Simplifying and Unifying Replacement Paths Algorithms in Weighted Directed Graphs

Shiri Chechik
Blavatnik School of Computer Science, Tel Aviv University, Israel
shiri.chechik@gmail.com

Moran Nechushtan
Blavatnik School of Computer Science, Tel Aviv University, Israel
morann@mail.tau.ac.il

Abstract

In the replacement paths (RP) problem we are given a graph G and a shortest path P between two nodes s and t [1]. The goal is to find for every edge $e \in P$, a shortest path from s to t that avoids e. The first result of this paper is a simple reduction from the RP problem to the problem of computing shortest cycles for all nodes on a shortest path.

Using this simple reduction we unify and extremely simplify two state of the art solutions for two different well-studied variants of the RP problem.

In the first variant (*algebraic*) we show that by using at most n queries to the Yuster-Zwick distance oracle [FOCS 2005], one can solve the the RP problem for a given directed graph with integer edge weights in the range $[-M, M]$ in $\tilde{O}(Mn^{\omega})$ time [2] [3]. This improves the running time of the state of the art algorithm of Vassilevska Williams [SODA 2011] by a factor of $\log^6 n$.

In the second variant (*planar*) we show that by using the algorithm of Klein for the multiple-source shortest paths problem ($MSSP$) [SODA 2005] one can solve the RP problem for directed planar graph with non negative edge weights in $O(n \log n)$ time. This matches the state of the art algorithm of Wulff-Nilsen [SODA 2010], but with arguably much simpler algorithm and analysis.

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

Keywords and phrases Fault tolerance, Distance oracle, Planar graph

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.29

Category Track A: Algorithms, Complexity and Games

Funding This publication is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803118 UncertainENV).

Acknowledgements This publication is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803118 UncertainENV)

1 Introduction

The replacement paths problem is defined as follows. We are given a graph G and a shortest path P connecting two nodes s and t. The problem is to find for every edge $e \in P$ a shortest path from s to t in the graph $(V, E \backslash e)$. As in previous works, we focus on computing the lengths of the replacement paths instead of the paths itself. Our results can be modified to return the paths as well.

[1] For the rest of this paper we assume $G = (V, E)$, $n = |V|$, $m = |E|$.
[2] ω denotes the matrix multiplication exponent ($\omega < 2.373$).
[3] The $\tilde{O}$ notation suppresses polylogarithmic factors.

47th International Colloquium on Automata, Languages, and Programming (ICALP 2020).
Editors: Artur Czumaj, Anuj Dawar, and Emanuela Merelli; Article No. 29; pp. 29:1–29:12
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The motivation for studying replacement paths is rooted in the fact that real-world graphs are vulnerable and are subject to nodes and links failures and having backup paths is a desirable property. Replacement paths are also well motivated by other important applications. One application stems from auction theory, where we use replacement paths to compute the Vickrey pricing of edges owned by selfish agents [15, 7]. In addition, Replacement paths are also used to compute the k shortest simple paths between two given vertices, as one can compute the k shortest simple paths by k calls to the replacement paths algorithm. The k shortest simple paths has many applications in and of itself [4].

1.1 Upper bounds

For undirected graphs with arbitrary (non negative) edge weights, Malik et al. [12] gave an $O(m+n\log n)$ algorithm; Nardelli et al. [14] later improve the running time to $O(m\alpha(m,n))$[4] in the Word RAM model for positive edge weights. For directed graphs with arbitrary edge weights (possibly negative), Gotthilf and Lewenstein [5] gave an $O(mn+n^2\log\log n)$ algorithm. For unweighted directed graphs, Roditty and Zwick [16] gave a randomized combinatorial algorithm that solves the problem in $\tilde{O}(m\sqrt{n})$ time. For directed graphs with integer edge weights in the range $[-M,M]$, Weimann and Yuster [17] obtain a randomized algorithm that runs in $\tilde{O}(Mn^{1+\frac{2}{3}\omega})$ time. Vassilevska Williams [18] improved the latter by giving a randomized algorithm that runs in $\tilde{O}(Mn^{\omega})$ time for $w>2$ and in $O(Mn^{\omega+\epsilon})$ time for any $\epsilon>0$ for $\omega=2$. The replacement paths problem was also studied for special family of graphs. For planar directed graphs with non negative edge weights, Emek, Peleg, and Roditty [3] obtain a recursive algorithm that runs in $O(n\log^3 n)$ time. Klein, Mozes, and Weimann [10] improve the running time to $O(n\log^2 n)$. At last, Wulff-Nilsen [20] improved the latter by giving an $O(n\log n)$ time algorithm.

1.2 Lower bounds

Hershberger et al. [8] showed a $\Omega(m\sqrt{n})$ time lower bound for the replacement paths problem for directed graphs with non negative edge weights, in the $path-comparison$ model of Karger et al [9]. Vassilevska-Williams and Williams [19] showed that the replacement paths problem in directed graphs with arbitrary edge weights is equivalent to the all pairs shortest paths problem (APSP), under subcubic reductions. Agarwal and Ramachandran [1] showed that the All-Nodes Shortest Cycles problem (ANSC) for directed graph with arbitrary edge weights, in which we are required to find for every node a shortest cycle containing it, is at least as hard as computing the replacement paths problem in directed graphs. This reduction can be used to solve the replacement paths problem using an oracle to the ANSC problem. However, the reduction does not preserve the range of the weights. That is, the reduction they present increases the weights of the graph by a factor of n. This means that the reduction is not applicable in the algebraic variant, as it would lead to a $\tilde{O}\left((Mn)\,n^{\omega}\right)$ solution. Moreover, the reduction does not preserve planarity, and therefore is not applicable in the planar variant as well.

1.3 Our result

The first result of this paper is a simple linear time reduction from the RP problem to the problem of computing shortest cycles for all nodes on a shortest path. The reduction maps the graph G into a new graph G^r with the following properties; The mapping from

[4] $\alpha(m,n)$ is the functional inverse of the Ackermann function.

G to G^r is invertible and is the inverse of itself. i.e. $(G^r)^r = G$. The reduction preserves the range of the edge weights i.e. if G has integer weights in the range $[-M, M]$ then so does G^r. The reduction preserves planarity i.e. if G is planar then so does G^r. The first property implies that the problem of computing shortest cycles for all nodes on a shortest path is equivalent to the RP problem. The second property enable us to solve the RP problem for directed graphs with integer edge weights in the range $[-M, M]$ in $O(Preprocess_{\mathcal{A}}(n, m, M) + n \cdot Query_{\mathcal{A}}(n, m, M))$ time, where $\mathcal{A}$ is a distance oracle. In particular, using Yuster-Zwick distance oracle, we solve the problem in deterministic $\tilde{O}(Mn^{\omega})$ time. Vassilevska Williams solved the problem using few techniques such as; Node sampling, graph compression and recursion. The running time of Vassilevska Williams's algorithm is at least $\log^{\frac{\omega}{\omega-2}}(n) \cdot Preprocess_{\mathcal{A}}(n, m, M)$. Therefore, when considering polylogarithmic factors, our algorithm improves the running time of Vassilevska Williams's algorithm by a factor of $\log^{\frac{\omega}{\omega-2}}(n)$. Hence, for $\omega < 2.373$ we improve the running time by at least a factor of $\log^6(n)$. The third property enable us to solve the RP problem for directed planar graphs with non negative edge weights using an oracle to the $MSSP$ problem. In particular, using Klein algorithm for the $MSSP$ problem, we solve the problem in $O(n \log n)$ time. The $MSSP$ problem is as follows; Given a planar graph G, and a list of distance queries from a source to a target, such that all sources share the same face, answer all distance queries. In fact, our reduction is to a simpler problem than the $MSSP$ problem, that is, computing at most $O(n)$ distance queries where both the source and the target are on the given face. Moreover, the boundary of the face consists of two shortest paths. The fact that our reduction is to a simpler problem than the $MSSP$ problem gives additional insights on the RP problem that might lead to further improvements.

2 Preliminaries

2.1 General notations

Let $G = (V, E)$ be a directed graph with n vertices and m edges, and let w be a weight function over E.

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 in P. We assume G does not contain negative cycles, hence the distance between every two nodes $u, v \in V$ is well defined and denoted by $d(u, v)$.

Let $e \in E$, we denote by $G \backslash e$ the graph $(V, E \backslash \{e\})$, and by $d_e(u, v)$ the distance from u to v in the graph $G \backslash e$.

Path concatenation: Given two paths $\Omega_1 = \langle u, ..., v \rangle$ $\Omega_2 = \langle v, ..., w \rangle$ we denote by $\Omega_1 \cdot \Omega_2$ the concatenation of Ω_1 and Ω_2.

Path slicing: Given a path $\Omega = \langle x_0, x_1, ..., x_k \rangle$, we denote by $\Omega[x_i, ..., x_j]$ for $i \leq j$ the subpath of Ω connecting x_i to x_j.

We denote the set $\{0, 1, ..., N\}$ by $[N]$.

2.2 Distance oracle

▶ **Definition 1.** Distance product - Let A be an $m \times n$ matrix and B an $n \times p$ matrix, where A and B have entries from $\mathbb{Z} \cup \{\infty\}$. Then their distance product $A \star B$ is an $m \times p$ matrix defined as

$$(A \star B)_{ij} = \min_{k \in [n]} A_{ik} + B_{kj}$$

▶ **Theorem 2** (Alon, Galil and Margalit [2]). *One can compute the distance product of two $n \times n$ matrices with entries in $[-M, M]$ in $\tilde{O}(Mn^{\omega})$ time.*

▶ **Theorem 3** (Yuster-Zwick [21]). *Given a directed graph G with edge weights in $[-M, ..., M]$, one can compute in $\tilde{O}(Mn^{\omega})$ time a $n \times n$ matrix A, so that the (i, j) entry of the distance product $A \star A$ is the distance between nodes i and j in G.*

By Theorem 3, there exists a distance oracle $\mathcal{A}$ with the following preprocessing and query time; $Preprocess_{\mathcal{A}}(n, m, M) = \tilde{O}(Mn^{\omega})$ and $Query_{\mathcal{A}}(n, m, M) = O(n)$. At the preprocess step, we compute the matrix A, and we answer a distance query between two nodes, i and j, by computing the (i, j) entry of $A \star A$.

2.3 Replacement path

Let G be a weighted directed graph, and let $P = \langle v_0, ..., v_k \rangle$ be a shortest path in G from $s = v_0$ to $t = v_k$.

We think of the path $P = \langle v_0, .., v_k \rangle$ as going from left to right, so for $v_i, v_j \in P$ whenever we say v_i is to the left (respectively right) of v_j we mean $i \leq j$ (respectively $i \geq j$).

▶ **Definition 4.** Let D be simple path connecting v_i to v_j for $i < j$. We call D a detour from v_i to v_j for the path P, if $D \cap P = \{v_i, v_j\}$ and $E(D) \cap E(P) = \emptyset$ (note that we need the second requirement for the case that $j = i + 1$).

Let D be a detour for the path P, connecting v_i to v_j. We denote by $Prefix(P, D)$ the path $P[v_0, ..., v_i]$, by $Suffix(P, D)$ the path $P[v_j, ..., v_k]$, and by $S(P, D)$ (S for segment) the path $P[v_i, ..., v_j]$. We say the detour D is skipping $e = (u, v)$ if v_i is to the left of u and v_j is to the right of v.

When P is clear from the context we abbreviate $Prefix(P, D)$ and simply write $Prefix(D)$ instead, and similarly for $Suffix(D)$ and $S(D)$.

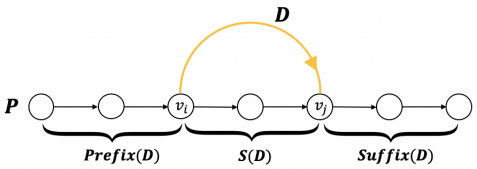

Figure 1 Partition of the path P, with respect to a detour D, into $Prefix(D)$, $S(D)$ and $Suffix(D)$.

The following Lemma is folklore.

▶ **Lemma 5.** *Let P be a shortest path in G connecting two nodes $s, t \in V$, and let $e \in P$. Suppose t is reachable from s in $G \backslash e$, then there exists a detour D skipping e s.t. $Prefix(D) \cdot D \cdot Suffix(D)$ is a shortest path from s to t in $G \backslash e$.*

In the next section we present the idea of reducing the RP problem to the problem of computing shortest cycles for all nodes on a shortest path. The results of this section are applicable for both the algebraic variant and the planar variant as well. In Section 4 we focus on directed graphs with integer weights in the range $[-M, M]$, and in section 5 we focus on directed planar graphs with non negative weights.

3 Replacement paths in weighted directed graphs

Let $s, t \in V$, and $P = \langle v_0 = s, v_1, ..., v_k = t \rangle$ be a shortest path connecting s to t in G.

We first start with two simple observations.

Observation 1. Let D be a detour of P. If we flip the orientation of P, i.e. replace every edge $e = (u, v)$ in P, with a new edge $e^r = (v, u)$, then in the new graph, the concatenation of D with $S(D)$ forms a directed cycle.

Observation 2. Let $e \in P$, and let D be a detour skipping e. D minimizes $w(Prefix(D)) + w(D) + w(Suffix(D))$ if and only if it minimizes $w(D) - w(S(D))$. This holds, as $P = Prefix(D) \cdot S(D) \cdot Suffix(D)$.

These two simple observations lead us to consider a new graph where we flip the orientation of P, and flip the sign of the weights of P. As we prove later, the minimal cycle containing e in the new graph, corresponds to some detour D skipping e, that minimize the expression $w(D) - w(S(D))$. The fact that the weight of the minimal cycle containing e is not smaller than what we are actually looking for ($\min_{D\ skip\ e} w(D) - w(S(D))$), requires some details.

▶ **Definition 6.** Let $Q = \langle u_0, u_1, .., u_N \rangle$ be a path in G, we denote by Q^r the path $\langle u_N, ..., u_1, u_0 \rangle$ with new weights $w^r(u_{i+1}, u_i) = -w(u_i, u_{i+1})$.

In other words, we flip the orientation of Q and flip the sign of it's weights.

▶ **Definition 7.** (See Figure 2 for illustration) Denote by $G^r = (V^r, E^r)$ the following graph:

$V^r = V$

$E^r = E \cup E(P^r) \backslash E(P)$

and denote by $G^+ = (V^+, E^+)$ the following graph:

$V^+ = V$

$E^+ = E \cup E(P^r)$

Finally, let $P^+ = P \cup P^r$

▶ Remark 8. If there were already edges in the opposite direction of P then we disregard them. Notice $w(v, u) \geq -w(u, v)$ (as otherwise we would have a negative cycle), hence after adding an edge (v, u) with weight of $-w(u, v)$ there is no reason to keep the original weight of (v, u) when considering shortest paths.

In the following we show that G^+ does not contain negative cycles.

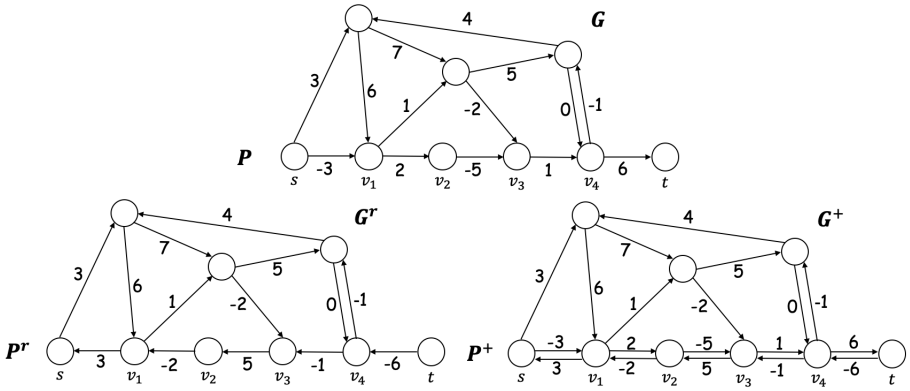

Figure 2 An example of transformation of a graph G into G^r and G^+.

▶ **Lemma 9.** *Consider the graph G^+, and let $u, v \in P^+$. The simple path from u to v that lies in P^+ is a shortest path (notice u can be to the left of v, to the right of v, or even equal to v).*

Proof. We first show that there is a shortest path (not necessarily simple) from u to v that is fully contained in P^+. Let $P' = \langle x_0 = u, ..., x_N = v \rangle$ be a shortest path connecting u to v in G^+. The proof is by induction on the number of edges of P'. The base case, $|P'| = 0$ is clear, so we assume $|P'| \geq 1$. If the first edge of P' belongs to P^+, we can easily continue by induction over $P'[x_1, .., x_N]$ so we assume otherwise. Let $x_i \in P'$ be the first vertex except x_0 itself that belongs to P (x_i might be equal to x_N). We partition P' into two paths, $P'_1 = P'[x_1, .., x_i]$ $P'_2 = P'[x_i, .., x_N]$. Notice that the edges of P'_1 are not in P^+ (hence the edges of P'_1 belong to E). We separate to two cases:

1. x_i is to the right of x_0. In that case we replace P'_1 with the path $P_1 \coloneqq P[x_0, ..., x_i]$.

   $w(P'_1) \geq d(x_0, x_i) = w(P[x_0, ..., x_i])$, where the first inequality follows as P'_1 is edge disjoint from P^+ and therefore is contained in G.

2. x_i is to the left of x_0. In that case we replace P'_1 with the path $P_1 \coloneqq P^r[x_0, ..., x_i]$.

   Notice $P[x_i, ..., x_0] \cdot P'_1$ is a cycle in G hence its weight is non negative, so we have the following:
   $w(P'_1) + w(P[x_i, ..., x_0]) \geq 0 \implies w(P'_1) \geq -w(P[x_i, ..., x_0]) = w^r(P^r[x_0, ..., x_i])$.

Therefore we can replace P'_1 with P_1 without increasing the total weight of P'. By the induction hypothesis we can replace P'_2 with a path P_2 that lies in P^+ without increasing the total weight of P'. The concatenation of the paths P_1 and P_2 is a path from u to v that lies entirely in P^+ and it's length is at most $w(P')$.

Finally, notice we can assume the shortest path from u to v that lies in P^+, lies entirely in P or entirely in P^r, depending if u is to the left or to the right of v (there is no reason to go left and right over the same edge as the weights will cancel each other). ◀

▶ **Corollary 10.** *The graph G^+ does not contain negative cycles (hence G^r as well).*

Proof. Let C be a cycle in G^+. If C does not contain any vertex in P we are done (since the cycle C exists in G, and G does not contain negative cycles), so we assume otherwise. Let $v \in C \cap P$. Note that C is a path from v to itself. By Lemma 9 we can replace C with a simple path in P^+ from v to itself without increasing the cycle weight, that path is simply $\langle v \rangle$ with zero weight. ◀

▶ **Corollary 11.** *The path P^r is a shortest path in the graph G^r.*

Proof. By Lemma 9, P^r is a shortest path in G^+, hence it's also a shortest path in G^r (G^r is a subgraph of G^+ that contains P^r) ◀

▶ **Lemma 12.** *Let $e = (u, v) \in P$. Suppose that v is reachable from u in G^r, then there exists a shortest cycle in G^r containing e^r of the form $D \cdot S(D)^r$ where D is a detour skipping e.*

Proof. Let $C = \langle x_0, ..., x_N \rangle$ be a minimal cycle in G^r containing e^r s.t. $x_0 = v, x_1 = u$ and $x_N = v$.

Let $l \in [N]$ be the largest index s.t. $x_l \in P$ and x_l is to the left of u in P.

Let $r \in \{l + 1, ..., N\}$ (i.e. r is greater than l) be the smallest index s.t. $x_r \in P$ (notice it implies that x_r is to the right of v in P). By Corollary 11 we can replace $C[x_0, ..., x_l]$ with $P^r[v, ..., x_l]$ and $C[x_r, ..., v]$ with $P^r[x_r, ..., v]$ without increasing the weight of the cycle. The middle part $D \coloneqq C[x_l, ..., x_r]$ is a detour skipping e, and $P^r[x_r, ..., x_l] = S(D)^r$. Hence $D \cdot S(D)^r$ is a cycle containing e^r with weight of at most $w^r(C)$. ◀

Notice the similarity of Lemma 5 and Lemma 12.

▶ **Definition 13.** Let $e \in P$. We denote by C_e the shortest cycle in G^r containing e^r. Moreover, by Lemma 12 we may assume that C_e is of the form $D \cdot S(D)^r$ for some detour D skipping e. If e^r is not contained in any cycle in G^r, then C_e is defined as having weight infinity (we define this to ease notation).

We denote by d^r the distance function over G^r.

▶ Remark 14. For any edge $e = (u, v) \in P$, $w^r(C_e) = d^r(u, v) - w(u, v)$.

▶ **Theorem 15.** *For any edge $e = (u, v) \in P$, $d_e(s, t) = w(P) + w^r(C_e)$ $(= w(P) - w(u, v) + d^r(u, v))$.*

Proof. From Lemma 5 it follows that $d_e(s,t) = \min_{D \text{ skip } e} w(Prefix(D)) + w(D) + w(Suffix(D))$ (where the right expression equals to ∞ if there is no such detour D).

From Lemma 12 it follows that $\min_{D \text{ skip } e} w(D) - w(S(D)) = \min_{C \text{ contain } e^r} w^r(C)$.

Hence in total we have:

$d_e(s,t) = \min_{D \text{ skip } e} w(Prefix(D)) + w(D) + w(Suffix(D)) = w(P) + \min_{D \text{ skip } e} w(D) - w(S(D)) = w(P) + \min_{C \text{ contain } e^r} w^r(C) = w(P) + w^r(C_e)$ ◀

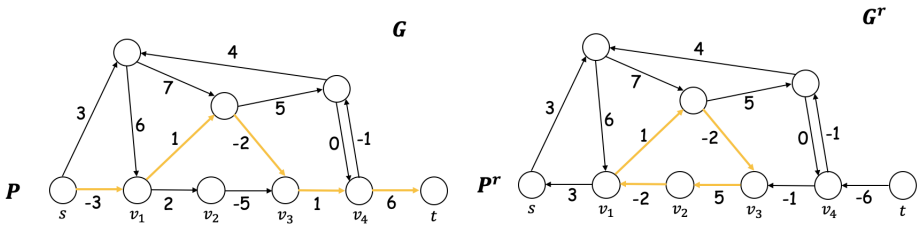

Figure 3 An example of a shortest path in $G \backslash (v_1, v_2)$ and the corresponding minimal cycle in G^r.

4 Replacement paths in graphs with integer weights

In this section, we aim to solve the replacement paths problem for directed graphs with integer weights in $[-M, M]$. Let $\mathcal{A}$ be a distance oracle for directed graphs with integer weights in $[-M, M]$. We let $Preprocess_{\mathcal{A}}(n, m, M)$ be the computation time of $\mathcal{A}$ to preprocess the graph and $Query_A(n, m, M)$ be the computation time of $\mathcal{A}$ to answer a distance query. Finally, we denote by $d_{\mathcal{A}}(u, v)$ the answer of the oracle $\mathcal{A}$ to the query of the distance from u to v.

▶ **Theorem 16.** *Let $\mathcal{A}$ be a distance oracle for weighted directed graph with integer weights in $[-M, M]$. Then, there is an algorithm for the replacement paths problem that runs in $O(Preprocess_{\mathcal{A}}(n, m, M) + n \cdot Query_{\mathcal{A}}(n, m, M))$ time.*

Proof. We first compute G^r in linear time. Then we preprocess G^r to compute a distance oracle $\mathcal{A}$ in $Preprocess_{\mathcal{A}}(n, m, M)$ time, and finally for every edge $e = (u, v) \in P$ we compute $d^r(u, v)$ using the oracle $\mathcal{A}$ in $Query_{\mathcal{A}}(n, m, M)$ time, and store $d_e(s, t) \leftarrow d^r(u, v) - w(e) + w(P)$ $(= w^r(C_e) + w(P))$. The number of calls to the distance query is equal to the number of edges in P which is bounded by n. The correctness of the algorithm follows directly from Theorem 15. ◀

▶ **Theorem 17.** *The replacement paths problem for weighted directed graph, with integer weights in* $[-M, M]$ *can be solved in deterministic* $\tilde{O}(Mn^{\omega})$ *time.*

Proof. We denote by $\mathcal{A}$ the distance oracle of $Yuster-Zwick$ (recall Theorem 3). $Preprocess_{\mathcal{A}}(n, m, M) = \tilde{O}(Mn^{\omega})$ and $Query_{\mathcal{A}}(n, m, M) = O(n)$.

Hence, by Theorem 16 we can solve the replacement paths problem in total time of $\tilde{O}(Mn^{\omega}) + O(n^2) = \tilde{O}(Mn^{\omega})$. ◀

Algorithm 1 ReplacementPaths.

1. Input: $[G, P, s, t]$
2. Compute G^r (for every edge in P, flip its orientation and its sign)
3. Compute a distance oracle $\mathcal{A}$ for the graph G^r
4. $d_e(s,t) \leftarrow d_{\mathcal{A}}(u,v) - w(e) + w(P)$ $for\, e = (u,v) \in P$
5. Output: $[d_e(s,t)$ $for\, e \in P]$

5 Replacement paths in planar graphs

In this section we aim to solve the replacement paths problem for directed planar graphs with non negative edge weights.

▶ **Definition 18** ($MSSP$ problem)**.** Let G be a directed planar graph with arbitrary weights (possibly negative). Let $U = \{u_1, u_2, .., u_\ell\}$ be a set of nodes sharing the same face, T be a shortest path tree rooted at u_1, and $L \subseteq U \times V$ be a list of k pairs. The $MSSP$ (Multiple-source shortest paths) problem is as follows; Given G, T, L, output $d(u, v)$ for all $(u, v) \in L$.

Let $\mathcal{A}$ be an algorithm that solves the $MSSP$ problem. We denote by $T_{\mathcal{A}}(n, k)$ the running time of $\mathcal{A}$ to solve the $MSSP$ problem for a graph with n nodes, and a list of k pairs.

▶ **Theorem 19** (Klein [11])**.** *There exists an algorithm* $\mathcal{A}$ *that solves the* $MSSP$ *problem in time* $T_{\mathcal{A}}(n, k) = O(n \log n + k \log n)$.

Let $e = (u, v) \in P$. By Theorem 15, $d_e(s, t) = w(P) + d^r(u, v) - w(u, v)$. That is, in order to solve the RP problem we need to compute $d^r(u, v)$ for every $(u, v) \in P$. We will do so by invoking the $MSSP$ algorithm. However, there are two slight obstacles with this approach. The first obstacle is that the nodes of P^r are not necessarily belong to the same face. The second obstacle is that even though the $MSSP$ algorithm can handle negative weights, it requires a computation of a shortest path tree. The graph G^r contains negative weights, thus a naive computation of a shortest path tree would be too time consuming, as the best running time for computing $SSSP$ (single source shortest path) with negative weights in planar graphs is $O\left(n \log^2 n / \log \log n\right)$ [13]. We will face the first obstacle using a standard approach of creating a new planar graph, denoted as H_1, such that all nodes of P^r (more precisely a copy of them) belong to the same face in H_1. For tackling the second obstacle, notice that the only edges in G^r with negative weights belong to $E(P^r)$. We will see how we can utilize this to compute a shortest path tree rooted at t in time $O(n)$ by creating a new planar graph, denoted by H_2.

▶ **Definition 20.** Given three edges $e_1, e_2, e_3 \in E(G^r)$ incident with v, we say that e_2 is between e_1 and e_3 if a counterclockwise traversal of the edges incident with v that begins at e_1 reaches e_2 before it reaches e_3.

Let $v_i \in P^r \setminus \{s,t\}$, and let e be an edge incident with v_i such that $e \notin E(P^r)$. We say that e is to the right of P^r at v_i, if e is between the edge (v_{i+1}, v_i) and the edge (v_i, v_{i-1}). Otherwise, we say that e is to the left of P^r.

▶ **Definition 21** (See Figure 4 for illustration)**.** We define a new graph, denoted as H_1, obtained from G^r, with the following modifications;

1. For every node $v_i \in \{v_1, v_2, ..., v_{k-1}\}$, replace v_i with two new nodes v_i^l and v_i^r (l for left and r for right).
   Notice that we leave s and t as they are. To ease notations, whenever we write v_0^l or v_0^r we actually mean v_0 $(= s)$. Similarly whenever we write v_k^l or v_k^r we actually mean v_k $(= t)$.
2. For every $i \in [k-1], x \in \{l, r\}$, add an edge $\left(v_{i+1}^x, v_i^x\right)$ with weight $w^r(v_{i+1}, v_i)$.
3. Let $v \in \{v_1, v_2, ..., v_{k-1}\}$, and let e be an edge incident with v in G^r such that $e \notin E(P^r)$. If e is to the left of P^r at v, we modify e by substitute v with v^l. Otherwise, we modify e by substitute v with v^r.

▶ **Definition 22** (See Figure 4 for illustration)**.** We define a new graph, denoted as H_2, obtained from H_1, with the following modifications;

For every $i \in [k-1], x \in \{l, r\}$, remove the edge $\left(v_{i+1}^x, v_i^x\right)$ and add an edge (t, v_i^x) with weight $d^r(t, v_i)$.

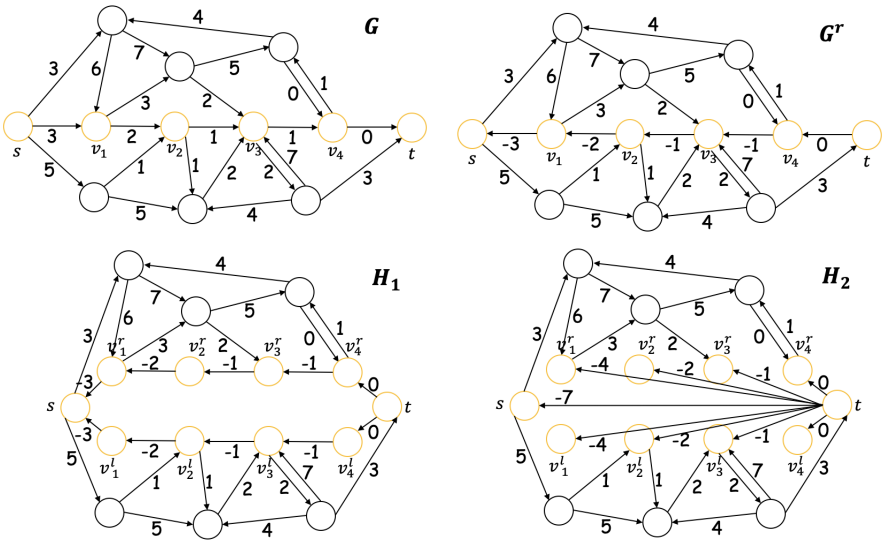

Figure 4 An example of transformation of a planar graph G into G^r,H_1 and H_2.

▶ Remark 23. H_1 and H_2 are planar graphs of size $O(n)$. Moreover, all nodes in the set $\left\{v_0^l, .., v_k^l, v_0^r, .., v_k^r\right\}$ share the same face in H_1.

Let us use the following notation; For every $i \in [k-1], x, y \in \{l, r\}$, $\rho_i^{xy} = d_{H_1}\left(v_i^x, v_{i+1}^y\right)$.

▶ **Lemma 24.** *For every $i \in [k-1]$ we have $d^r(v_i, v_{i+1}) = \min\left\{\rho_i^{ll}, \rho_i^{lr}, \rho_i^{rl}, \rho_i^{rr}\right\}$.*

Proof. Let Q be a path in H^1. We can simulate the path Q in the graph G^r, by replacing every instance of v_i^l and v_i^r with v_i. The resulting path has the same length of Q. In particular this holds for a path from v_i^x to v_{i+1}^y for any choice of $x, y \in \{l, r\}$. Therefore $d^r(v_i, v_{i+1}) \leq \min\left\{\rho_i^{ll}, \rho_i^{lr}, \rho_i^{rl}, \rho_i^{rr}\right\}$. Let P_i be a shortest path from v_i to v_{i+1} in G^r. By Lemma 12 we can assume P_i is of the form $P^r[v_i, ..., u] \cdot D_{uv} \cdot P^r[v, ..., v_{i+1}]$, where

$u \in P^r$ is to the left of v_i, $v \in P^r$ is to the right of v_{i+1}, and D_{uv} is a detour from u to v skipping the edge (v_i, v_{i+1}). The first and last edges of D_{uv} can either be to the left or to the right of P^r. Therefore there are four possible cases for the departure and entrance orientation of D_{uv} with respect to P^r. Let $x \in \{l, r\}$ be the orientation of the first edge of D_{uv}, and $y \in \{l, r\}$ be the orientation of the last edge of D_{uv}. We can simulate P_i in the graph H_1 by replacing all nodes $v_j \in P^r[v_i, ..., u]$ with v_j^x, and all nodes $v_j \in P^r[v, ..., v_{i+1}]$ with v_j^y. The resulting path is a path from v_i^x to v_{i+1}^y, and has the same length of P_i. Therefore $d^r(v_i, v_{i+1}) \geq \rho_i^{xy} \geq \min\left\{\rho_i^{ll}, \rho_i^{lr}, \rho_i^{rl}, \rho_i^{rr}\right\}$. It follows that $d^r(v_i, v_{i+1}) = \min\left\{\rho_i^{ll}, \rho_i^{lr}, \rho_i^{rl}, \rho_i^{rr}\right\}$. ◀

Our current goal is to compute a shortest path tree rooted at t in the graph H_1 in $O(n)$ time.

▶ **Lemma 25.** *For all $v \in V(H_1)\,(= V(H_2))$, $d_{H_1}(t, v) = d_{H_2}(t, v)$.*

Proof. For the first direction, let us prove $d_{H_1}(t, v) \leq d_{H_2}(t, v)$. Let Q_2 be a shortest path from t to v in H_2. We can simulate the path Q_2 in the graph H_1 as follows; If the first edge of the path Q_2 is of the form (t, v_i^x), then replace that edge with a path from t to v_i^x of length $d^r(t, v_i^x)$. The rest of the simulated path is identical to Q_2. The resulting path has the same length as Q_2. Therefore $d_{H_1}(t, v) \leq w(Q_2) = d_{H_2}(t, v)$. For the second direction, let us prove that $d_{H_2}(t, v) \leq d_{H_1}(t, v)$. Let Q_1 be a shortest path from t to v in H_1. The path $\langle t = v_k^x, ..., v_1^x, v_0^x = s\rangle$ is a shortest path in H_1 for $x \in \{l, r\}$. Let v_i^x be the last node in Q_1 that belongs to the set $\left\{v_k^l, ..., v_1^l, v_0^l, v_k^r, ..., v_1^r, v_0^r\right\}$. We can simulate the path Q_1 in the graph H_2 as follows; Replace the path $Q_1[t, ..., v_i^x]$ with a direct edge (t, v_i^x), and the rest of the simulated path is identical to Q_1. The resulting path is of the same length as Q_1. Therefore $d_{H_2}(t, v) \leq w(Q_1) = d_{H_1}(t, v)$. It follows that $d_{H_1}(t, v) = d_{H_2}(t, v)$. ◀

▶ **Theorem 26** (Henzinger, Klein, Rao and Subramanian [6])**.** *Let G be a directed planar graph with non negative weights, and let $s \in V(G)$. One can compute a shortest path tree rooted at s in $O(n)$ time.*

Notice that the only edges with negative weights in H_2 are incident with t. Let $c_0 = \min\{d^r(t, v_i) \mid i \in [k-1]\}$. We modify H_2 by increasing the weights of all edges going out from t by $-c_0$. The modified graph is a planar graph with non negative weights. Therefore by Theorem 26, we can compute a shortest path tree rooted at t, denoted as T_2, in the modified graph in $O(n)$ time. T_2 is a shortest path tree for the graph H_2.

Next we show how to modify T_2 to a shortest path tree in the graph H_1.

▶ **Definition 27.** Let $A = \left\{\left(v_{i+1}^x, v_i^x\right) \mid i \in [k-1], x \in \{l, r\}\right\}$. We define a new tree, denoted as T_1, obtained from T_2, with the following modifications;

For every edge $e = (u, v) \in A$, remove the edge $(p(v), v)$ from T_2, and insert the edge (u, v) with weight $d^r(u, v)$ to T_2, where $p(v)$ is the parent of v in T_2.

Note that for every node v_i^x for $x \in \{l, r\}$, the shortest path from t to v_i^x in T_1 is of the same length as in T_2. By Lemma 25, we conclude that T_1 is a shortest path tree in H_1.

▶ **Corollary 28.** *One can compute a shortest path tree T_1 rooted at t in H_1 in $O(n)$ time.*

▶ **Theorem 29.** *Let G be a directed planar graph with non negative weights. Let $\mathcal{A}$ be an algorithm that solves the MSSP problem on a graph with n nodes, and a list of k pairs, in time $T_{\mathcal{A}}(n, k)$. Then, there is an algorithm that solves the replacement paths problem on G, that runs in $O(T_{\mathcal{A}}(2n, 4n))$ time.*

Proof. By Corollary 28, we can compute H_1 and T_1 in $O(n)$ time.

Let $L_1 = \left\langle \left(v_i^x, v_{i+1}^y\right) \mid i \in [k-1], x \in \{l,r\} \right\rangle$ be a list of $4k$ pairs. By invoking $\mathcal{A}$ with (H_1, T_1, L_1) as an input, we have computed $\rho_i^{xy} = d_{H_1}\left(v_i^x, v_{i+1}^y\right)$ for all $i \in [k-1], x \in \{l,r\}$, in $T_{\mathcal{A}}(2n, 4n)$ time. Let $e = (v_i, v_{i+1}) \in P$. By Theorem 15, $d_e(s,t) = w(P) - w(e) + d^r(v_i, v_{i+1})$. By Lemma 24, $d^r(v_i, v_{i+1}) = \min\left\{\rho_i^{ll}, \rho_i^{lr}, \rho_i^{rl}, \rho_i^{rr}\right\}$. Thus for every edge $e = (v_i, v_{i+1}) \in P$, we store $d_e(s,t) \leftarrow w(P) - w(e) + \min\left\{\rho_i^{ll}, \rho_i^{lr}, \rho_i^{rl}, \rho_i^{rr}\right\}$. The total running time to solve the replacement paths problem is $O(n) + T_{\mathcal{A}}(2n, 4n) = O\left(T_{\mathcal{A}}(2n, 4n)\right)$. ◀

▶ **Theorem 30.** *Let G be a directed planar graph with non negative weights. Then, there is an algorithm for the replacement paths problem over G that runs in $O(n \log n)$ time.*

Proof. We denote by $\mathcal{A}$ the algorithm of Klein for the $MSSP$ problem [11]. By Theorem 19, $T_{\mathcal{A}}(n,n) = O(n \log n)$. Hence by Theorem 29, we can solve the replacement paths problem in $O(2n \log(4n)) = O(n \log n)$ time. ◀

References

1. Udit Agarwal and Vijaya Ramachandran. Fine-grained complexity for sparse graphs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 239–252, New York, NY, USA, 2018. ACM. doi:10.1145/3188745.3188888.
2. Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *Journal of Computer and System Sciences*, 54(2):255–262, 1997.
3. Yuval Emek, David Peleg, and Liam Roditty. A near-linear-time algorithm for computing replacement paths in planar directed graphs. *ACM Transactions on Algorithms (TALG)*, 6(4):64, 2010.
4. David Eppstein. Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673, 1998.
5. Zvi Gotthilf and Moshe Lewenstein. Improved algorithms for the k simple shortest paths and the replacement paths problems. *Information Processing Letters*, 109(7):352–355, 2009.
6. Monika R Henzinger, Philip Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *journal of computer and system sciences*, 55(1):3–23, 1997.
7. John Hershberger and Subhash Suri. Vickrey prices and shortest paths: What is an edge worth? In *Proceedings 2001 IEEE International Conference on Cluster Computing*, pages 252–259. IEEE, 2001.
8. John Hershberger, Subhash Suri, and Amit Bhosle. On the difficulty of some shortest path problems. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 343–354. Springer, 2003.
9. 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.
10. Philip Klein, Shay Mozes, and Oren Weimann. Shortest paths in directed planar graphs with negative lengths: A linear-space o (n log2 n)-time algorithm. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 236–245. SIAM, 2009.
11. Philip N Klein. Multiple-source shortest paths in planar graphs. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 146–155. Society for Industrial and Applied Mathematics, 2005.
12. Kavindra Malik, Ashok K Mittal, and Santosh K Gupta. The k most vital arcs in the shortest path problem. *Operations Research Letters*, 8(4):223–227, 1989.
13. Shay Mozes and Christian Wulff-Nilsen. Shortest paths in planar graphs with real lengths in o (nlog 2 n/loglogn) time. In *European Symposium on Algorithms*, pages 206–217. Springer, 2010.
14. Enrico Nardelli, Guido Proietti, and Peter Widmayer. A faster computation of the most vital edge of a shortest path? *Information Processing Letters*, 79(2):81–85, 2001.

15 Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic behavior*, 35(1-2):166–196, 2001.

16 Liam Roditty and Uri Zwick. Replacement paths and k simple shortest paths in unweighted directed graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 249–260. Springer, 2005.

17 Oren Weimann and Raphael Yuster. Replacement paths via fast matrix multiplication. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 655–662. IEEE, 2010.

18 Virginia Vassilevska Williams. Faster replacement paths. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1337–1346. Society for Industrial and Applied Mathematics, 2011.

19 Virginia Vassilevska Williams and R Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *Journal of the ACM (JACM)*, 65(5):27, 2018.

20 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*, pages 756–765. Society for Industrial and Applied Mathematics, 2010.

21 Raphael Yuster and Uri Zwick. Answering distance queries in directed graphs using fast matrix multiplication. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 389–396. IEEE, 2005.