{s,t,e}$. Assume $d_{s,t,e} \le d(s, u) + 4\sqrt{n}$. Let P be a shortest path from s to t in $G \setminus \{e\}$. Let x be the last vertex along P such that $x \notin T_{s,v}$ (such a vertex exists since $s \notin T_{s,v}$) and denote by P_1 the subpath of P from s to x and by P_2 the subpath of P from x to t. Then, $w(P) = d_{s,t,e} = d_{s,x,e} + d_{x,t,e}, w(P_1) = d_{s,x,e}$ and $w(P_2) = d_{x,t,e}$.

Since x is the last vertex along P such that $x \notin T_{s,v}$ it follows that every edge of P_2 touches a vertex of $T_{s,v}$ and in addition note that every vertex z in P_2 satisfies $d_{s,z,e} \le d(s, u) + 4\sqrt{n}$. This implies that $E(P_2) \subseteq E_e^2$ and therefore P_2 is contained in G_e. Let P' be the path $(s, x) \circ P_2$. Since $x \notin T_{s,v}$ it follows that $(s, x) \in E_e^1$, we get that P' is a path from s to t in G_e and hence $d_e(s, t) \le w(P')$.

Since $x \notin T_{s,v}$ then the shortest path from s to x in T_s does not contain the edge $e = (u, v)$. Hence, $d_{s,x,e} = d(s, x)$. Therefore, $w(P') = w(s, x) + w(P_2) = d(s, x) + d_{x,t,e} = d_{s,x,e} + d_{x,t,e} = d_{s,t,e}$. Since P' is a path in G_e, $|P'| \le d_{s,t,e}$ and we get that $d_e(s, t) \le d_{s,t,e}$.

We are ready to prove the correctness of the computation of h_1.

LEMMA 5.3. *Equation 3.2 holds. That is, If $d(s, t) < d_{s,t,e} \le d(s, u) + 4\sqrt{n}$ then we have $h_1[t, e] = d_{s,t,e}$, and otherwise $h_1[t, e] = \infty$.*

Proof. First, assume $d(s, t) < d_{s,t,e} \le d(s, u) + 4\sqrt{n}$. According to Lemma 5.1 it must be that $t \in \bar{T}_{s,v}$. According to Lemma 5.2 if $d_{s,t,e} \le d(s, u) + 4\sqrt{n}$ then $d_e(s, t) = d_{s,t,e}$. Hence, if $d(s, t) < d_{s,t,e} \le d(s, u) + 4\sqrt{n}$ then we have $d(s, t) < d_e(s, t) \le d(s, u) + 4\sqrt{n}$ and when $d_e(s, t) = d(s, t)$ we set $h_1[t, e] = d_e(s, t) = d_{s,t,e}$.
Now assume it does not hold that $d(s, t) < d_{s,t,e} \le d(s, u) + 4\sqrt{n}$. In this case, we either have $d(s, t) = d_{s,t,e} \le d(s, u) + 4\sqrt{n}$ or $d_{s,t,e} > d(s, u) + 4\sqrt{n}$. If $d(s, t) = d_{s,t,e} \le d(s, u) + 4\sqrt{n}$ then according to Lemma 5.2, we also have $d(s, t) = d_e(s, t) \le d(s, u) + 4\sqrt{n}$ and we set $h_1[t, e] = \infty$. Furthermore, if $d_{s,t,e} > d(s, u) + 4\sqrt{n}$ then according to Lemma 5.2 we have $d_e(s, t) > d(s, u) + 4\sqrt{n}$ and then we set $h_1[t, e] = \infty$.

5.1 The Short Cycle Lemma

Let $x \in V$ and let $e = (u, v)$ be an edge on the shortest path from s to x such that u is closer to s than v (i.e., $d(s, u) = d(s, v) - 1$). In this Section we prove a sufficient condition for having $\min\{h_0[x, e], h_1[x, e]\} = d_{s,x,e}$. We prove that if e is contained in a short cycle C of length at most $2\sqrt{n}$ and $d(v, x) \le 2\sqrt{n}$ then either $h_0[x, e] = d_{s,x,e}$ or $h_1[x, 0] = d_{s,x,e}$.

LEMMA 5.4. *Let $x \in V$ and let P be a shortest path from s to x, let $e = (u, v) \in P$ be an edge in P such that u is closer to s than v (i.e., $d(s, u) = d(s, v) - 1$). Assume $d(v, x) \le 2\sqrt{n}$ and that e is contained in a cycle C of length at most $2\sqrt{n}$. Then either $h_0[x, e] = d_{s,x,e}$ or $h_1[x, e] = d_{s,x,e}$.*

Note that in the above lemma it was required to add also the case where $h_0[x, e] = d_{s,x,e}$, as it might be that $d(s, x) = d_{s,x,e}$ and then (s, x, e) belongs to Case 1 (the replaceable edge case) and not to Case 2 (the small fall case), and therefore $h_0[x, e] = d_{s,x,e}$ but $h_1[x, e] = \infty$.

Proof. If $d_{s,x,e} = d(s,x)$ then according to Lemma 4.1 it holds that $h_0[x,e] = d(s,x)$. Therefore, we assume $d_{s,x,e} > d(s,x)$ for the rest of the proof. We prove that $d_{s,x,e} \leq d(s,u) + 4\sqrt{n}$ and then by Lemma 5.3 it holds that $h_1[x,e] = d_{s,x,e}$.
Let P_1 be the subpath of P from s to u and let P_3 be the subpath of P from v to x, then $P = P_1 \circ e \circ P_3$. Since P is a shortest path then P is a simple path and thus $e \notin P_1, P_3$. Let P_2 be the path from u to v obtained by removing the edge e from the cycle C. It follows that P_2 is a path from u to v of length $|P_2| = |C| - 1 < 2\sqrt{n}$.
Let $P' = P_1 \circ P_2 \circ P_3$, then P' is a path from s to x avoiding e and $|P'| = |P_1| + |P_2| + |P_3| = d(s,u) + (|C| - 1) + d(v,x)$. Since $|C| \leq 2\sqrt{n}$ and $d(v,x) \leq 2\sqrt{n}$ it hods that $d_{s,x,e} \leq |P'| \leq d(s,u) + 4\sqrt{n}$, and the lemma follows.

6 Handling The Single Pivot Case

We compute a table h_2 of size $O(n^2)$ such that for every $t \in V, e \in E(T_s)$ Equation 3.3 holds, *i.e.*, we will have $h_2[t,e] = \min_{x \in B}\{\min\{h_0[x,e], h_1[x,e]\} + d(x,t) \mid e \text{ is among the } 2\sqrt{n} \text{ last edges of the path from } s \text{ to } x \text{ in } T_s\}$.

The procedure for computing h_2: First initialize h_2 such that $h_2[t,e] = \infty$ for all $t \in V, e \in E(T_s)$. We loop over every $t \in V, x \in B$ and every edge e that is among the last $2\sqrt{n}$ edges on the path from s to x in the tree T_s. At each iteration we set $h_2[t,e] = \min\{h_2[t,e], \min\{h_0[x,e], h_1[x,e]\} + d(x,t)\}$.

Lemma 6.1. *The above procedure computes h_2 in $\widetilde{O}(n^2)$ time (w.h.p.).*

Proof. There are $O(n)$ vertices $t \in V$, and $\widetilde{O}(\sqrt{n})$ vertices $x \in B$ (w.h.p.). In addition, for every vertex $x \in B$, there are $2\sqrt{n}$ edges e that are among the $2\sqrt{n}$ last edges on the path from s to x in the tree T_s. Therefore, there are $\widetilde{O}(n^2)$ triple t, x, e that the procedure above considers, and for each such triple is processed in constant time, thus the total processing time for computing h_2 is $\widetilde{O}(n^2)$ (w.h.p.).

In the remaining of this Section we prove that for every $t \in V, e \in E(T_s)$ we have $h_2[t,e] \geq d_{s,t,e}$ and when several sufficient conditions hold we have $h_2[t,e] = d_{s,t,e}$. We first prove that $h_2[t,e] \geq d_{s,t,e}$.

Lemma 6.2. *For every $t \in V, e \in E(T_s)$ it holds that $h_2[t,e] \geq d_{s,t,e}$.*

Proof. We prove that $\min\{h_0[x,e], h_1[x,e]\} + d(x,t) \geq d_{s,x,e} + d_{x,t,e}$ for every $x \in B$ and e among the last $2\sqrt{n}$ edges of the path from s to x in T_s.
According to Equation 3.1 it follows that $h_0[x,e] \geq d_{s,x,e}$ and according to Equation 3.2 it follows that $h_1[x,e] \geq d_{s,x,e}$. Hence, $\min\{h_0[x,e], h_1[x,e]\} \geq d_{s,x,e}$.

Let $P(s,t)$ be the shortest path from s to t in the tree T_s. Consider the two cases, either $e \in P(s,t)$ or $e \notin P(s,t)$.

If $e \in P(s,t)$ then according to Lemma 4.2 either $e \notin P(s,x)$ or $e \notin P(x,t)$. But we assume that e is among the last $2\sqrt{n}$ edges of the path from s to x in T_s, that is, $e \in P(s,x)$. It follows that e is not on the shortest path from x to t, and thus $d_{x,t,e} = d(x,t)$. Hence, $\min\{h_0[x,e], h_1[x,e]\} + d(x,t) \geq d_{s,x,e} + d(x,t) = d_{s,x,e} + d_{x,t,e} \geq d_{s,t,e}$, where the last inequality holds by the triangle inequality in the graph $G \setminus \{e\}$.

If $e \notin P(s,t)$ then $d_{s,t,e} = d(s,t)$ and then $\min\{h_0[x,e], h_1[x,e]\} + d(x,t) \geq d_{s,x,e} + d(x,t) \geq d(s,x) + d(x,t) \geq d(s,t) = d_{s,t,e}$, where the second inequality holds since distances after edge failures may only be longer then the distance before the failure (*i.e.*, $d_{s,x,e} \geq d(s,x)$) and the last inequality holds by the triangle inequality in the graph G.

In both cases we obtain that $\min\{h_0[x,e], h_1[x,e]\} + d(x,t) \geq d_{s,t,e}$ and thus $h_2[t,e] \geq d_{s,t,e}$.

6.1 The Short Detour Case

In this Section we prove that when $\text{Detour}_{s,t,e}$ is short (containing at most $\sqrt{n}$ edges) then with high probability we have $\min\{h_0[t,e], h_1[t,e], h_2[t,e]\} = d_{s,t,e}$. This case is one of the motivations for the single pivot case.

Lemma 6.3. *Let $e = (u,v)$ such that u is closer to s than v and e is on the path from s to t in T_s. Assume there exists a replacement path $P_{s,t,e}$ such that* $\text{Detour}_{s,t,e}$ *contains at most $\sqrt{n}$ edges. Then w.h.p. either $h_0[t,e] = d_{s,t,e}$ or $h_1[t,e] = d_{s,t,e}$ or $h_2[t,e] = d_{s,t,e}$.*

Proof. If $d_{s,t,e} = d(s,t)$ then according to Equation 3.1 it follows that $d_{s,t,e} = h_0[t,e]$. If $d(s,t) < d_{s,t,e} \leq d(s,u) + 4\sqrt{n}$ then according to Equation 3.2 it holds that $h_1[t,e] = d_{s,t,e}$.

We are therefore left with the case where $d_{s,t,e} > d(s,t)$ and $d_{s,t,e} > d(s,u) + 4\sqrt{n}$. For the rest of proof we assume that $d_{s,t,e} > d(s,t)$ and $d_{s,t,e} > d(s,u) + 4\sqrt{n}$ and prove that in this case we have $h_2[t,e] = d_{s,t,e}$. *I.e.*, we prove that in this case there exists a vertex $x \in B$ such that $d_{s,t,e} = h_1[x,e] + d(x,t)$ where e is among the last $2\sqrt{n}$ edges of the path from s to x in T_s.

Let $P_{s,t,e}$ be a shortest path in $G \setminus \{e\}$ whose detour part $\text{Detour}_{s,t,e}$ contains at most $\sqrt{n}$ edges. Since $e = (u,v) \notin P_{s,t,e}$ where u is closer to s than v, it follows that $|\text{CommonPref}_{s,t,e}| \leq d(s,u)$. Hence, $d_{s,t,e} = |P_{s,t,e}| = |\text{CommonPref}_{s,t,e}| + |\text{Detour}_{s,t,e}| + |\text{CommonSuff}_{s,t,e}| \leq d(s,u) + |\text{Detour}_{s,t,e}| + |\text{CommonSuff}_{s,t,e}| \leq d(s,u) + \sqrt{n} + |\text{CommonSuff}_{s,t,e}|$. Therefore, if $d_{s,t,e} > d(s,u) + 4\sqrt{n}$ it follows that $|\text{CommonSuff}_{s,t,e}| > 3\sqrt{n}$. According to Lemma 2.1, w.h.p. at least one of the vertices of $\text{CommonSuff}_{s,t,e}$ was sampled, or even more precisely, w.h.p. at least one of the first $\sqrt{n}$ vertices on the subpath $\text{CommonSuff}_{s,t,e}$ was sampled, and let $x \in B \cap \text{CommonSuff}_{s,t,e}$ be the first sampled vertex along $\text{CommonSuff}_{s,t,e}$. Let $P_{s,x,e}$ be the subpath of $P_{s,t,e}$ from s to x, note that $P_{s,x,e}$ is a valid replacement from s to x that avoids e. As mentioned above, w.h.p, the subpath of $\text{CommonSuff}_{s,t,e}$ from its beginning until x (this subpath is $\text{CommonSuff}_{s,x,e}$) contains at most $\sqrt{n}$ edges with high probability (*i.e.*, $|\text{CommonSuff}_{s,x,e}| \leq \sqrt{n}$ w.h.p.).

Since $P_{s,t,e}$ and $P_{s,x,e}$ have the same prefix and detour, it holds that $|\text{CommonPref}_{s,x,e}| \leq d(s,u)$

and $|\text{Detour}_{s,x,e}| \leq \sqrt{n}$. It follows that $d_{s,x,e} = |\text{CommonPref}_{s,x,e}| + |\text{Detour}_{s,x,e}| + |\text{CommonSuff}_{s,x,e}| \leq d(s,u) + |\text{Detour}_{s,x,e}| + |\text{CommonSuff}_{s,x,e}| \leq d(s,u) + 2\sqrt{n}$ w.h.p. According to Equation 3.2 this implies that $h_1[x,e] = d(s,x,e)$ w.h.p.

Since $x \in P_{s,t,e}$ it holds that $d_{s,t,e} = d_{s,x,e} + d_{x,t,e}$ and since $x \in \text{CommonSuff}_{s,t,e}$ it holds that x appears after e along $P(s,t)$. Hence, the subpath of $P(s,t)$ from x to t does not contain e, and thus $d_{x,t,e} = d(x,t)$. We get that $d_{s,t,e} = d_{s,x,e} + d(x,t) = h_1[x,e] + d(x,t)$, as required.

We are still left to prove that e is among the last $2\sqrt{n}$ edges of the path from s to x in T_s. Since $x \in \text{CommonSuff}_{s,t,e}$ then by definition of $\text{CommonSuff}_{s,t,e}$, e must be on the path from s to x in T_s. Since $d_{s,x,e} \leq d(s,u) + 2\sqrt{n}$ and $e = (u,v)$ is on the path from s to x in T_s, it follows that e is among the last $2\sqrt{n}$ edges of the path from s to x in T_s.

7 The Algorithm For Handling The Double Pivot Case

Let $t \in V, e \in E(T_s)$. We first define when (t,e) is considered as a "double pivot" type, and then show how to find $d_{s,t,e}$ for every (t,e) which is of "double pivot" type.

Definition 7.1. *Let $t \in V, e \in E(T_s)$. We say that (t,e) is of type "double pivot" if the following conditions hold: (1) There exists a replacement path $P_{s,t,e}$ and $x, y \in B$ such that both x and y appear on the detour part of $P_{s,t,e}$ and in addition x appears before y on the detour part of $P_{s,t,e}$ or $x = y$. (2) e is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable.*

Observe that we allow $x = y$ to be the same vertex of B. For the sake of abbreviation, in the rest of the text when we say that x appears before y on the detour part of $P_{s,t,e}$ we mean that either x appears strictly before y on the detour or that $x = y$.

In this section we describe how to compute in $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.) a table h_3 such that:

$$h_3[t,e] = \begin{cases} d_{s,t,e}, & \text{if } (t,e) \text{ is of type "double pivot"} \\ \geq d_{s,t,e}, & \text{otherwise} \end{cases}$$

First, for every vertex $x \in B$ we preprocess the replaceability data-structure R_x according to Theorem 2.1. This takes $\widetilde{O}(m\sqrt{n})$ time (w.h.p.): there are $\widetilde{O}(\sqrt{n})$ vertices $x \in B$ (w.h.p.), and each replaceability oracle R_x is preprocessed in time $\widetilde{O}(m)$ as described in Theorem 2.1.

To give some intuition, we first describe a simple $\widetilde{O}(n^3)$ time (w.h.p). algorithm for computing $h'_3[t,e]$ for every pair (t,e). Here we compute $h'_3[t,e] = \min_{x,y \in B}\{d(s,x) + d(x,y) + d(y,t) | e \text{ is } (s,x)-\text{replaceable}, (x,y)-\text{replaceable and } (y,t)-\text{replaceable}\}$. We initialize the table h'_3 of size $O(n^2)$ where every entry $t \in V, e \in E(T_s)$ is set to infinity $h'_3[t,e] = \infty$. Then we loop over every pair of pivots $x, y \in B$ (observe that we allow $x = y$ to be the same vertex of B, and there are $\widetilde{O}(n)$ pairs of pivots $B \times B$ w.h.p.) and every vertex $t \in V$ (there are n vertices in V) and every edge $e \in P(s,t)$ (there are $O(n)$ edges along $P(s,t)$). Thus, w.h.p. there are $\widetilde{O}(n^3)$ such quadruples x,y,t,e. For every such quadruple x,y,t,e we check in constant time (using the data-structures R_x, R_y) if e is (s,x)-replaceable, (x,y)-replaceable, and (y,t)-replaceable, and if so we set $h'_3[t,e] \leftarrow \min\{h'_3[t,e], d(s,x) + d(x,y) + d(y,t)\}$.

Lemma 7.1. *If (t,e) is of type "double pivot" then $h'_3[t,e] = d_{s,t,e}$, and otherwise $h'_3[t,e] \geq d_{s,t,e}$.*

Proof. We first prove that $h'_3[t,e] \geq d_{s,t,e}$. If $h'_3[t,e] = \infty$ then it is trivial that $h'_3[t,e] \geq d_{s,t,e}$, we assume $h'_3[t,e] < \infty$ and prove that $h'_3[t,e] \geq d_{s,t,e}$. Let $x, y \in B$ such that e is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable it follows that $d(s,x) + d(x,y) + d(y,t) = d_{s,x,e} + d_{x,y,e} + d_{y,t,e}$. According to the triangle inequality in the graph $G \setminus \{e\}$ it follows that $d_{s,x,e} + d_{x,y,e} + d_{y,t,e} \geq d_{s,t,e}$ for every $t \in V, e \in E(T_s)$. Hence, $h'_3[t,e] = \min_{x,y \in B}\{d(s,x) + d(x,y) + d(y,t) | e \text{ is } (s,x)-\text{replaceable}, (x,y)-\text{replaceable and } (y,t)-\text{replaceable}\} \geq d_{s,t,e}$.

We now assume that (t,e) is of type "double pivot" and prove that $h'_3[t,e] = d_{s,t,e}$. Since (t,e) is of type "double pivot", then x appears before y on the detour part of some replacement path $P_{s,t,e}$ such that e is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable. Since x appears before y on the detour part of some replacement path $P_{s,t,e}$ it follows that $d_{s,t,e} = d_{s,x,e} + d_{x,y,e} + d_{y,t,e}$. Since e is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable it follows that $d_{s,x,e} + d_{x,y,e} + d_{y,t,e} = d(s,x) + d(x,y) + d(y,t)$. It follows that $h'_3[t,e] \leq d_{s,t,e}$ and also $h'_3[t,e] \geq d_{s,t,e}$ and we get $h'_3[t,e] = d_{s,t,e}$.

In order to reduce the running time from $\widetilde{O}(n^3)$ to $\widetilde{O}(m\sqrt{n} + n^2)$ we cannot afford spending even a constant time on every quadruple x,y,t,e. Instead, we show that using more sophisticated data-structures and new ideas it is sufficient to spend time only on triples x,y,t such that $x,y \in B$ and $t \in V$. Note that w.h.p. there are $\widetilde{O}(n)$ pairs of pivots $(x,y) \in B \times B$, and $O(n)$ vertices $t \in V$. Hence, w.h.p. there are only $\widetilde{O}(n^2)$ such triples x,y,t and we process all these triples in $\widetilde{O}(n^2)$ total time. Computing the replaceability oracles R_x for every $x \in B$ takes w.h.p. additional $\widetilde{O}(m\sqrt{n})$ time. In order to process only triples x,y,t and not quadruples x,y,t,e we need, somehow, to handle many edges $e \in P(s,t)$ at once. To do so, we define path intervals $I_{x,y,t}$, which are subpaths of $P(s,t)$ such that we handle all the edges of every path interval at once.

7.1 Path Intervals

As in the simple $\widetilde{O}(n^3)$ time algorithm described above, we would like to find all quadruples x,y,t,e such that e is (s,x)-replaceable, (x,y)-replaceable, and (y,t)-replaceable. Consider an edge e and a vertex t such that (t,e) is of type double pivot. Let x and y be the vertices as defined in Definition 7.1. We say that (e,t) is "(x,y)-double pivot". Instead of actually finding all quadruples x,y,t,e such that e is (s,x)-replaceable, (x,y)-replaceable, and (y,t)-replaceable, for all vertices $x, y \in B$ and $t \in V$, we show how to find a subpath of $P(s,t)$ such that all edges e in this subpath are (s,x)-replaceable, (x,y)-replaceable, and (y,t)-replaceable, and in addition all edges e that are "(x,y)-double pivot" are contained in this subpath. This will be enough for our needs.

Loosely speaking, we will show that the the following subpath satisfies these properties. We are looking for the maximum subpath $I_{x,y,t}$ (hereafter referred as path interval) such that there is a replacement path of length $d(s,x) + d(x,y) + d(y,t)$ from s to t passing through x and y that excludes the entire path $I_{x,y,t}$. Note that there may be many replacement paths from s to t that go through x and y (all which are of length $d(s,x)+d(x,y)+d(y,t)$). However some of these replacements paths exclude a smaller subset of edges of $P(s,t)$ and we want to find the one that excludes the most number of edges of $P(s,t)$ (we will show that such a path exists). We next define the indices i_1 and i_2 such that the subpath of $P(s,t)$ from v_{i_1} to v_{i_2} is exactly the interval $I_{x,y,t}$.

Let $t \in V, x, y \in B$ (recall that we allow $x = y$ to be the same vertex of B). Consider the enumeration of the vertices of $P(s,t) = (v_0, \ldots, v_{d(s,t)})$. Let $0 \le i_1 \le d(s,t)$ be the minimum index such that (v_i, v_{i+1}) is (s,x)-replaceable for every $i \ge i_1$. Let $1 \le i_2 \le d(s,t)$ be the maximum index such that (v_{i-1}, v_i) is (y,t)-replaceable for every $i \le i_2$. We define the path interval $I_{x,y,t} = \{(v_i, v_{i+1}) | i_1 \le i < i_2\}$. Note that if $i_2 \le i_1$ then $I_{x,y,t} = \emptyset$. Since all the edges (v_i, v_{i+1}) for every $i \ge i_1$ are (s,x)-replaceable, all the edges (v_i, v_{i+1}) for every $i < i_2$ are (y,t)-replaceable, it follows that $I_{x,y,t}$ is a (possibly empty) subpath of $P(s,t)$ of consecutive edges, and every edge $e \in I_{x,y,t}$ is both (s,x)-replaceable and (y,t)-replaceable. We say that $I_{x,y,t}$ is a *valid* path interval if all the edges $e \in P(s,t)$ are (x,y)-replaceable. Since $I_{x,y,t} \subseteq P(s,t)$ it follows that if $I_{x,y,t}$ is a valid path interval, then all of its edges $e \in I_{x,y,t}$ are (x,y)-replaceable (as every edge in $I_{x,y,t}$ is also in $P(s,t)$). We get that if $I_{x,y,t}$ is a valid path interval, then all of its edges $e \in I_{x,y,t}$ are (s,x)-replaceable, (x,y)-replaceable, and (y,t)-replaceable. See Figure 2 which illustrates a path interval $I_{x,y,t}$.

Finding the indices i_1 and i_2 turned out to be quite complicated and technical. One may tempt to find the index i_1 using a binary search, by searching for the first edge e (closest to s) on the path $P(s,t)$ such that e is (s,x)-replaceable. This would work if magically we would have an index i such that all edges appearing before v_i on the path $P(s,t)$ are not (s,x)-replaceable and all edges appearing after are (s,x)-replaceable. Unfortunately, this is not necessarily the case, as there might be some subpath of $P(s,t)$ such that all of its edges are (s,x)-replaceable and then a subpath where all of its edges are not and then again a subpath of replaceable edges and so on (for instance if there is a subpath $P(u,v)$ of $P(s,t)$ from some vertex u to some vertex v where there is a completely disjoint subpath from u to v of the same length). See Figure 3 as an example. Section 8 is devoted for efficient computation of i_1 and i_2.

We next prove some properties about path intervals.

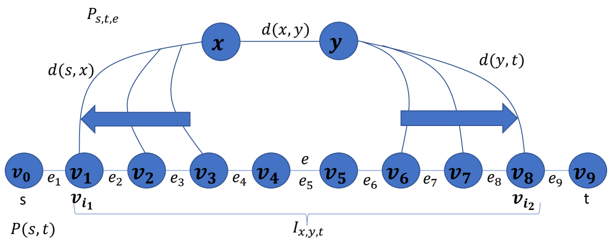
$P_{s,t,e}$
$d(x,y)$
x
y
$d(s,x)$
$d(y,t)$
v_0 e_1 v_1 e_2 v_2 e_3 v_3 e_4 v_4 e e_5 v_5 e_6 v_6 e_7 v_7 e_8 v_8 e_9 v_9
s
v_{i_1}
v_{i_2}
t
$P(s,t)$
$I_{x,y,t}$

Figure 2: An illustration of the path interval $I_{x,y,t}$ where $x, y \in B$ and $t \in V$. It can be observed from the figure that, intuitively, the path interval $I_{x,y,t}$ is a maximum subpath of $P(s,t)$ such that there is a replacement path $(P_{s,t,e})$ of length $d(s,x) + d(x,y) + d(y,t)$ from s to t passing through x and y that excludes the entire subpath $I_{x,y,t}$

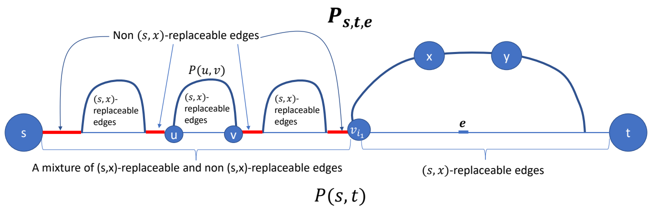

Figure 3: An illustration of the path interval $I_{x,y,t}$ where $x, y \in B$ and $t \in V$. In this figure there are several shortest paths from s to x, one that leaves $P(s,t)$ on vertex v_3, one that leaves $P(s,t)$ on vertex v_2 and one the leaves $P(s,t)$ on vertex v_1. The vertex v_{i_1} is defined to be the first vertex such that there is a shortest path from s to x that leaves $P(s,t)$ on the vertex v_{i_1}, in the figure above $v_{i_1} = v_1$. Similarly, $v_{i_2} = v_8$.

LEMMA 7.2. *Let $e \in P(s,t), t \in V$ such that (t,e) is of double pivot type. Let $x, y \in B$ be the two vertices in B such that 1. there exists a replacement path $P_{s,t,e}$ and x appears before y on the detour part of $P_{s,t,e}$ and 2. e is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable. (two such vertices exist by the definition of double pivot type).*

Then $e \in I_{x,y,t}$ and $I_{x,y,t}$ is a valid path interval.

Proof. We first prove that $e \in I_{x,y,t}$.
Let $0 \le i < d(s,t)$ be the index such that $e = (v_i, v_{i+1})$ (recall that $P(s,t) = (v_0, \ldots, v_{d(s,t)})$). Since e is (s,x)-replaceable then the subpath $P(s,x)$ of $P_{s,t,e}$ from s to x is also a shortest path from s to x in the original graph G. Since x is on the detour part of $P_{s,t,e}$, it follows that $P(s,x)$ does not contain any of the edges $X = \{(v_j, v_{j+1}) | i \le j < d(s,t)\}$, and hence all the edges of X are (s,x)-replaceable. Therefore by definition of i_1 we have $i_1 \le i$ (see Figure 4).

Similarly, since e is (y,t)-replaceable then the subpath $P(y,t)$ of $P_{s,t,e}$ from y to t is also a shortest path from y

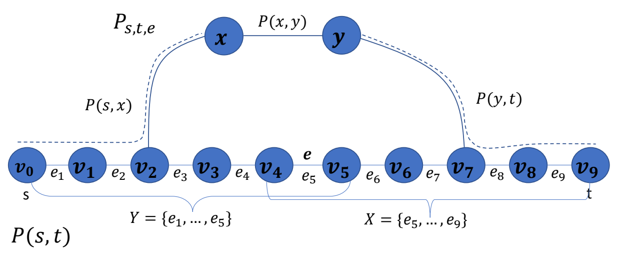
$P_{s,t,e}$
x
$P(x,y)$
y
$P(s,x)$
$P(y,t)$
e
v_0 e_1 v_1 e_2 v_2 e_3 v_3 e_4 v_4 e_5 v_5 e_6 v_6 e_7 v_7 e_8 v_8 e_9 v_9
s
t
$Y = \{e_1, \dots, e_5\}$
$X = \{e_5, \dots, e_9\}$
$P(s,t)$

Figure 4: An illustration of the proof of Lemma 7.2. Here, $P(s,t) = (v_0, \dots, v_9)$, the edge $e = (v_i, v_{i+1}) = (v_4, v_5)$. All the edges $X = \{e_5, \dots, e_9\}$ are not on the shortest path $P(s,x)$ and thus all the edges of X are (s,x)-replaceable. All the edges $Y = \{e_1, \dots, e_5\}$ are not on the shortest path $P(y,t)$ and thus all the edges of Y are (y,t)-replaceable.

to t in the original graph. Since y is on the detour part of $P_{s,t,e}$, it follows that $P(y,t)$ does not contain any of the edges $Y = \{(v_j, v_{j+1}) | 0 \le j \le i\}$, and hence all the edges of Y are (y,t)-replaceable. Therefore, $i_2 > i$.

Since $e = (v_i, v_{i+1})$ with $i_1 \le i < i_2$, it follows that $e \in I_{x,y,t} = \{(v_j, v_{j+1}) | i_1 \le j < i_2\}$.

We now prove that $I_{x,y,t}$ is a valid path interval. Since x appears before y on the detour part of $P_{s,t,e}$ and e is (x,y)-replaceable then the subpath $P(x,y)$ of $P_{s,t,e}$ from x to y is a shortest path from x to y in the original graph G. Therefore, the entire path $P(x,y)$ is contained in the detour part of $P_{s,t,e}$. Hence all the edges of $P(s,t)$ are edge-disjoint from $P(x,y)$, i.e., all the edges of $P(s,t)$ are (x,y)-replaceable and therefore $I_{x,y,t}$ is a valid path interval.

We assign a weight for every valid path interval $w(I_{x,y,t}) = d(s,x) + d(x,y) + d(y,t)$. If $I_{x,y,t} = \emptyset$ or $I_{x,y,t}$ is not valid, then we assign its weight $w(I_{x,y,t}) = \infty$.

Lemma 7.3. *Let $e \in I_{x,y,t}$ then $w(I_{x,y,t}) \ge d_{s,t,e}$.*

Proof. If $I_{x,y,t}$ is not a valid path interval then $w(I_{x,y,t}) = \infty \ge d_{s,t,e}$. So, assume $I_{x,y,t}$ is a valid path interval. Then, every edge of $I_{x,y,t}$ is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable, and in particular, e is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable. Therefore, $d_{s,x,e} = d(s,x), d_{x,y,e} = d(x,y)$ and $d_{y,t,e} = d(y,t)$. By substitution we get $w(I_{x,y,t}) = d(s,x) + d(x,y) + d(y,t) = d_{s,x,e} + d_{x,y,e} + d_{y,t,e}$. According to the triangle inequality in the graph $G \setminus \{e\}$ it holds that $w(I_{x,y,t}) = d_{s,x,e} + d_{x,y,e} + d_{y,t,e} \ge d_{s,t,e}$.

Let $e \in E(T_s)$. We define $I(e,t) = \{I_{x,y,t} | e \in I_{x,y,t}\}$. We define $h_3[t,e]$ as follows.

$$h_3[t,e] = \min_{I_{x,y,t} \in I(e,t)} \{w(I_{x,y,t})\}. \tag{7.5}$$

According to Lemma 7.3 it follows that $h_3[t,e] \ge d_{s,t,e}$. We show that in the double pivot case, it holds that $h_3[t,e] = d_{s,t,e}$.

Lemma 7.4. *Let $t \in V$, $e \in P(s,t)$ such that (t,e) is of double pivot type. Then $h_3[t,e] = d_{s,t,e}$.*

Proof. In Lemma 7.3 we have already proved that $h_3[t,e] \ge d_{s,t,e}$. Since $h_3[t,e] = \min_{I_{x,y,t} \in I(e,t)} \{w(I_{x,y,t})\}$, we only need to prove that there exists a path interval $I_{x,y,t} \in I(e,t)$ such that $d_{s,t,e} = w(I_{x,y,t})$.

Since (t,e) is of double pivot type then there exists a replacement path $P_{s,t,e}$ and $x, y \in B$ such that both x and y appear on the detour part of $P_{s,t,e}$ and in addition x appears before y on the detour part of $P_{s,t,e}$ or $x = y$. and e is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable. by Lemma 7.2 it follows that $e \in I_{x,y,t}$ and $I_{x,y,t}$ is a valid path interval. Therefore, $I_{x,y,t} \in I(e,t)$ and $w(I_{x,y,t}) = d(s,x) + d(x,y) + d(y,t)$.

By definition 7.1, there exists a replacement path $P_{s,t,e}$ such that x appears before y on the detour part of $P_{s,t,e}$ and e is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable. Since x appears before y on the detour part of $P_{s,t,e}$ then $d_{s,t,e} = d_{s,x,e} + d_{x,y,e} + d_{y,t,e}$. Since e is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable it also follows that $d_{s,x,e} + d_{x,y,e} + d_{y,t,e} = d(s,x) + d(x,y) + d(y,t)$. Therefore, $w(I_{x,y,t}) = d(s,x) + d(x,y) + d(y,t) = d_{s,x,e} + d_{x,y,e} + d_{y,t,e} = d_{s,t,e}$, and the Lemma follows.

Every path interval $I_{x,y,t}$ is defined using two indices $0 \le i_1, i_2 \le d(s,t)$ such that $I_{x,y,t} = \{(v_i, v_{i+1}) | i_1 \le i < i_2\}$. Let $i_1(I_{x,y,t})$ and $i_2(I_{x,y,t})$ be the two indices $0 \le i_1, i_2 \le d(s,t)$ associated with the path interval $I_{x,y,t}$.

In Section 8 we describe how to compute all the path intervals $I_{x,y,t}$ for every $x, y \in B, t \in V$ in $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.). As this Section is long, we leave it towards the end, and currently just state the Lemma we prove there.

Lemma 7.5. *One can compute all the path intervals $I_{x,y,t}$ for every $x, y \in B, t \in V$ in $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.). I.e., one can compute all the indices $i_1(I_{x,y,t})$ and $i_2(I_{x,y,t})$ for every $x, y \in B, t \in V$ in $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.).*

7.2 Efficient Testing Of Path Intervals Validity

In this Section we assume that the algorithm already computed the path intervals $I_{x,y,t}$ for every $x, y \in B, t \in V$ and prove the following Lemma.

Lemma 7.6. *We can check for all $x, y \in B, t \in V$ whether or not the path interval $I_{x,y,t}$ is a valid path interval, in total $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.).*

For every $x, y \in B$ we create a tree $T_{x,y}$ which is a weighted copy of T_s. The weight of the edge $e \in T_{x,y}$ is set to 1 if e is not (x,y)-replaceable edge, and otherwise its weight in $T_{x,y}$ is set to 0. Finally, for every $t \in V$ we set a bit $b_{x,y}(t) = 1$ if the path from s to t in $T_{x,y}$ contains at least one edge whose weight is 1, otherwise we set $b_{x,y}(t) = 0$. The n bits $\{b_{x,y}(t) | t \in V\}$ can be computed by a pre-order (or DFS) scan of the tree $T_{x,y}$ in $O(n)$ time: during the scan, whenever we encounter an edge whose weight is 1, then for every vertex t in its subtree we set $b_{x,y}(t) = 1$.

Lemma 7.7. *Let $x, y \in B, t \in V$, then $b_{x,y}(t) = 0$ iff every edge $e \in P(s,t)$ is (x,y)-replaceable. Furthermore, we can compute all the bits $b_{x,y}(t)$ in all the trees $T_{x,y}$ in $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.).*

Proof. If $b_{x,y}(t) = 0$ then all the edges on the path from s to t in $T_{x,y}$ have weight 0, which means that they are (x,y)-replaceable. If $b_{x,y}(t) = 1$ then there exists an edge e on the path from s to t in $T_{x,y}$ whose weight is 1, which means that e is not (x,y)-replaceable.

We now describe the runtime of the algorithm. Constructing all the replaceability oracles R_x for every $x \in B$ takes $\widetilde{O}(m\sqrt{n})$ time, as there are $\widetilde{O}(\sqrt{n})$ vertices $x \in B$, an constructing one replacebility oracle R_x takes $O(m)$ time according to Theorem 2.1. Then, for every pair of pivots $x, y \in B$ we generate a tree $T_{x,y}$ which is a copy of T_s in $O(n)$ time, and for every edge $e \in T_{x,y}$ we use the reachability oracle R_x to check in constant time if e is (x,y)-replaceable. If so, we set $w(e) = 0$ in the tree $T_{x,y}$, otherwise we set $w(e) = 1$. This construction of the tree $T_{x,y}$ takes $O(n)$ time. Then, we compute the bits $b_{x,y}(t)$ using a pre-order scan of the tree $T_{x,y}$ which again takes $O(n)$ time. Since there are w.h.p. $\widetilde{O}(n)$ pairs of pivots $x, y \in B$, and for every pair of pivots building the tree $T_{x,y}$ and computing the bits $b_{x,y}(t)$ takes $O(n)$ time, we get that the total runtime (after the construction of the replaceability oracles R_x) is $\widetilde{O}(n^2)$ (w.h.p.). All together, we get $\widetilde{O}(m\sqrt{n} + n^2)$ time.

Lemma 7.7 proves Lemma 7.6, as $I_{x,y,t}$ is valid iff $b_{x,y}(t) = 1$, and the computation of all the bits $b_{x,y}(t)$ takes $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.).

7.3 Combining The Components For An Efficient Algorithm For Computing h_3 We are now ready to describe the algorithm for computing h_3 in $\widetilde{O}(m\sqrt{n} + n^2)$ time.

Recall that according to Equation 7.5 we defined h_3 for every $t \in V, e \in E(T_s)$ as $h_3[t,e] = \min_{I_{x,y,t} \in I(e,t)}\{w(I_{x,y,t})\}$.

First, for every $x, y \in B$ and for every $t \in V$ we compute the two indices $i_1(I_{x,y,t}), i_2(I_{x,y,t})$ as in Lemma 7.5 in total $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.), which gives us all the path intervals $\{I_{x,y,t} | x, y \in B, t \in V\}$. Then, for every $x, y \in B, t \in V$ we compute the bits $b_{x,y}(t)$ as in Lemma 7.7 in $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.).

We describe an algorithm, whose input is: (1) The vertex $t \in V$. (2) For every $x, y \in B$ we are given the path interval $I_{x,y,t}$ and the bit $b_{x,y}(t)$. There are $\widetilde{O}(n)$ pairs $x, y \in B$ (w.h.p.), and thus for the given vertex $t \in V$ there are $\widetilde{O}(n)$ path intervals $I_{x,y,t}$ and bits $b_{x,y}(t)$. Recall that the weight of the path interval $I_{x,y,t}$ is defined as $w(I_{x,y,t}) = d(s,x) + d(x,y) + d(y,t)$ if $b_{x,y}(t) = 1$ and otherwise $w(I_{x,y,t}) = \infty$.

Note that computing $h_3[t,e]$ for every $e \in P(s,t)$ by iterating over all intervals $I_{x,y,t}$ such that $e \in I_{x,y,t}$ is too slow for our needs. We therefore present a more efficient algorithm such that for the given vertex $t \in V$, the algorithm described hereinafter computes the values $h_3[t,e] = \min_{e \in I_{x,y,t}} w(I_{x,y,t})$ for all $e \in P(s,t)$ in total $\widetilde{O}(n)$ time (w.h.p.). Thus, given $\{I_{x,y,t}, w(I_{x,y,t}) | x, y \in B, t \in V\}$ the total processing time to compute $h_3[t,e]$ for all $t \in V, e \in P(s,t)$ is $\widetilde{O}(n^2)$ time (w.h.p.).

We use in our algorithm the classic data structure of binary search tree (BST) that maintains a set of numbers and supports operations like: insert, delete, search, successor, predecessor in $O(\log n)$ time per operation (see e.g. [9])

Following is the description of the algorithm for a fixed vertex $t \in V$. Consider again the enumeration of the vertices of $P(s,t) = (v_0, \ldots, v_{d(s,t)})$ and recall that $I_{x,y,t} = \{(v_i, v_{i+1}) | i_1 \le i < i_2\}$. We sort the path intervals $I_{x,y,t}$ in ascending order of their weights $w(I_{x,y,t})$. Let $I_1, \ldots, I_k$ be all the path intervals $\{I_{x,y,t} | x, y \in B\}$ according to the sorted order $w(I_1) \le w(I_2) \le \ldots \le w(I_k)$. For a path interval I_j $(1 \le j \le k)$ we denote by $i_1^j = i_1(I_j)$ the index of the first edge along $P(s,t)$ which belongs to I_j and by $i_2^j = i_2(I_j)$ the index of the last edge along $P(s,t)$ which belongs to I_j (these are the indices i_1, i_2 of the path interval I_j, i.e., $I_j = \{(v_i, v_{i+1}) | i_1 \le i \le i_2\}$). We create a binary search tree (BST) B_t and insert to it the $O(n)$ edges $e_1, \ldots, e_{d(s,t)}$ along the path $P(s,t)$ such that their key equals to their index (here the index and the key of the edge $e_i \in \{e_1, \ldots, e_{d(s,t)}\}$ is i).

We process the path intervals $I_1, \ldots, I_k$ in this order as follows. When processing the j^{th} path interval I_j we search in the BST B_t for all the edges whose index is between i_1^j to i_2^j. For every such edge $e \in I_j$ we set $h_3[t,e] = w(I_j)$ and remove e from the BST B_t. Observe that searching for the edge whose key is the minimum key which is at least i_1^j is exactly a successor(i_1^j) search in the tree B_t, and this can be done $O(\log n)$ time. Thus, we can find every edge e in the BST B_t whose key is between i_1^j to i_2^j in $O(\log n)$ time per edge.

When the above loop is finished, it is easy to verify that every edge $e \in P(s,t)$ was assigned the weight $w(I_j)$ of the minimum index j such that $e \in I_j$, which means that $h_3[t,e]$ is the minimum weight of a path interval containing e, as required.

Theorem 7.1. *One can compute all the values $h_3[t,e] = \min_{e \in I_{x,y,t}} w(I_{x,y})$ for all $t \in V, e \in P(s,t)$ in $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.). When (t,e) is of "double pivot" type it holds that $h_3[t,e] = d_{s,t,e}$ and otherwise $h_3[t,e] \ge d_{s,t,e}$.*

Proof. Finding all intervals $I_{x,y,t}$ takes $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.) as in Lemma 7.5, computing all the bits $b_{x,y}(t)$ takes $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.) as in Lemma 7.7. Then, apply the above algorithm for every $t \in V$ to find $h_3[t,e]$ for every $t, e \in P(s,t)$ using additional $\widetilde{O}(n^2)$ time (w.h.p.).

When (t,e) is of "double pivot" type then according to Definition 7.1 there exists a replacement path $P_{s,t,e}$ such that x appears before y on the detour part of $P_{s,t,e}$ and e is (s,x)-replaceable, (x,y)-replaceable and (y,t)-replaceable. Therefore, according to Lemma 7.4 it holds that $h_3[t,e] = d_{s,t,e}$.

If (t,e) is not of "double pivot" type then according to Lemma 7.3 it holds that $h_3[t,e] \ge d_{s,t,e}$.

8 An Efficient Algorithm For Computing The Path Intervals $I_{x,y,t}$

In this Section we prove Lemma 7.5, *i.e.*, we describe how to compute all the path intervals $I_{x,y,t}$ for every $x, y \in B, t \in V$ in $\widetilde{O}(m\sqrt{n} + n^2)$ time (w.h.p.).

We start by computing, for every vertex $y \in B$, a BFS tree T_y with source vertex y in the graph G. This takes $\widetilde{O}(m\sqrt{n})$ time (w.h.p.).

Let $t \in V$, let $P(s,t)$ be the shortest path from s to t in T_s, and let $x, y \in B$ such that $x, y \notin P(s,t)$. As before, consider the enumeration of the vertices of $P(s,t) = (v_0, \ldots, v_{d(s,t)})$. Recall that: (a) $0 \leq i_1 \leq d(s,t)$ is the minimum index such that (v_i, v_{i+1}) is (s,x)-replaceable for every $i \geq i_1$. (b) $1 \leq i_2 \leq d(s,t)$ is the maximum index such that (v_{i-1}, v_i) is (y,t)-replaceable for every $i \leq i_2$. (c) $I_{x,y,t} = \{(v_i, v_{i+1}) | i_1 \leq i < i_2\}$.

Thus, in order to compute the path intervals $I_{x,y,t}$ we need to find the indices i_1, i_2.

We describe how to find the index i_2. Finding the index i_1 can be done similarly. We first define an index $0 \leq i'_2 \leq i_2$, we find i'_2 using a binary search on the vertices of $P(s,t)$ in $O(\log n)$ time. Then, using the index i'_2 and a dominator tree rooted at y we find the index i_2.

We start by defining the index i'_2 and prove that we can find i'_2 in $O(\log n)$ time using a binary search on the vertices of $P(s,t)$. We obtain this using a monotonicity property of the vertices of $P(s,t)$ (see Figure 5), as described in the following Lemma.

Lemma 8.1. *Let $y \in B$. There exists an index $0 \leq i'_2 \leq d(s,t)$ such that for every $0 \leq i < i'_2$ it holds that $d(y,t) < d(y,v_i) + d(v_i,t)$ and for every $i'_2 \leq i \leq d(s,t)$ it holds that $d(y,t) = d(y,v_i) + d(v_i,t)$. Furthermore, we can find the index i'_2 in $O(\log n)$ time. See Figure 5.*

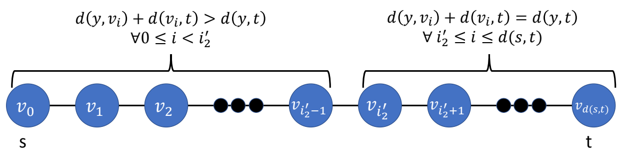
$d(y,v_i) + d(v_i,t) > d(y,t)$
$\forall 0 \leq i < i'_2$
$d(y,v_i) + d(v_i,t) = d(y,t)$
$\forall i'_2 \leq i \leq d(s,t)$
v_0
v_1
v_2
$v_{i'_2-1}$
$v_{i'_2}$
$v_{i'_2+1}$
$v_{d(s,t)}$
s
t

Figure 5: The monotonicity property that for every $0 \leq i < i'_2$ it holds that $d(y,t) < d(y,v_i) + d(v_i,t)$ and for every $i'_2 \leq i \leq d(s,t)$ it holds that $d(y,t) = d(y,v_i) + d(v_i,t)$. Using a binary search on the index $i \in [0, d(s,t)]$ we find the minimum index i'_2 such that $d(y,t) = d(y,v_{i'_2}) + d(v_{i'_2},t)$.

Proof. Let $0 \leq i'_2 \leq d(s,t)$ be the minimum index such that $d(y,t) = d(y,v_{i'_2}) + d(v_{i'_2},t)$. Such an index i'_2 exists as the equation $d(y,t) = d(y,v_{i'_2}) + d(v_{i'_2},t))$ trivially holds when $i'_2 = d(s,t)$, since then $t = v_{i'_2}$. By the minimality of i'_2, it follows that for every $0 \leq i < i'_2$ it holds that $d(y,t) \neq d(y,v_i) + d(v_i,t)$. According to the triangle inequality $d(y,t) \leq d(y,v_i) + d(v_i,t)$, and hence for every $0 \leq i < i'_2$ it holds that $d(y,t) < d(y,v_i) + d(v_i,t)$.

We are left to prove that for every $i'_2 \leq i \leq d(s,t)$ it holds that $d(y,t) = d(y,v_i) + d(v_i,t)$. Consider an index i such that $i'_2 \leq i \leq d(s,t)$.

We claim that $d(y,v_i) = d(y,v_{i'_2}) + d(v_{i'_2},v_i)$. To see this assume, toward contradiction, that $d(y,v_i) < d(y,v_{i'_2}) + d(v_{i'_2},v_i)$. We get the following $d(y,t) = d(y,v_{i'_2}) + d(v_{i'_2},t) = d(y,v_{i'_2}) + d(v_{i'_2},v_i) + d(v_i,t) > d(y,v_i) + d(v_i,t) \geq d(y,t)$, contradiction.

We therefore get, $d(y,t) = d(y,v_{i'_2}) + d(v_{i'_2},t) = d(y,v_{i'_2}) + d(v_{i'_2},v_i) + d(v_i,t) = d(y,v_i) + d(v_i,t)$, as required.

To find the index i'_2 we can use a binary search on the range $0 \leq i \leq d(s,t)$ in $O(\log n)$ time. Observe that the distances $d(y,t), d(y,v_i), d(v_i,t)$ can be obtained in $O(1)$ time: the distances $d(y,t), d(y,v_i)$ were computed and stored during the BFS computation from the source vertex $y \in B$ (during the computation of T_y). Since $v_i \in P(s,t)$, the distance $d(v_i,t)$ can be computed as $d(v_i,t) = d(s,t) - d(s,v_i)$ and the distances $d(s,t), d(s,v_i)$ were computed and stored during the BFS computation from the source vertex s (during the computation of T_s). During the binary search, if $d(y,t) < d(y,v_i) + d(v_i,t)$ then $i < i'_2$ and if $d(y,t) = d(y,v_i) + d(v_i,t)$ then $i'_2 \leq i$.

In the following Lemma we prove that $0 \leq i'_2 \leq i_2 \leq d(s,t)$.

Lemma 8.2. *It holds that $0 \leq i'_2 \leq i_2 \leq d(s,t)$.*

Proof. Recall the definition of i'_2, i_2: (a) $0 \leq i'_2 \leq d(s,t)$ is the minimum index such that $d(y,t) = d(y,v_{i'_2}) + d(v_{i'_2},t)$. (b) $1 \leq i_2 \leq d(s,t)$ is the maximum index such that (v_{i-1}, v_i) is (y,t)-replaceable for every $i \leq i_2$.

By definition, $0 \leq i'_2$ and $i_2 \leq d(s,t)$, so we only need to prove that $i'_2 \leq i_2$.

Let $P(y, v_{i'_2})$ be an arbitrary shortest path from y to $v_{i'_2}$. Let $P(v_{i'_2}, t) = (v_{i'_2+1}, \ldots, v_{d(s,t)})$ be the subpath of the shortest path $P(s,t)$ from $v_{i'_2}$ to t. Since $P(v_{i'_2},t)$ is a subpath of a shortest path then $P(v_{i'_2},t)$ is also a shortest path from $v_{i'_2}$ to t. Since $d(y,t) = d(y,v_{i'_2}) + d(v_{i'_2},t)$ it follows that the path $P' = P(y,v_{i'_2}) \circ P(v_{i'_2},t)$ is a shortest path from y to t (as $|P'| = |P(y,v_{i'_2}) \circ P(v_{i'_2},t)| = |P(y,v_{i'_2})| + |P(v_{i'_2},t)| = d(y,v_{i'_2}) + d(v_{i'_2},t) = d(y,t)$).

Furthermore, it cannot be that $P(y,v_{i'_2})$ contains a vertex v_i for any $i < i'_2$, as otherwise $d(y,t) = d(y,v_i) + d(v_i,t)$ for $i < i'_2$, which is a contradiction to the definition of i'_2 as the minimum index such that $d(y,t) = d(y,v_{i'_2}) + d(v_{i'_2},t)$. Hence, P' is a shortest path from y to t which does not contain any of the vertices $v_0, \ldots, v_{i'_2-1}$. Therefore, P' is a shortest path from y to t which does not contain any of the edges (v_i, v_{i+1}) for every $0 \leq i < i'_2$. It follows that all the edges (v_i, v_{i+1}) for every $0 \leq i < i'_2$ are (y,t)-replaceable.

The index $1 \leq i_2 \leq d(s,t)$ is the maximum index such that (v_{i-1}, v_i) is (y,t)-replaceable for every $i \leq i_2$. Since the edges (v_i, v_{i+1}) are (y,t)-replaceable for every $0 \leq i < i'_2$, it follows that $i_2 \geq i'_2$.

8.1 Computing The Index i_2 Given i'_2 We first give an overview of how to compute the index i_2 given i'_2

(see Figure 6). Given i'_2 let z be the vertex on the path from y to t in T_y which is on the same level as the vertex $u = v_{i'_2}$ in the tree T_y. Note that in case $v_{i'_2}$ is an ancestor of t in T_y then $z = v_{i'_2}$. Even though, $v_{i'_2}$ is on a shortest path from y to t, it might be that $v_{i'_2}$ is not an ancestor of t in T_y as there might be different shortest paths from y to t (some of which do not contain $v_{i'_2}$) and in this case we have $z \neq v_{i'_2}$.

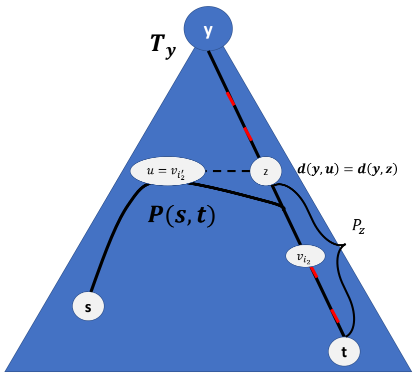

Figure 6: An illustration of how to find the index i_2 given the index i'_2. Given the index i'_2 we can find the vertex $u = v_{i'_2} \in P(s,t)$. Let z be the vertex on the path from y to t in T_y such that $d(y,z) = d(y,u)$. Using the data-structure from Lemma 8.6 we find the first edge (v_{i_2}, v_{i_2+1}) on the path from z to t which is not (y,t)-replaceable, and this edge is associated with the index i_2. In this Figure, red edges represent edges that are not (y,t)-replaceable. In the figure we draw for simplicity that from the first vertex that $P(s,t)$ enters the path P_z both paths share a common suffix. We remark that generally this may not be the case, only the red edges must be common to both $P(s,t)$ and $P(z)$ but the rest of the edges may be disjoint.

Next, given z and using a data-structure based on dominator trees described in Lemma 8.6 we find the first edge on the path from z to t in T_y which is not (y,t)-replaceable. In Lemma 8.3 we prove that this edge is (v_{i_2}, v_{i_2+1}) and thus we can find the index i_2 (we use a perfect hash table that maps the edges $(v_i, v_{i+1}) \in P(s,t)$ to the index i for every $0 \leq i < d(s,t)$).

In the rest of this Section we describe in detail how to find the index i_2 given the index i'_2. Let $u = v_{i'_2} \in P(s,t)$, where $P(s,t)$ is the shortest path from s to t in T_s. According to Lemma 8.1, $d(y,t) = d(y,u) + d(u,t)$. Let z be the vertex on the path from y to t in T_y such that $d(y,z) = d(y,u)$. Then we also have $d(y,t) = d(y,z) + d(z,t)$ and thus $d(u,t) = d(z,t)$.

Let $P_z = (u_0, u_1, \ldots, u_k)$ be the path from z to t in T_y (with $u_0 = z, u_k = t$). We claim that either all the edges of P_z are (y,t)-replaceable, or the edge $(v_{i_2}, v_{i_2+1}) \in P_z$ is the first edge along P_z which is not (y,t)-replaceable.

Lemma 8.3. *It holds that either all the edges of P_z are (y,t)-replaceable, or the edge $(v_{i_2}, v_{i_2+1}) \in P_z$ is the first edge along P_z which is not (y,t)-replaceable (see Figure 7).*

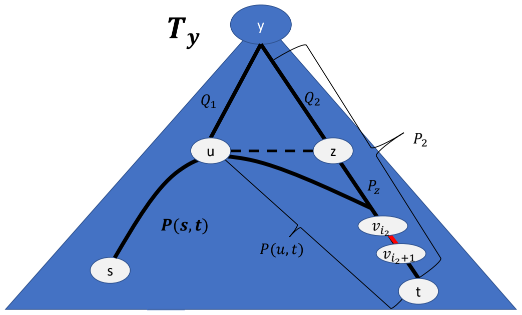

Figure 7: Notations for the proof of Lemma 8.3. In this Figure we see the tree T_y. Here P_2 is the path from y to t in T_y, and $P(s,t) = (v_0, \ldots, v_{d(s,t)})$ is the shortest path from s to t in T_s. The vertex $u = v_{i'_2} \in P(s,t)$ is defined as in Lemma 8.1 where $d(y,t) = d(y,u) + d(u,t)$ and thus the $P_1 = Q_1 \circ P(u,t)$ (which is the concatenation of a shortest path from y to u and a shortest path from u to t) is a shortest path from y to t. Here, Q_1 is an arbitrary shortest path from y to u and $P(u,t)$ is the subpath of $P(s,t)$ from u to t (since $P(s,t)$ is a shortest path then its subpath $P(u,t)$ is also a shortest path). The path P_z is a shortest path from z to t in T_y. We prove in Lemma 8.3 that the first edge of $P(u,t)$ which is not (y,t)-replaceable (which is (v_{i_2}, v_{i_2+1}) by definition of i_2) is also the first edge of P_z which is not (y,t)-replaceable. Hence, in order to find the index i_2 we can search for the first edge of P_z which is not (y,t)-replaceable.

Proof. Let Q_1 be any shortest path from y to u, and let $P(u,t) \subset P(s,t)$ be the shortest path from u to t in $P(s,t)$. Let $P_1 = Q_1 \circ P(u,t)$. Since $d(y,t) = d(y,u) + d(u,t)$ and $|P_1| = |Q_1| + |P(u,t)| = d(y,u) + d(u,t) = d(y,t)$, then P_1 is a shortest path from y to t which goes through u. Let P_2 be the shortest path from y to t in T_y then $z \in P_2$. Let Q_2 be the subpath of P_2 from y to z, then $P_2 = Q_2 \circ P_z$.

Let $e_1 = (a,b)$ be the first edge of P_z which is not (y,t)-replaceable such that $d(y,a) < d(y,b)$. Then e_1 appears on all the shortest paths from y to t. In particular e_1 appears on the shortest path P_1 from y to t. Since $e_1 = (a,b) \in P_z$ it follows that $d(y,z) \leq d(y,a)$. Since $d(y,u) = d(y,z)$ it follows that $d(y,u) \leq d(y,a)$ and hence $e_1 \in P_1$ appears on $P_1 = Q_1 \cdot P(u,t)$ after the vertex u, i.e., $e_1 \in P(u,t)$.

Let $e_2 = (c,d)$ be the first edge of $P(u,t)$ which is not (y,t)-replaceable such that $d(y,c) < d(y,d)$ A symmetric claim shows that $e_2 \in P_z$; the edge e_2 appears on all the shortest paths from y to t. In particular e_2 appears on the shortest path P_2 from y to t. Since $e_1 = (c,d) \in P(u,t)$ it follows that $d(y,u) \leq d(y,c)$. Since $d(y,u) = d(y,z)$ it follows that $d(y,z) \leq d(y,c)$ and hence $e_2 \in P_2$ appears on $P_2 = Q_2 \cdot P_z$ after the vertex z, i.e., $e_2 \in P_z$.

Since $e_2 \in P(u,t) \subseteq P(s,t) = (v_0, \dots, v_{d(s,t)})$ then there exists an index $0 \leq i < d(s,t)$ such that $e_2 = (v_i, v_{i+1})$. Since e_1 is the first edge of P_z which is not (y,t)-replaceable and $e_1 \in P(u,t)$, and e_2 is the first edge of $P(u,t)$ which is not (y,t)-replaceable and $e_2 \in P_z$, it follows that $e_1 = e_2 = (v_i, v_{i+1})$.

Finally, we claim that $i = i_2$. Indeed, according to Lemma 8.1 for every $0 \leq i' < i'_2$ it holds that $d(y,t) < d(y, v_{i'}) + d(v_{i'}, t)$ and thus no shortest path from y to t may pass through $v_{i'}$. Hence, $(v_{i'}, v_{i'+1})$ is not (y,t)-replaceable for every $0 \leq i' < i'_2$. In particular, P_2 is a shortest path from y to t which does not contain any of the edges $\{(v_{i'}, v_{i'+1}) | 0 \leq i' < i'_2\}$, and thus all the edges $\{(v_{i'}, v_{i'+1}) | 0 \leq i' < i'_2\}$ are (y,t)-replaceable. Furthermore, since (v_i, v_{i+1}) is the first edge of $P(u,t)$ which is not (y,t)-replaceable and recall $u = v_{i'_2}$, it follows that (v_i, v_{i+1}) is the first edge of $P(s,t)$ which is not (y,t)-replaceable. According to the definition of the index i_2, it follows that $i = i_2$.

It follows from Lemma 8.3 that we can find i_2 by searching for the first edge $e' = (v_{i_2}, v_{i_2+1})$ along the path from z to t in T_y such that e' is (y,t)-replaceable. Following we describe a data-structure based on dominator trees to find the edge e' in $O(\log n)$ time.

8.2 The Graph G_y And Dominator Tree D_y

We describe dominator trees as defined by [21]. Given a directed graph G and a source vertex s, we assume all the vertices of G are reachable from s. We say that a vertex v dominates a vertex w if every path from s to w in G passes through v. Note that every vertex v has at least one dominating vertex, as s dominates all the vertices. We define the immediate dominator $v = \text{idom}(w)$ of vertex w as the vertex $v \neq w$ which dominates w such that every vertex $u \neq w$ that dominates w also dominates v. It was proven in [21] that every vertex $w \in G \setminus \{s\}$ has a unique immediate dominator $\text{idom}(w)$, and the dominator tree $\text{DomTree}_G(s)$ which contains all the edges $(\text{idom}(w), w)$ for every $w \in V \setminus \{s\}$ forms a valid tree where s is the root of the tree. They also prove that one can compute $\text{DomTree}_G(s)$ in $\widetilde{O}(m+n)$ time. Note that, for a vertex $w \in V$ the set of nodes that are on any shortest path from s to w are exactly the set of ancestors of w in $\text{DomTree}_G(s)$.

The Graph G_y:

Let $y \in B$. We construct a DAG G_y by keeping only edges of E which are from some level i of the tree T_y to level $i+1$ in T_y, and we direct these edges away from the root y. Furthermore, we add the following auxiliary vertices $p(e)$ to G_y. For every edge $e = (u,v) \in E(T_y)$ we add an auxiliary vertex $p(e)$ in G_y and replace the edge (u,v) with the edges $(u, p(e)), (p(e), v)$.

Lemma 8.4. *Let $e \in P_z$ be an edge on the path from z to t in T_y, then $p(e)$ dominates t in G_y iff e is not (y,t)-replaceable.*

Proof. As in Theorem 2.1, every shortest path from y to t in G appears in G_y (when directed the edges from y to t), and vice versa, every path from y to t in G_y is a shortest path from y to t in G (when removing the direction of the edges). It is not hard to verify now that $p(e)$ is a dominating vertex in G_y iff $p(e)$ appears on all the paths from y to t in the graph G_y, iff e appears on all the shortest paths from y to t in G, iff e is not (y,t)-replaceable.

The Dominator Tree D_y:

We build a dominator tree $D_y = \text{DomTree}_{G_y}(y)$ rooted at y for the DAG G_y such that given a vertex $t \in V$ the dominator tree D_y can list all the vertices on the path from y to t that dominate t. We augment the vertices $v \in D_y$ with a pointer $\text{first}(v)$ as follows. We run a pre-order scan on the tree D_y and for every vertex $v \in D_y$ we store $\text{first}(v)$, a pointer to the lowest ancestor of v which is an auxiliary vertex of the form $p(e)$ that represents an edge e of G, if no such ancestor of v exists then we set $\text{first}(v) = \text{null}$.

Lemma 8.5. *Let $y \in B, t \in V, 0 \leq i \leq d(y,t)$. Using D_y, one can find the first auxiliary vertex $p(e)$ on the path from y to t in D_y (here by "first" we mean closest to y) whose level in G_y is at least i in $O(\log n)$ time. If there is no auxiliary vertex $p(e)$ on the path from y to t in D_y whose level in G_y is at least i, we report that this is the case.*

Proof. Let $L \leftarrow y, R \leftarrow \text{idom}(t)$. We run a binary search on the path from L to R in D_y. Let v be a vertex on the middle of the path from L to R in D_y. Let $[L,R]$ represent the path from L to R in D_y including both L and R, and let $(L,R]$ represent the path from L to R in D_y including R and excluding L. If $\text{level}(\text{first}(v)) < i$ then we continue the search on the subpath $(v,R]$ and otherwise we continue the search in $[L,v]$. Assume the binary search finished with the vertex v. Then, if $\text{level}(\text{first}(v)) \geq i$ we return $p(e) = \text{first}(v)$, and otherwise we report that there is no auxilary vertex $p(e)$ on the path from y to t in D_y whose level in G_y is at least i. The run time of this procedure is $O(\log n)$ as we do a binary search on the path from y to t in D_y which is of length at most n.

Lemma 8.6. *Let $y \in B$. One can construct a data-structure D_y which supports the following operations.*

*(1) $D_y \leftarrow$ **Construct(T_y)** : given a BFS tree rooted at y denoted by T_y, construct the data-structure D_y in $\widetilde{O}(m)$ time and store it using $\widetilde{O}(n)$ space. (2) **Query(t,i)** : the query procedure is given $t \in V$ and an integer $0 \leq i \leq d(y,t)$. Let z be the i^{th} vertex along the path from y to t in T_y. Let P_z be the shortest path from z to t in T_y. The answer to the query (t,i) is the first edge e' on the path P_z which is not (y,t)-replaceable, or reporting that all the edges of P_z are (y,t)-replaceable.*

Proof. During Construct(T_y) we build the dominator tree D_y as described above, including the pointers $\text{first}(v)$ for every $v \in D_y$.

Upon Query(t,i) we run a binary search as in Lemma 8.5 to find the first auxiliary vertex $p(e)$ on the path from y to t in D_y whose level in G_y is at least i. According to Lemma 8.4, e' is the first edge on the path P_z which is not

(y,t)-replaceable. If there is no auxilary vertex $p(e')$ on the path from y to t in D_y whose level in G_y is at least i, we report that all the edges of P_z are (y,t)-replaceable, which is also correct according to Lemma 8.4.

Corollary 8.1. *Let $y \in B, t \in V$. Using D_y and i'_2, we can find the index i_2 in $O(\log n)$ time.*

Proof. As above, let $u = v_{i'_2} \in P(s,t)$, since we have already computed the index i'_2 in Lemma 8.1 in $O(\log n)$ time, we can retrieve $u = v_{i'_2}$ from the path $P(s,t) = (v_0, \dots, v_{d(s,t)})$ in constant time. So we can assume the algorithm knows u.

Let z be the vertex on the path from y to t in T_y such that $d(y,z) = d(y,u)$, and let $i = d(y,u)$. Thus, we can query the data-structure D_y from Lemma 8.6 with the query (t,i) in $O(\log n)$ time.

If D_y reports that all the edges of P_z are (y,t)-replaceable then it follows that $i_2 = d(s,t)$ and $v_{i_2} = t$. Otherwise, D_y answers the query (t,i) by returning the first edge e' in P_z which is not (y,t)-replaceable. According to Lemma 8.3 it follows that the edge $e' = (v_{i_2}, v_{i_2+1}) \in P_z$ is the first edge along P_z which is not (y,t)-replaceable. During preprocessing, we associate using a perfect hash table every edge (v_i, v_{i+1}) of $P(s,t)$, for every $0 \le i < d(s,t)$, with the index i. Then, given the edge $e' = (v_{i_2}, v_{i_2+1})$ we can retrieve from the hash table the index i_2, and return it. Query time is dominated by the time it takes to query D_y with (t,i), which is $O(\log n)$ time.

We outline how to compute i'_1 and i_1 in a similar way of computing i'_2 and i_2. We leave out the details, as the computation of i'_1, i_1 and proofs are similar to the computation of i'_2, i_2.

Lemma 8.7. *Given $x \in B$ one can compute the index i_1 in $O(\log n)$ time.*

Proof. As in Lemma 8.1, let $0 \le i'_1 \le d(s,t)$ be the maximum index such that $d(s,x) = d(s, v_{i'_1}) + d(v_{i'_1}, x)$. According to the definition of i'_1, it follows that for every $i'_1 < i \le d(s,t)$ it holds that $d(s,v_i) + d(v_i,x) > d(s,x)$, and by a similar proof as in Lemma 8.1 it follows that for every $0 \le i \le i'_1$ it holds that $d(s,x) = d(s,v_i) + d(v_i,x)$. Thus, we can find the index i'_1 using a binary search on the vertices of $P(s,t)$ in $O(\log n)$ time.

Let $u = v_{i'_1} \in P(s,t)$ and recall that $P(s,t)$ is the shortest path from s to t in T_s. According to the claim above, $d(s,x) = d(s,u) + d(u,x)$. Let z be the vertex on the path from x to s in T_x such that $d(x,z) = d(x,u)$. Let $i = d(x,z) = d(x,u)$ be the depth of the vertex z in T_x. The distance $d(x,u)$ was computed during the BFS with source vertex x (when computing the tree T_x), and thus it is stored and known to the algorithm, and hence we can compute i the depth of z in T_x in constant time.

Denote by $P_z = (u_0, u_1, \dots, u_k)$ the path from z to s in T_x (with $u_0 = z, u_k = s$). Similar to the proof of 8.3, we claim that either all the edges of P_z are (s,x)-replaceable, or the edge $(v_{i_1}, v_{i_1+1}) \in P_z$ is the first edge along P_z which is not (s,x)-replaceable.

Let D_x be the data-structure as in Lemma 8.6. It follows that (v_{i_1}, v_{i_1+1}) can be found by Query(s,i) to the data-structure D_x, as in Lemmas 8.1 and 8.6 and thus we can obtain i_1 in $O(\log n)$ time.

We are now ready to prove Lemma 7.5.

Proof. [Proof of Lemma 7.5] For every $y \in B$ we compute the tree T_y using BFS in time $O(m+n)$, and we build the data-structure D_y as in Lemma 8.6 in $\widetilde{O}(m)$ time. Since the of vertices in B is $\widetilde{O}(\sqrt{n})$ (w.h.p.) we get that the time to compute T_y, D_y for all $y \in B$ is $\widetilde{O}((m+n)\sqrt{n})$ (w.h.p.).

Then, for every $x, y \in B$ and $t \in V$ we find the indices i'_1, i'_2 in $O(\log n)$ time using a binary search on the vertices of the path $P(s,t)$ as in Lemmas 8.1 and 8.7. Then we find the indices i_1, i_2 using D_x and D_y as in Lemmas 8.1 and 8.7 in $O(\log n)$ time. Since there are $\widetilde{O}(n^2)$ triples $x, y \in B, t \in V$ (w.h.p.), given T_x, D_x, T_y, D_y the total time to find all indices i_1, i_2 for all the triples x, y, t is $\widetilde{O}(n^2)$. So, total processing time is $\widetilde{O}(m\sqrt{n} + n^2)$ (w.h.p.).

Finding all the indices i_1, i_2 for every $x, y \in B, t \in V$ gives us all the intervals $I_{x,y,t}$ as $I_{x,y,t} = \{(v_i, v_{i+1}) | i_1 \le i < i_2\}$.

9 Case Analysis: Proof Of Correctness

In this Section we prove Theorem 3.1, which proves the correctness of our algorithm. Intuitively, we prove that one of the cases must hold: the replaceable edge case ($d_{s,t,e} = h_0[t,e]$), the small fall case ($d_{s,t,e} = h_1[t,e]$), the single pivot case ($d_{s,t,e} = h_2[t,e]$), or the double pivot case ($d_{s,t,e} = h_3[t,e]$). Our goal is to prove Theorem 3.1, that for every $t \in V, e \in P(s,t)$ it holds that $\hat{d}_{s,t,e} = d_{s,t,e}$.

Recall that $\hat{d}_{s,t,e} = \min\{h_0[t,e], h_1[t,e], h_2[t,e], h_3[t,e]\}$. We first prove that $\hat{d}_{s,t,e} \ge d_{s,t,e}$.

Lemma 9.1. *For every $t \in V, e \in P(s,t)$ it holds that $\hat{d}_{s,t,e} \ge d_{s,t,e}$.*

Proof. It follows from Lemma 4.1 that $h_0[t,e] \ge d_{s,t,e}$. It follows from Lemma 5.3 that $h_1[t,e] \ge d_{s,t,e}$. It follows from Lemma 6.2 that $h_2[t,e] \ge d_{s,t,e}$. It follows from Lemma 7.1 that $h_3[t,e] \ge d_{s,t,e}$. Since $\hat{d}_{s,t,e} = \min\{h_0[t,e], h_1[t,e], h_2[t,e], h_3[t,e]\}$ it follows that $\hat{d}_{s,t,e} \ge d_{s,t,e}$.

According to Lemma 9.1, in order to prove $\hat{d}_{s,t,e} = d_{s,t,e}$ we only have to prove that either one of the cases hold: $h_0[t,e] = d_{s,t,e}$ or $h_1[t,e] = d_{s,t,e}$ or $h_2[t,e] = d_{s,t,e}$ or $h_3[t,e] = d_{s,t,e}$.

To prove Theorem 3.1, it is sufficient to assume that $h_0[t,e], h_1[t,e], h_2[t,e] > d_{s,t,e}$ and prove that $h_3[t,e] = d_{s,t,e}$ in this case.

Definition 9.1. *Let $t \in V, e \in E(T_s)$. We say that (t,e) is of type "strict double pivot" iff $h_0[t,e], h_1[t,e], h_2[t,e] > d_{s,t,e}$.*

In this Section we prove Theorem 3.1 by proving the following equivalent Lemma.

Lemma 9.2. *Let $t \in V, e \in E(T_s)$ such that (t, e) is of type "strict double pivot". Then $h_3[t, e] = d_{s,t,e}$.*

Let $P_{s,t,e}$ be a replacement path. Let $x_1, \ldots, x_k \in B$ be all the sampled vertices on the path $P_{s,t,e}$ ordered according to their appearance along the path $P_{s,t,e}$ from s to t, and let $x_0 = s, x_{k+1} = t$.

Lemma 9.3. *If (t, e) is "strict double pivot" then with high probability* $\text{Detour}_{s,t,e}$ *contains at least 2 vertices of B.*

Proof. Since we assume $h_0[t, e], h_1[t, e], h_2[t, e] > d_{s,t,e}$ it follows from Lemma 6.3 that $\text{Detour}_{s,t,e}$ contains at least $\sqrt{n}$ edges. Therefore, according to Lemma 2.1, with high probability, $\text{Detour}_{s,t,e}$ contains at least 2 sampled vertices of B.

Lemma 9.4. *Let $0 \le i \le k$ be an index and $e = (u, v)$ such that u is closer to s than v (i.e., $d(s, u) < d(s, v)$). Then with high probability either:*
1. e is (x_i, x_{i+1})-replaceable or
2. $d(v, x_{i+1}) < \sqrt{n}$ and either $h_1[x_{i+1}, e] = d_{s,x_{i+1},e}$ or $h_0[x_{i+1}, e] = d_{s,x_{i+1},e}$.

Proof. If e is (x_i, x_{i+1})-replaceable then we are done. So we assume that e is not (x_i, x_{i+1})-replaceable, *i.e.*, $d_{x_i,x_{i+1},e} > d(x_i, x_{i+1})$. We prove that either $h_1[x_{i+1}, e] = d_{s,x_{i+1},e}$ or $h_0[x_{i+1}, e] = d_{s,x_{i+1},e}$. Let $P = P(x_i, x_{i+1})$ be a shortest path from x_i to x_{i+1} and let $P' = P_{x_i,x_{i+1},e}$ be a shortest path from x_i to x_{i+1} avoiding e, therefore $|P| < |P'|$. Note that $e \in P$ since e is not (x_i, x_{i+1})-replaceable (and therefore e appears on all shortest paths from x_i to x_{i+1} and in particular on P). Since P, P' are both paths from x_i to x_{i+1} such that $e \in P$ and $e \notin P'$ it follows that $P \cup P'$ contains a cycle C such that $e \in C$, and $|C| \le |P| + |P'|$.

Since x_i and x_{i+1} are two consecutive sampled vertices of B along $P_{s,t,e}$ it follows from Lemma 2.1 that $|P'|$ (which is the length of the subpath of $P_{s,t,e}$ from x_i to x_{i+1}) is less than $\sqrt{n}$ with high probability. Hence, $|P| < |P'| < \sqrt{n}$ with high probability, and $|C| \le |P| + |P'| < 2\sqrt{n}$ with high probability.

Since $e \in P$ then $v \in P$ and hence $d(v, x_{i+1}) \le |P| = d(x_i, x_{i+1})$. As $d(x_i, x_{i+1}) < \sqrt{n}$ w.h.p, we also get that $d(v, x_{i+1}) < \sqrt{n}$ w.h.p.

We get that the conditions of Lemma 5.4 hold (when substituting $x = x_{i+1}$ in Lemma 5.4):

(a) e is contained in a cycle C of length at most $2\sqrt{n}$.
(b) $d(v, x_{i+1}) < \sqrt{n} < 2\sqrt{n}$.

Hence, by Lemma 5.4, either $h_1[x_{i+1}, e] = d_{s,x_{i+1},e}$ or $h_0[x_{i+1}, e] = d_{s,x_{i+1},e}$ with high probability.

Lemma 9.5. *Assume (t, e) is of "strict double pivot" type and let $0 \le i \le k$ be an index. Then with high probability either e is (x_i, x_{i+1})-replaceable or $h_3[t, e] = d_{s,t,e}$.*

Proof. If e is (x_i, x_{i+1})-replaceable then we are done. So we assume that e is not (x_i, x_{i+1})-replaceable and prove that $h_3[t, e] = d_{s,t,e}$ w.h.p.

If $x_{i+1} = t$ then by Lemma 9.4, with high probability either $h_0[t, e] = d_{s,t,e}$ or $h_1[t, e] = d_{s,t,e}$ which is a contradiction to the assumption that (t, e) is of "strict double pivot" type. Therefore, for the rest of the proof assume $x_{i+1} \ne t$. Since either $x_{i+1} \in B$ or $x_{i+1} = t$, it follows that $x_{i+1} \in B$.

Since e is not (x_i, x_{i+1})-replaceable we claim that it must hold that e is (x_{i+1}, t)-replaceable. Let P' be the subpath of $P_{s,t,e}$ from x_i to t. Then P' is a shortest from x_i to t avoiding e and therefore P' is also a replacement path for (x_i, t, e). Furthermore, the vertex x_{i+1} appears after x_i along $P_{s,t,e}$ and therefore $x_{i+1} \in P'$ is a vertex on the replacement path of (x_i, t, e). By Lemma 4.3 (substitute $x = x_i, y = t, p = x_{i+1}$ in Lemma 4.3) we either have e is (x_i, x_{i+1})-replaceable or e is (x_{i+1}, t)-replaceable. But since we assume that e is not (x_i, x_{i+1})-replaceable then we get that e is (x_{i+1}, t)-replaceable.

Let $P = P(s, x_{i+1})$ be the shortest path from s to x_{i+1} in T_s and consider two cases $e \in P$ and $e \notin P$. If $e \notin P$ then e is both (s, x_{i+1})-replaceable and (x_{i+1}, t)-replaceable. In that case, according to Definition 7.1 we get that (t, e) is of type "double pivot" (substitute $x = y = x_{i+1}$ in the definition). According to Lemma 7.1 it holds that $h_3[t, e] = d_{s,t,e}$ and we are done.
We assume by contradiction that $e \in P$. Recall the definition of h_2 is as follows. $h_2[t, e] = \min_{x \in B}\{\min\{h_0[x, e], h_1[x, e]\} + d(x, t)$ where e is among the $2\sqrt{n}$ last edges of the path from s to x in $T_s\}$. We prove that if $e \in P$ then $h_2[t, e] = d_{s,t,e}$ which is a contradiction to the Lemma's assumptions. Let $x = x_{i+1}$. We prove that: (1) e is among the $2\sqrt{n}$ last edges of the path from s to x in T_s, and (2) $d_{s,t,e} = \min\{h_0[x, e], h_1[x, e]\} + d(x, t)$.

(1) Let $e = (u, v)$ such that u is closer to s than v (*i.e.*, $d(s, u) < d(s, v)$). Since we assumed e is not (x_i, x_{i+1})-replaceable and $x = x_{i+1}$ then we have already proved in Lemma 9.4 that $d(v, x) = d(v, x_{i+1}) < \sqrt{n}$ with high probability.

Since $e = (u, v) \in P$ and $d(v, x) < \sqrt{n}$ then e is among the $\sqrt{n}$ last edges of the path from s to x in T_s.

(2) We have already proved in Lemma 9.4 that $h_0[x, t] = d_{s,x,e}$ or $h_1[x, t] = d_{s,x,e}$. Since $h_0[x, e] \ge d_{s,x,e}$ and $h_1[x, e] \ge d_{s,x,e}$ it follows that $d_{s,x,e} = \min\{h_0[x, e], h_1[x, e]\}$. Since e is (x, t)-replaceable it follows that $d_{x,t,e} = d(x, t)$. Since $x \in P_{s,t,e}$ then $d_{s,t,e} = d_{s,x,e} + d_{x,t,e} = \min\{h_0[x, e], h_1[x, t]\} + d(x, t)$.

We get that $h_2[t, e] = d_{s,t,e}$ which is a contradiction to our assumptions.

We are now ready to prove Theorem 3.1.

Proof of Theorem 3.1:

We assume $h_0[t, e], h_1[t, e], h_2[t, e] > d_{s,t,e}$ and prove that $h_3[t, e] = d_{s,t,e}$ with high probability.

If there exists an index $1 \le i \le k$ such that e is not (x_i, x_{i+1})-replaceable then according to Lemma 9.4 it holds with high probability that $h_3[t, e] = d_{s,t,e}$. Assume for the rest of the proof that for every $1 \le i \le k$ it holds that e is (x_i, x_{i+1})-replaceable.

Let $1 \le \ell \le k$ be the minimum index such that $x_\ell \in \text{Detour}_{s,t,e}$ and let r be the maximum index such that

$\ell < r \leq k$ and $x_r \in \text{Detour}_{s,t,e}$. According to Lemma 9.3, with high probability $\text{Detour}_{s,t,e}$ contains at least two vertices of B and therefore with high probability the indices $\ell < r$ exist.

Since e is $(x_{\ell-1}, x_\ell)$-replaceable, there exists a shortest path P_1 from $x_{\ell-1}$ to x_ℓ which avoids e (*i.e.*, $|P_1| = d(x_{\ell-1}, x_\ell)$). Since $1 \leq \ell$ is the minimum index such that $x_\ell \in \text{Detour}_{s,t,e}$ it follows that $x_{\ell-1}$ is on $\text{CommonPref}_{s,t,e}$ (recall that $x_0 = s$, so $x_{\ell-1}$ is well defined). Therefore $P(s, x_{\ell-1})$, the subpath of $P(s,t)$ from s to $x_{\ell-1}$, is a shortest path from s to $x_{\ell-1}$. We get that $P_1' = P(s, x_{\ell-1}) \circ P_1$ is a path from s to x_ℓ in G and it also avoids e.

We claim that e is (s, x_ℓ)-replaceable. If e is not (s, x_ℓ)-replaceable then e appears on every shortest path from s to x_ℓ. Since $x_{\ell-1} \in \text{CommonPref}_{s,t,e}$ appears before e on the shortest path $P(s,t)$ then we may alter every shortest path from s to x_ℓ to contain $x_{\ell-1}$ be replacing its prefix from s to e with the prefix of $P(s,t)$ from s to e. Therefore, since there exists a shortest path from s to x_ℓ that contains $x_{\ell-1}$ it follows that $d(s, x_\ell) = d(s, x_{\ell-1}) + d(x_{\ell-1}, x_\ell)$. We claim that $P_1' = P(s, x_{\ell-1}) \circ P_1$ is a shortest path from s to x_ℓ in G (and we have already proved that P_1' does not contain e). It holds that $d(s, x_\ell) = d(s, x_{\ell-1}) + d(x_{\ell-1}, x_\ell) = |P(s, x_{\ell-1})| + |P_1| = |P(s, x_{\ell-1}) \circ P_1|$, and hence $P(s, x_{\ell-1}) \circ P_1$ is a shortest path from s to x_ℓ in G that avoids e. Hence, e is (s, x_ℓ)-replaceable.

Similarly, since e is (x_r, x_{r+1})-replaceable, there exists a shortest path P_2 from x_r to x_{r+1} which avoids e (*i.e.*, $|P_2| = d(x_r, x_{r+1})$). Since $r \leq k$ is the maximum index such that $x_r \in \text{Detour}_{s,t,e}$ it follows that x_{r+1} is on $\text{CommonSuff}_{s,t,e}$ (recall that $x_{k+1} = t$, so x_{r+1} is well defined). Therefore $P(x_{r+1,t})$ is a shortest path from x_{r+1} to t. We get that $P_2' = P_2 \circ P(x_{r+1}, t)$ is a path from x_r to t in G which also avoids e.

We claim that e is (x_r, t)-replaceable. If e is not (x_r, t)-replaceable then e appears on every shortest path from x_r to t. Since $x_{r+1} \in \text{CommonSuff}_{s,t,e}$ appears after e on the shortest path $P(s,t)$ then we may alter every shortest path from x_r to t to contain x_{r+1} be replacing its suffix from e to t with the suffix of $P(s,t)$ from e to t. Therefore, since there exists a shortest path from x_r to t that contains x_{r+1} it follows that $d(x_r, t) = d(x_r, x_{r+1}) + d(x_{r+1}, x_t)$. We claim that $P_2' = P_2 \circ P(x_{r+1}, t)$ is a shortest path from x_r to t in G (and we have already proved that P_2' does not contain e). It holds that $d(x_r, t) = d(x_r, x_{r+1}) + d(x_{r+1}, t) = |P_2| + |P(x_{r+1}, t)| = |P_2 \circ P(x_{r+1}, t)|$, and hence $P(x_r, x_{r+1}) \circ P_2$ is a shortest path from x_r to t in G that avoids e. Hence, e is (x_r, t)-replaceable.

Let i be the minimum index such that $\ell < i \leq r$ and e is (x_i, t)-replaceable. It follows that either $i = \ell + 1$ or e is not (x_{i-1}, t)-replaceable. We prove that in either case e is (s, x_{i-1})-replaceable. If $i = \ell + 1$, then we have already proved that e is (s, x_ℓ)-replaceable. If $i > \ell + 1$ then by definition of $\ell < i \leq r$ as the minimum index such that e is (x_i, t)-replaceable, it follows that e is not (x_{i-1}, t)-replaceable. According to Corollary 4.2 it must hold that e is (s, x_{i-1})-replaceable.

Let $x = x_{i-1}, y = x_i$. It follows that with high probability $P_{s,t,e}$ is a replacement path such that $x, y \in B$ and x appears before y on the detour part of $P_{s,t,e}$, and e is (s, x)-replaceable, (x, y)-replaceable and (y, t)-replaceable. Therefore, by Definition 7.1 it follows that (t, e) is of type "double pivot", and according to Lemma 7.1 it holds that $h_3[t, e] = d_{s,t,e}$ with high probability. ∎

10 Conditional Lower Bounds

In this section we prove a combinatorial conditional lower bound of $\Omega(m\sqrt{n})$ for the SSRP problem in undirected unweighted graphs by showing a combinatorial reduction from Boolean Matrix Multiplication (BMM) to SSRP. We prove that BMM with $n \times n$ matrices containing a total number of m 1's can be solved using $\sqrt{n}$ independent calls to SSRP.

A combinatorial algorithm is often referred to as an algorithm that does use any matrix multiplication tricks. The interest in combinatorial algorithms stems from the assumption that in practice combinatorial algorithms are more efficient as the constants hide in the matrix multiplication bounds are considered high.

Denote by $BMM(n, m)$ the combinatorial BMM problem for multiplying $n \times n$ matrices with a total number of m 1's where only combinatorial algorithm are allowed. Our conditional lower bound relies on the following Conjecture for combinatorial BMM, that there is no truly subcubic algorithm for combinatorial BMM. The combinatorial BMM conjecture was used to prove conditional lower bounds for the replacement paths problems, for example in [16], [31], and [26].

CONJECTURE 10.1. *In the Word RAM model with words of $O(\log n)$ bits, any combinatorial algorithm for multiplying two Boolean $n \times n$ matrices with a total number of m 1's (i.e., any combinatorial algorithm for the $BMM(n, m)$ problem) requires $(mn)^{1-o(1)}$ time in expectation to compute the Boolean product of the two $n \times n$ matrices.*

We denote by $A_{SSRP}(n, m)$ an algorithm which solves SSRP in undirected unweighted graphs with n vertices and m edges. In this Section We prove the following reduction.

THEOREM 10.1. *Given a combinatorial algorithm $A_{SSRP}(n, m)$ whose runtime is $T(n, m)$, there is a combinatorial algorithm for the $BMM(n, m)$ problem whose runtime is $O(\sqrt{n} \cdot T(O(n), O(m))$.*

Proof. For simplicity, assume $m \geq n$. Let X, Y be two $n \times n$ boolean matrices with total m non-zero entries and denote by $Z = X \cdot Y$ the result of the boolean multiplication of X and Y. We show how to compute Z in $O(\sqrt{n} \cdot T(O(n), O(m))$ time.

First of all we create a graph G_0 containing 3 layers of n vertices: $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_n\}$, and $C = \{c_1, \ldots, c_n\}$. Each of the sets A, B, C is an independent set of n vertices. For every $1 \leq i, j, k \leq n$, there is an edge between a_i and b_k iff $X[i, k] = 1$, and there is an edge between b_k and c_j iff $Y[k, j] = 1$.

We construct $\sqrt{n}$ graphs $G_1, \ldots, G_{\sqrt{n}}$, each graph with $O(n)$ vertices and $O(m)$ edges. The k^{th} graph G_k (for $1 \le k \le \sqrt{n}$) will discover the values of $Z[(k-1) * \sqrt{n} + i, j]$ for all $1 \le i \le \sqrt{n}, 1 \le j \le n$. The idea is to construct a path $P_k = (v_{k,1}, \ldots, v_{k,\sqrt{n}}$ of length $\sqrt{n}-1$ of auxiliary vertices, and let $e_{k,i} = (v_{k,i}, v_{k,i+1})$ for every $1 \le i < \sqrt{n}$ be the edges of the path P_k. Let $s_k = v_{k,\sqrt{n}}$ be the source vertex of the graph G_k.

We connect $v_{k,i}$ with the vertex $a_{(k-1)*\sqrt{n}+i}$ by a path $P_{k,i}$ of $2(\sqrt{n}-i)+1$ auxiliary vertices. Observe that $v_{k,1}$ is connected to $a_{(k-1)\sqrt{n}}+1$ via a path of length 1, $v_{k,2}$ is connected to $a_{(k-1)\sqrt{n}+2}$ via a path of length 3, …, and $v_{k,\sqrt{n}} = s_k$ is connected to $a_{k\sqrt{n}}$ via a path of length $2\sqrt{n}-1$.

It follows that the following procedure will reveal the entries of $Z[(k-1)*\sqrt{n}+i, j]$ for all $1 \le i \le \sqrt{n}, 1 \le j \le n$. In the graph G_k run SSRP from s_k. If the shortest path from s_k to c_j when non of the edges fail is of length $\sqrt{n}+3$ then $Z[(k-1)\sqrt{n}+1, j] = 1$ (if there is a path $a_1 \to b_\ell \to c_j$ then the shortest path from s_k to c_j is $v_{k,\sqrt{n}} \to v_{k,\sqrt{n}-1} \to \ldots \to v_{k,1} \to a_1 \to b_\ell \to c_j$ whose length is $\sqrt{n}+3$), otherwise $Z[(k-1)\sqrt{n}+1, j] = 0$. If the shortest path from s_k to c_j when e_1 fails is of length $\sqrt{n}+5$ then $Z[(k-1)\sqrt{n}+2, j] = 1$ (if there is a path $a_2 \to b_\ell \to c_j$ then the shortest path from s_k to c_j in $G_k \setminus \{e_1\}$ is $v_{k,\sqrt{n}} \to v_{k,\sqrt{n}-1} \to \ldots \to v_{k,2} \to P_{k,2} \to a_2 \to b_\ell \to c_j$ whose length is $\sqrt{n}+5$), otherwise $Z[(k-1)\sqrt{n}+2, j] = 0$. In general, for every $1 \le i \le \sqrt{n}$, if the shortest path from s_k to c_j when e_i fails is of length $\sqrt{n}+1+2i$ then $Z[(k-1)\sqrt{n}+i, j] = 1$ (if there is a path $a_i \to b_\ell \to c_j$ then the shortest path from s_k to c_j in $G_k \setminus \{e_i\}$ is $v_{k,\sqrt{n}} \to v_{k,\sqrt{n}-1} \to \ldots \to v_{k,i} \to P_{k,i} \to a_i \to b_\ell \to c_j$ whose length is $\sqrt{n}+1+2i$), otherwise $Z[(k-1)\sqrt{n}+i, j] = 0$.

A recent line of research regarding conditional lower bounds is to reprove and strengthen many conditional lower bounds using a unified approach by reductions to a problem called Online Boolean Matrix-Vector multiplication (OMv) [16]. One of the goals of this approach is to provide stronger lower bounds, showing that many conditional lower bounds apply not only in the combinatorial settings, but also apply for algebraic algorithms. The OMv problem, in some manner, restricts the algorithm from using Strassen-like algorithms. Therefore, a conditional lower bound given by a reduction from the OMv problem implies that the given lower bound cannot be improved by an algebraic approach, *e.g.*, by using matrix multiplication.

We note that the conditional lower bound of $\widetilde{\Omega}(m\sqrt{n})$ cannot be achieved by a reduction from OMv. To see that, recall that [15] proved that for positive integer edge weights in the range $[1, M]$, SSRP can be computed by an algebraic algorithm in $\widetilde{O}(Mn^\omega)$ time. In our case of an undirected unweighted graph in holds that $M = 1$, and thus according to [15] SSRP can be computed by an algebraic algorithm in $\widetilde{O}(n^\omega)$ time, which is better than the combinatorial conditional lower bound of $\widetilde{\Omega}(m\sqrt{n})$ for dense graphs (*e.g.*, when $m = \Theta(n^2)$). Thus, our algorithm which is near optimal for combinatorial algorithms (with $\widetilde{O}(m\sqrt{n})$ upper bound and conditional lower bound of $\widetilde{\Omega}(m\sqrt{n})$ by a reduction from BMM) is not optimal for algebraic algorithms, as the algebraic SSRP algorithm of [15] runs in $\widetilde{O}(n^\omega)$. In fact, the claim above suggests that one cannot prove a conditional lower bound for SSRP by giving a reduction from any non-combinatorial conjecture to SSRP that implies that SSRP can be solved in time better than $\widetilde{O}(n^\omega)$.

10.1 Conditional Lower Bound For Weighted Graphs

In this Section we explain why it is hard to alter our $\widetilde{O}(m\sqrt{n}+n^2)$ SSRP algorithm to handle weighted graphs in the same asymptotic runtime. More generally, we refer to a reduction (which can be concluded from prior work) that a combinatorial algorithm for SSRP in weighted undirected graphs yields an algorithm of the same asymptotic runtime for APSP in weighted undirected graphs. Assuming the APSP conjecture that no combinatorial algorithm for APSP runs in subcubic $O(n^{3-\epsilon})$ time (for constant $0 < \epsilon$) this implies that a combinatorial algorithm for SSRP in weighted directed graphs cannot be solved in subcubic time.

The reduction in this section works both for directed and undirected graphs.

The following theorem is obtained by combining [31] and the observation in [2] showing that the reduction from APSP to Replacement Paths can preserve sparsity. In the full version of the paper we give a self-contained proof to Theorem 10.2.

THEOREM 10.2. *Let $A(m; n; W)$ be an algorithm for the weighted directed (undirected) SSRP problem with n vertices, m edges whose weights are integers in the range $[-W, W]$. Then, there is an algorithm for the static APSP problem for weighted graphs that runs in $O(A(m + 2n - 1; 2n; 2n^2W) + m + n^2)$ time.*

In particular, any comparison-based, or any combinatorial algorithm whose runtime is independent of W takes $\Omega(APSP(n, m))$ time, where $APSP(n, m)$ is the time required to compute the all pairs shortest paths in a graph with n vertices and m edges.

References

[1] Yehuda Afek, Anat Bremler-Barr, Haim Kaplan, Edith Cohen, and Michael Merritt. Restoration by path concatenation: fast recovery of mpls paths. *Distributed Computing*, 15(4):273–283, Dec 2002.

[2] Udit Agarwal and Vijaya Ramachandran. Fine-grained complexity and conditional hardness for sparse graphs. *CoRR*, abs/1611.07008, 2016.

[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, pages 34–43, 2008.

[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)*, pages 101–110, 2009.

[5] Davide Bilò, Luciano Gualà, Stefano Leucci, and Guido Proietti. Compact and fast sensitivity oracles for single-source distances. In *ESA*, volume 57 of *LIPIcs*, pages 13:1–13:14, 2016.

[6] Shiri Chechik, Sarel Cohen, Amos Fiat, and Haim Kaplan. $(1+\epsilon)$ approximatee f-sensitive distance oracles. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 1479–1496, 2017.

[7] Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. f-sensitivity distance oracles and routing schemes. *Algorithmica*, 63(4):861–882, 2012.

[8] Keerti Choudhary. An Optimal Dual Fault Tolerant Reachability Oracle. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 130:1–130:13, 2016.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

[10] Camil Demetrescu and Giuseppe F. Italiano. Experimental analysis of dynamic all pairs shortest path algorithms. *ACM Trans. Algorithms*, 2(4):578–601, October 2006.

[11] Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, January 2008.

[12] Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 506–515, 2009.

[13] Yuval Emek, David Peleg, and Liam Roditty. A near-linear time algorithm for computing replacement paths in planar directed graphs. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 428–435, 2008.

[14] David Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998.

[15] 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, 20-23, 2012*, pages 748–757, 2012.

[16] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC)*, pages 21–30, 2015.

[17] J. Hershberger and S. Suri. Vickrey prices and shortest paths: what is an edge worth? In *Proceedings 2001 IEEE International Conference on Cluster Computing*, pages 252–259, Oct 2001.

[18] David R. Karger, Daphne Koller, and Steven J. Phillips. Finding the hidden path: Time bounds for all-pairs shortest paths. *SIAM Journal on Computing*, 22(6):1199–1217, 1993.

[19] Philip N. Klein, Shay Mozes, and Oren Weimann. Shortest paths in directed planar graphs with negative lengths: A linear-space $o(n \log^2 n)$-time algorithm. *ACM Trans. Algorithms*, 6(2):30:1–30:18, April 2010.

[20] François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303, New York, NY, USA, 2014. ACM.

[21] Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Trans. Program. Lang. Syst.*, 1(1):121–141, January 1979.

[22] 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.

[23] 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, June 2001.

[24] Enrico Nardelli, Guido Proietti, and Peter Widmayer. Finding the most vital node of a shortest path. *Theor. Comput. Sci.*, 296(1):167–177, March 2003.

[25] Noam Nisan and Amir Ronen. Algorithmic mechanism design (extended abstract). In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 129–140, New York, NY, USA, 1999. ACM.

[26] Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, October 2011.

[27] 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, October 2012.

[28] Oren Weimann and Raphael Yuster. Replacement paths and distance sensitivity oracles via fast matrix multiplication. *ACM Trans. Algorithms*, 9(2):14, 2013.

[29] Virginia Vassilevska Williams. Faster replacement paths. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, 23-25, 2011*, pages 1337–1346, 2011.

[30] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 887–898, New York, NY, USA, 2012. ACM.

[31] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010*, pages 645–654, 2010.

[32] Christian Wulff-Nilsen. Solving the replacement paths problem for planar directed graphs in $o(n \log n)$ time. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 756–765, 2010.