A Polynomial-Time Algorithm for the Next-to-Shortest Path Problem on Positively Weighted Directed Graphs

Kuowen Chen*
IIIS, Tsinghua University

Nicole Wein†
University of Michigan

Yiran Zhang‡
IIIS, Tsinghua University

Abstract

Given a graph and a pair of terminals s, t, the *next-to-shortest path* problem asks for an $s \to t$ (simple) path that is shortest among all *not* shortest $s \to t$ paths (if one exists). This problem was introduced in 1996, and soon after was shown to be NP-complete for directed graphs with non-negative edge weights, leaving open the case of positive edge weights. Subsequent work investigated this open question, and developed polynomial-time algorithms for the cases of undirected graphs and planar directed graphs. In this work, we resolve this nearly 30-year-old open problem by providing an algorithm for the next-to-shortest path problem on directed graphs with positive edge weights.

*ckw22@mails.tsinghua.edu.cn
†nswein@umich.edu
‡zhangyir22@mails.tsinghua.edu.cn

1 Introduction

Given a directed graph with positive edge weights, and a pair of terminals s, t, the *next-to-shortest path* problem (also known as the *strictly second-shortest path* problem) asks for an $s \to t$ (simple) path that is shortest among all *not* shortest $s \to t$ paths (if one exists). We present the first polynomial time algorithm for this problem.

The next-to-shortest path problem was first introduced by Lalgudi, Papaefthyrniou, and Potkonjak in 1996 for the application of processing of data streams in hardware [LPP96, LPP00]. Soon after, Lalgudi and Papaefthymiou proved that the problem is NP-complete on directed graphs with non-negative edge weights [LP97], but that there is a polynomial-time algorithm for the relaxation where the path need not be simple. In their NP-hardness reduction, the weight-0 edges play a crucial role, and the case of positive edge weights remained open for almost 30 years until the present work. In the meantime there was a long line of work on the next-to-shortest path problem in *undirected* graphs.

In 2004, Krasikov and Noble gave the first polynomial-time algorithm for the undirected next-to-shortest path problem with positive edge weights [KN04]. (They also conjectured that it is NP-complete for directed graphs; we disprove this conjecture if $P \neq NP$.) Their algorithm ran in time $O(n^3 m)$ time, and was subsequently improved by Li, Sun, and Chen to $O(n^3)$ [LSC06], then by Kao, Chang, Wang, and Juan to $O(n^2)$ [KCWJ11], and finally by Wu to $O(n \log n + m)$ with the additional guarantee that the path can be found in linear time if the distances from s and t to all other vertices have already been computed [Wu13]. This was extended to non-negative edge weights by Wu, Guo, and Wang [WGW12] (see also independent work by Zhang and Nagamochi with larger running time [ZN12]). Before the general solution was known, the problem was also studied on special classes of graphs [MP06, BMP09]. See also [DHLS14] for a data structure that implicitly represents near-shortest s-t paths. Additionally, the problem of finding an *induced not-shortest* path is also solvable in polynomial time on unweighted graphs [BSS21].

For *directed* graphs with positive edge weights, Wu and Wang gave a polynomial-time algorithm of the next-to-shortest path problem when the graph is planar [WW15]. They also proved some properties of a solution for general positively weighted directed graphs. However no further progress was made, and in fact, it was not even known how to find *any* not-shortest $s \to t$ path in polynomial time.

At least 11 papers have explicitly stated the open problem solved in this paper:

Is the next-to-shortest path problem (or even the not-shortest path problem) on positively weighted directed graphs solvable in polynomial time?

In these papers [KN04, KCWJ11, WGW12, Wu13, WW15, DKQ18, AW23, AVWWX23, HMPS23, FGL+23, AHW25], the problem has been called "important", "interesting", "surprising", and "arguably one of the most tantalizing open questions within the area of graph algorithms."

1.1 Our Contribution

We provide a polynomial-time algorithm for the next-to-shortest path problem on directed graphs with positive edge weights:

Theorem 1.1. *Given a directed graph $G = (V, E, w)$, where $w : E \to \mathbb{R}_{>0}$ assigns the edge weights, and two vertices $s, t \in V$, there exists an $O(|V|^4|E|^3 \log |V|)$-time algorithm that determines if there exists a next-to-shortest path from s to t, and if so, outputs such a path.*

Our focus was to obtain a polynomial-time algorithm rather than to optimize the running time. We introduce some inefficiencies into the algorithm for the sake of clarity. For example, by modifying the algorithm to not subdivide edges, we believe it could be possible to get time $O(|V|^4|E|^2 \log |V|)$, but getting a much faster algorithm is an interesting open question.

1.2 Related Work

Detours. In the k-detour problem, the input is a graph, specified vertices s, t, and a positive integer k, and the goal is to determine if G has an $s \to t$ path of length exactly $d(s, t)+k$. This problem is fixed-parameter tractable for both undirected graphs [AVWWX23, BCDF19] and directed planar graphs [FGL$^+$23, HMPS23].

Paths of exact and forbidden lengths. The problem of finding an $s \to t$ path of an exact given length has been studied [NU02, SSMT16], as well as the more general problem of finding paths whose length falls into a specified range [DKQ18].

Disjoint Paths. The next-to-shortest path problem is closely related to disjoint path problems. Indeed, ensuring that a next-to-shortest path is simple requires the segments of the path to be disjoint from each other. For example, the NP-completeness proof for the next-to-shortest path problem in directed graphs with non-negative edge weights [LP97] uses a simple reduction from the 2-disjoint paths problem [FHW80] (given a directed graph and terminals s_1, t_1, s_2, t_2, find an $s_1 \to t_1$ path and an $s_2 \to t_2$ path that are disjoint). As another example, our algorithm employs a subroutine for solving the 2-disjoint paths problem in directed acyclic graphs [Tho12].

The k-disjoint paths problem has been studied extensively on undirected graphs [Lyn75, MP93, RS95, KKR12], directed graphs [FHW80], directed acyclic graphs [FGLZ07, FHW80, Sli10, Chi23], directed planar graphs [Sch94, CMPP13], and undirected planar graphs [LMP$^+$20, WZ23, LMP$^+$20, COO23].

There is also a wealth of work on disjoint paths problems with various objective functions such as *min-sum* (minimizing the sum of the lengths of the paths), *min-max*, and *min-min*. Additionally, there is a literature on the *disjoint shortest paths* problem where each of the paths must individually be shortest.

1.3 Organization

In Section 2, we provide a technical overview. Section 3 is the preliminaries. In Section 4, we present a reduction from a positively weighted graph to a special class of graphs, which we call (s, t)-*layered* graphs. In Section 5, we state the Key Lemma (Lemma 5.3) without proof and, based on it, present a polynomial-time algorithm for the next-to-shortest path problem. Finally, in Section 6, we prove the Key Lemma.

2 Technical Overview

First (Section 4) we reduce to the case of (s,t)-*layered* graphs, which we define in Definition 4.1. Informally, (1) every vertex is on some shortest $s \to t$ path, and (2) if we partition the vertices into layers based on their distance from s, every edge either goes exactly one layer forward (forward-edge), or an arbitrary number of layers backward (back-edge). This reduction requires careful treatment of technical details, but it is not the main conceptual idea of the proof.

Instead, the main idea is captured by the Key Lemma (Lemma 5.3), which can be informally stated as follows:

Key Lemma (Informal)**.** *Given an (s,t)-layered graph, fix a next-to-shortest path P^*, and let P_0^* be the subpath from first to last back-edge, denoting the endpoints of P_0^* as A, B. Then, there exist a pair of forward edges (X', X), (Y', Y) on the same layer as each other, so that* all *pairs of disjoint shortest paths P_1, P_2 of the form $s \to X' \to X \to A$ and $B \to Y' \to Y \to t$ (respectively) are disjoint from P_0^*.*

Intuitively, the Key Lemma is powerful because it yields a polynomial-time algorithm (Section 5) that tries all possible choices of A, B, (X', X), (Y', Y), and works roughly as follows. For each choice, pick an *arbitrary* pair P_1, P_2 of disjoint shortest paths of the form $s \to X' \to X \to A$ and $B \to Y' \to Y \to t$, respectively. It is not difficult to show that this is possible by utilizing a known linear-time algorithm for finding 2 disjoint paths in a directed acyclic graph [Tho12]. After computing P_1, P_2, remove their edges from the graph and take P_0 to be a shortest $A \to B$ path in the remaining graph. Then return the concatenation $P_1 \circ P_0 \circ P_2$. The Key Lemma guarantees that it suffices to first fix P_1, P_2 before finding P_0 since *all* such P_1, P_2 are guaranteed to be disjoint from P_0, so the resulting path is indeed a simple path.

Conceptually, the main part of the overall proof is to prove the Key Lemma. Towards doing so, we suppose for contradiction that $P_1 \cup P_2$ intersects P_0^*. Ultimately, the contradiction comes from constructing an $s \to t$ not-shortest path P that is *shorter* than P^*, which contradicts the fact that P^* is a *next-to-shortest* path. To construct such a path P, we will stitch together segments of P_1, P_2, and P^* (and other paths that we show exist). In doing so, the assumption that $P_1 \cup P_2$ intersects P_0^* is important since these intersection points enable the concatenation of segments of different paths.

In Section 6.1, we consider how $P_1 \cup P_2$ intersects with the other parts of P^* besides P_0^*, since such intersection points will also be useful to construct P: Let P_1^* be the segment of P^* from s to A and let P_2^* be the segment of P^* from B to t. Intuitively, using the fact that P_1 and P_2 are *shortest* paths, we conclude that neither P_1 nor P_2 can intersect P_1^* and then subsequently intersect P_0^* because this would constitute a "shortcut" along P^* which would result in a not-shortest path that would be shorter than P^*. For a similar reason, neither P_1 nor P_2 can intersect P_0^* and then subsequently intersect P_2^*. These properties provide structure that aids the construction of P.

Additionally, for technical reasons, it is important to establish *multiple* intersection points between $P_1 \cup P_2$ and P_0^* to provide flexibility when constructing P. To do so, we need to choose the pair of edges (X', X), (Y', Y) carefully. A key definition (in Section 6.2) is that (X', X) and (Y', Y) are defined to be edges on P_1^*, P_2^* respectively, so that:

1. there exist $s \to X$ and $B \to Y$ disjoint shortest paths whose union intersects P_0^*, and
2. there do *not* exist $s \to X'$ and $B \to Y'$ disjoint shortest paths whose union intersects P_0^*.

Item 1 establishes an intersection point v between some $P_1 \cup P_2$ and P_0^*, where v appears before the layer of X, Y. Item 2 establishes another intersection point: Consider shortest disjoint paths $s \to X' \to X \to A$ and $B \to Y' \to Y \to t$. If no such paths intersect P_0^* then we are already done, and if they do, then the intersection point must appear *after* the layer of X, Y by the condition of item 2. These two intersection points, before and after the layer of X, Y, are critical in constructing the path P.

After establishing the above properties and intersection points, in Section 6.3 we construct P by considering two cases based on whether the first point along P_0^* that is shared with $P_1 \cup P_2$ appears in a layer before or after X, Y. For each of these two cases, we stitch together path segments in two different ways. To do so, we use both the above structural properties of how P_1, P_2 can intersect with P^*, and the above pair of intersection points between $P_1 \cup P_2$ and P_0^*. The final path P is the concatenation of 7 or 9 path segments (in each of the two cases respectively), and is depicted in Figure 6 and Figure 7.

3 Preliminaries

Graphs In this paper, we only consider directed graphs with positive edge weights. We define a graph as a triple $G = (V, E, w)$, where V is the set of vertices, $E \subseteq V \times V$ is the set of edges, and $w : E \to \mathbb{R}_{>0}$ assigns a positive weight to each edge. Without loss of generality, throughout this paper, all graphs are assumed to be simple directed graphs, i.e., they contain no self-loops or multiple edges, unless stated otherwise.

When the graph is *unweighted*, we treat it as a special case where all edge weights are equal to 1; that is, $w(e) = 1$ for all $e \in E$. In this case, we abbreviate the graph as $G = (V, E)$, omitting the weight function.

Walks and Paths Let $G = (V, E, w)$ be a directed graph, where $w : E \to \mathbb{R}_{>0}$ assigns positive weights to edges. A *walk* in G is a finite sequence of vertices

$$W = (v_0, v_1, \ldots, v_k)$$

such that $(v_{i-1}, v_i) \in E$ for all $i = 1, \ldots, k$. A walk may contain repeated vertices and edges.

We define the following notation:

- **Length(or Cost)**: $w(W) := \sum_{1 \le i \le k} w(v_{i-1}, v_i)$ denotes the sum of edge weights in W.
- $\mathrm{V}(W) := \{v_0, v_1, \ldots, v_k\}$ denotes the set of vertices appearing in the walk W;
- $\mathrm{E}(W) := \big\{(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\big\}$ denotes the set of edges in W;

For two walks $W_1 = (u_0, u_1, \ldots, u_k)$ and $W_2 = (u'_0, u'_1, \ldots, u'_\ell)$, if $u_k = u'_0$, we define their **concatenation** as

$$W_1 \circ W_2 := (u_0, u_1, \ldots, u_k, u'_1, \ldots, u'_\ell),$$

which is a walk formed by appending W_2 to W_1 at the shared vertex $u_k = u'_0$. In addition, we say W_1 and W_2 are **fragments** of $W_1 \circ W_2$ in the context of concatenation.

A **path** is a walk in which all vertices are distinct, i.e., $v_i \neq v_j$ for all $i \neq j$, and hence all edges are distinct as well.

Given a path $P = (v_0, v_1, \ldots, v_k)$, a **subpath** is any contiguous subsequence of vertices. For two vertices $x = v_i$ and $y = v_j$ with $0 \leq i < j \leq k$, we write $P_{x \to y} := (v_i, v_{i+1}, \ldots, v_j)$ to denote the subpath of P from x to y.

Distance Consider a directed graph $G = (V, E, w)$. For any two vertices $u, v \in V$, we call a $s \to t$ path P shortest iff the cost of P is minimum among all paths from s to t. we define the distance $d_G(u, v)$ as the cost of the shortest path from u to v, i.e.,

$$d_G(u, v) := \min\{w(P) : P \text{ is a path from } u \text{ to } v\}.$$

Specially, if there is no path from u to v, we denote $d_G(u, v) = \infty$. Since all edge weights are positive, the shortest-path distance function $d_G(\cdot, \cdot)$ satisfies the triangle inequality:

$$d_G(u, v) + d_G(v, w) \geq d_G(u, w), \quad \text{for all } u, v, w \in V.$$

We will use this property freely throughout the paper.

Not-Shortest Path and Next-to-Shortest Path Now we show the definition of not-shortest path and next-to-shortest path.

Definition 3.1 ($s \to t$ Not-Shortest and Next-to-Shortest Paths)**.** Let $G = (V, E, w)$ be a directed graph and let $s, t \in V$. For an $s \to t$ path P, we define:

- P is an $s \to t$ **not-shortest path** if P is not a shortest path (i.e., $w(P) > d_G(s, t)$).
- P is an $s \to t$ **next-to-shortest path** if P has the minimum length among all $s \to t$ not-shortest paths. In other words, for any $s \to t$ not-shortest path P_1, if $w(P_1) < w(P)$, then $w(P_1) = d_G(s, t)$.

Definition 3.2 (The Next-to-Shortest Path Problem)**.** In the next-to-shortest path problem, we are given a directed graph $G = (V, E, w)$ and two vertices $s, t \in V$. The objective is to find a $s \to t$ next-to-shortest path or to return that there is no such path.

If we only consider the decision version of next-to-shortest path problem, we can notice that there exists a next-to-shortest path if and only if there exists a not-shortest path.

To show that a walk W is a not-shortest path, it is typically necessary to verify the following two conditions:

1. W contains no repeated vertices; that is, W is a simple path.
2. The cost of W is strictly greater than the length of a $s \to t$ shortest path, i.e., $w(W) > d_G(s, t)$.

Back-Edges Given a graph G with vertices s, t, we call an edge $(u, v) \in E$ **back-edge** if $d_G(s, u) + w(u, v) > d_G(s, v)$. Otherwise, if $d_G(s, u) + w(u, v) = d_G(s, v)$, we call (u, v) **forward-edge**. In addition, we denote $E_B(G)$ and $E_F(G)$ as the sets of back-edge and forward-edge respectively.

Similar concepts have been studied in several prior works, including [WW15]. We use the terminology *back-edge* because when restricting to the class of (s,t)-*layered* graphs that we will define in Definition 4.1, back-edges go backwards through the layers.

Lemma 3.1. *In a graph $G = (V, E, w)$, an $s \to t$ path P is an $s \to t$ not-shortest path if and only if* $\mathrm{E}(P)$ *contains at least one back-edge.*

Proof. Let path $P = (v_1, v_2, \ldots, v_k)$, where $k = |\,\mathrm{V}(P)|$. We know

$$\begin{aligned} w(P) &= \sum_{i=1}^{k-1} w(v_i, v_{i+1}) = \sum_{i=1}^{k-1} \left(d_G(s, v_{i+1}) - d_G(s, v_i) + (w(v_i, v_{i+1}) - d_G(s, v_{i+1}) + d_G(s, v_i))\right) \\ &= d_G(s,t) + \sum_{i=1}^{k-1} \left(w(v_i, v_{i+1}) - d_G(s, v_{i+1}) + d_G(s, v_i)\right) \end{aligned}$$

Due to the definition of $d_G(\cdot,\cdot)$, we know $w(v_i, v_{i+1}) - d_G(s, v_{i+1}) + d_G(s, v_i) \geq 0$ for all $1 \leq i < k$. Hence, $w(P) > d_G(s,t)$ if and only if there exists some $1 \leq j < k$, $w(v_j, v_{j+1}) - d_G(s, v_{j+1}) + d_G(s, v_j) > 0$, which means that (v_j, v_{j+1}) is a back-edge. □

Vertex Disjoint Paths Recall that the k-Vertex Disjoint Paths (k-VDP) problem asks for k vertex-disjoint paths in a graph, each connecting a specified pair of terminals.

If we only consider the decision version of next-to-shortest path (i.e., determine if there exists a next-to-shortest path), an intuitive approach is to enumerate all back-edges $(u, v) \in E$ and check if there exist 2 vertex-disjoint paths that from s to u and from v to t respectively. Unfortunately, 2-VDP is NP-hard by [FHW80].

For the 2-VDP problem in a directed acyclic graph, we have the following lemma:

Lemma 3.2 ([Tho12])**.** *Given a directed acyclic graph G with n vertices and m edges, and two vertex pairs (s_1, t_1) and (s_2, t_2), there exists an algorithm that, in $O(n+m)$ time, outputs two vertex-disjoint paths from s_1 to t_1 and from s_2 to t_2, respectively, if such a pair of paths exists.*

4 Reduction to (s,t)-Layered Graphs

In this section, we reduce the next-to-shortest path problem on directed graphs with positive edge weights to a special class of digraphs which we called (s,t)-**layered** graphs.

Definition 4.1 ((s,t)-Layered Graph)**.** Let $G = (V, E, w)$ be a positively weighted directed graph, and let $s, t \in V$. We say that G is an (s,t)-**layered graph** if the following three conditions hold:

1. For every vertex $u \in V$, $d_G(s,u) + d_G(u,t) = d_G(s,t)$. In other words, every vertex lies on at least one shortest $s \to t$ path.
2. For every edge $(u,v) \in E$, $d_G(s,u) \neq d_G(s,v)$.
3. For every edge $(u,v) \in E$ with $d_G(s,u) < d_G(s,v)$, we have that for all $x \in V$,

$$d_G(s,x) \leq d_G(s,u) \quad \text{or} \quad d_G(s,x) \geq d_G(s,u) + w(u,v).$$

Here we provide some intuition behind Definition 4.1. In an (s,t)-layered graph G, the vertex set V admits a natural partition into layers according to the distance from s. Every forward edge goes only from one layer to the next, but never skips layers. In addition, each back-edge goes strictly backwards through the layers. (These structural properties are not guaranteed in a general directed graph. For instance, if we take an arbitrary directed graph and partition the vertices into layers by their distance from s, then the back-edges may not go backwards across the layers.) For more details about the intuition, see Lemma 5.1 in Section 5. Throughout this paper, when we refer to an (s,t)-layered (or (s,t)-straight) graph, it is implicitly understood that the next-to-shortest path problem under consideration is the problem of finding an $s \to t$ next-to-shortest path (not merely deciding its existence). In order to analyze the complexity, we further introduce the following concepts:

- $V_B(G) = \{u \in V : \exists v \in V,\ (u,v) \in E_B(G) \text{ or } (v,u) \in E_B(G)\}$ is the set of vertices incident to back-edges.
- $E_{FP}(G) = \{((u_1,v_1),(u_2,v_2)) \in E_F(G) \times E_F(G) : d_G(s,u_1) = d_G(s,u_2)\}$ is the set of pairs of same-layer forward-edges.

Here is the main theorem in this section:

Theorem 4.1. *Assume that:*

- *There exists a $O\left(|V_B(G_L)|^2 |E_{FP}(G_L)| \left(|E(G_L)| + |V(G_L)| \log |V(G_L)|\right)\right)$-time algorithm that can solve the next-to-shortest path problem on any (s,t)-layered graph G_L.*

Then

- *There exists a $O(|V|^4 |E|^3 \log |V|)$-time algorithm that can solve the next-to-shortest path problem on any positively weighted directed graph $G = (V,E)$.*

To prove this theorem, we proceed in two steps. First, we reduce the original problem to the next-to-shortest path problem restricted to graphs that satisfy Property (1) of Definition 4.1, namely that every vertex lies on some shortest $s \to t$ path; we refer to such graphs as (s,t)**-straight graphs**. Second, we further reduce the next-to-shortest path problem on (s,t)-straight graphs to the case of (s,t)-layered graphs.

4.1 Reducing to (s,t)-Straight Graphs

In this subsection, we reduce the original problem to the next-to-shortest path problem restricted to (s,t)**-straight graphs**. The concept of (s,t)-straight graphs follows the terminology introduced by Berger et al. [BSS21]. In that work, the authors studied induced paths in unweighted undirected graphs and showed that the problem of finding an induced not-shortest (s,t)-path can be reduced to the same problem on (unweighted, undirected) (s,t)-straight graphs. In this section, we apply a similar, though more involved, argument to reduce the next-to-shortest path problem (on positively weighted, directed graphs) to the same problem on (positively weighted, directed) (s,t)-straight graphs.

In this subsection, we are given a positively weighted directed graph G and $s,t \in V$. To show the reduction, assume we have a polynomial-time algorithm $\mathsf{NextSP\text{-}Straight}(G;s,t)$ that can solve

the next-to-shortest path problem when G is a (s,t)-straight graph. Based on such an oracle, we present a recursive algorithm $\mathsf{NextSP}(G;s,t)$ that solves the next-to-shortest path problem when G is any positively weighted directed graph. In particular, $\mathsf{NextSP}(G;s,t)$ (or $\mathsf{NextSP\text{-}Straight}(G;s,t)$) outputs a next-to-shortest path if one exists, or $\perp$ otherwise. In the second case, where there is no next-to-shortest path, we define the weight w of the output as $w(\perp) = +\infty$.

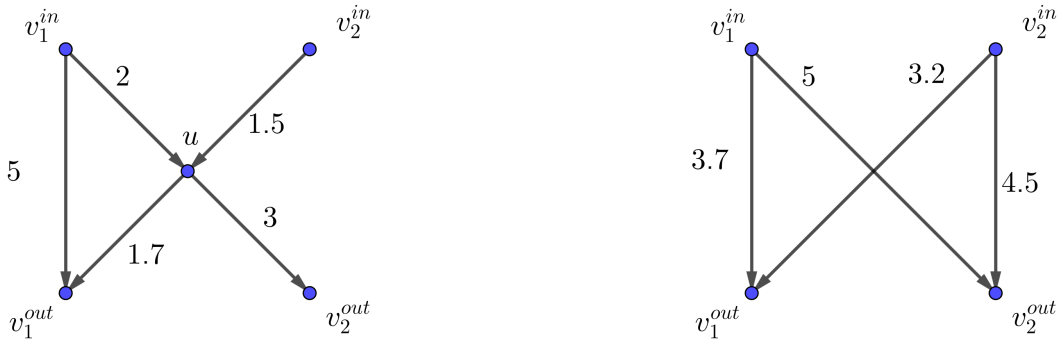
If G is already a (s,t)-straight graph, then we call $\mathsf{NextSP\text{-}Straight}(G;s,t)$. Otherwise, we can find some $u \in V$ that $d_G(s,u)+d_G(u,t) > d_G(s,t)$. If $d_G(s,u) = \infty$ or $d_G(u,t) = \infty$, we can simply output $\mathsf{NextSP}(G[V \setminus \{u\}];s,t)$ since u doesn't appear in any $s \to t$ path.

Therefore, we can assume that $d_G(s,t) < d_G(s,u) + d_G(u,t) < \infty$. Let $N^{in}(u)$ and $N^{out}(u)$ in-neighbors and out-neighbors of u, respectively. i.e.,

$$N^{in}(u) = \{v \in V : (v,u) \in E\} \quad N^{out}(u) = \{v \in V : (u,v) \in E\}$$

We would like to construct a graph $G' = (V', E', w')$ that does not contain u, since u violates the definition of (s,t)-straight. To do so, we will replace u with the set of all edges from its in-neighborhood to its out-neighborhood with appropriate weights (omitting self-loops), as shown in Figure 1. Formally, denote $N^{\times}(u) = (N^{in}(u) \times N^{out}(u)) \setminus \{(x,x) : x \in V\}$. We construct G' from G by removing vertex u and adding edges connecting the neighbors $N^{in}(u)$ and $N^{out}(u)$. In particular, $V' = V \setminus \{u\}$, $E' = \{(x,y) \in E : x \neq u, y \neq u\} \cup N^{\times}(u)$, and for $(x,y) \in E'$, we set

$$w'(x,y) = \begin{cases} w(x,y) & \text{if} \quad (x,y) \in E \text{ and } (x,y) \notin N^{\times}(u); \\ w(x,u) + w(u,y) & \text{if} \quad (x,y) \notin E \text{ and } (x,y) \in N^{\times}(u); \\ \min(w(x,y), w(x,u) + w(u,y)) & \text{if} \quad (x,y) \in E \text{ and } (x,y) \in N^{\times}(u). \end{cases}$$



(a) A subgraph of G.

(b) The modified subgraph in G'.

Figure 1: Illustration of the reduction: (a) a subgraph of G, and (b) the corresponding subgraph in G' obtained by modification.

Definition 4.2 (Shortcut Edge)**.** Let $x, y \in V'$ and let P' be an $x \to y$ path in G'. Call an edge $(v^{in}, v^{out}) \in N^{\times}(u)$ a **shortcut edge** if

$$(v^{in}, v^{out}) \notin E \text{ or } w'(v^{in}, v^{out}) < w(v^{in}, v^{out}).$$

There is a natural correspondence between paths in G and paths in G', described in the following two definitions:

Definition 4.3 (π)**.** For $x, y \in V'$ and P is a $x \to y$ path in G, we define $\pi(P)$ as follows:

- If P does not contain u, then $\pi(P) = P$.
- If P contains u, let v^{in} and v^{out} denote the predecessor and successor of u on P, respectively. Then $\pi(P)$ is the path obtained by replacing the subpath (v^{in}, u, v^{out}) with the single edge (v^{in}, v^{out}).
- Let $\pi(\perp) = \perp$.

Definition 4.4 (π')**.** Let $x, y \in V'$ and let P' be a $x \to y$ path in G'.

- If P' contains no shortcut edges, define $\pi'(P') = P'$.
- If P' contains at least one shortcut edge, let (v_1^{in}, v_1^{out}) and (v_2^{in}, v_2^{out}) be the first and last shortcut edges traversed by P' (in order along P'), and set

$$R \;=\; (v_1^{in}, u, v_2^{out}).$$

  Writing $P'_{a \to b}$ for the subpath of P' from a to b, and using $\circ$ for path concatenation, define

$$\pi'(P') \;=\; P'_{x \to v_1^{in}} \;\circ\; R \;\circ\; P'_{v_2^{out} \to y}.$$

- Let $\pi'(\perp) = \perp$.

Now we will prove a lemma relating the paths and distances in G and G':

Lemma 4.2. *For any $x, y \in V'$, we have the following properties:*

(a) For any $x \to y$ path P in G, $\pi(P)$ is a $x \to y$ path in G' and $w'(\pi(P)) \leq w(P)$.

(b) For any $x \to y$ path P' in G', $\pi'(P')$ is a $x \to y$ path in G and $w(\pi'(P')) \leq w'(P')$.

(c) $d_G(x, y) = d_{G'}(x, y)$

Proof. For (a), since $u \notin V'$, the mapping $\pi(P)$ does not change the endpoints of P. Now we will show that for every edge $(a, b) \in E$ where $a, b \in V'$, we have $w'(a, b) \leq w(a, b)$. If P does not go through u, then $w'(\pi(P)) \leq w(P)$. Otherwise, let v^{in} and v^{out} denote, respectively, the predecessor and successor of u on P. We know that $\pi(P)$ is obtained from P by removing u and adding (v^{in}, v^{out}). Since $w'(v^{in}, v^{out}) \leq w(v^{in}, u) + w(u, v^{out})$, $w'(\pi(P)) \leq w(P)$.

For (b), it is clear that $\pi'(P')$ is also a $x \to y$ path in G. Similarly, assume that P' goes through k shortcut edges $(v_1^{in}, v_1^{out}), \ldots, (v_k^{in}, v_k^{out})$.

- If $k = 0$, then it is clear that $w(\pi'(P')) = w'(P')$.
- If $k = 1$, then $w'\left(P'_{v_1^{in} \to v_k^{out}}\right) = w'(v_1^{in}, v_1^{out}) = w(v_1^{in}, v_1^{out})$. Therefore, $w(\pi'(P')) \leq w'(P')$.
- If $k \geq 2$, then $w'\left(P'_{v_1^{in} \to v_k^{out}}\right) \geq w'(v_1^{in}, v_1^{out}) + w'(v_k^{in}, v_k^{out}) \geq w(v_1^{in}, u) + w(u, v_k^{out})$, so $w(\pi'(P')) \leq w'(P')$.

For (c), we consider the shortest path from x to y ($x, y \in V'$). (a) implies $d_{G'}(x, y) \leq d_G(x, y)$, and (b) implies $d_G(x, y) \leq d_{G'}(x, y)$. Hence $d_{G'}(x, y) = d_G(x, y)$ □

Given an oracle $\mathsf{NextSP\text{-}Straight}(G; s, t)$, the algorithm is formally defined in the algorithm block below. The idea of this algorithm is to recursively call $\mathsf{NextSP}(G'; s, t)$. However, it would not be accurate to directly return the output. The reason is that u could appear in the only next-to-shortest path P in G, but the corresponding path $\pi(P)$ in G' may be a *shortest* path, since the weight of an edge $(x, y) \in E \cap N^{\times}(u)$ is defined as $\min(w(x, y), w(x, u) + w(u, y))$. For this reason, in such cases, we explicitly compute these candidate next-to-shortest paths that are not preserved in G', in step 5 of the algorithm block below.

Algorithm $\mathsf{NextSP}(G; s, t)$
Input: A positively weighted directed graph $G = (V, E, w)$, and two vertices $s, t \in V$
Output: A $s \to t$ next-to-shortest path if one exists, or report $\perp$ otherwise.

1. If G is a (s, t)-straight graph, then return $\mathsf{NextSP\text{-}Straight}(G; s, t)$.
2. Select a vertex $u \in V$ such that $d_G(s, u) + d_G(u, t) \neq d_G(s, t)$.
3. If $d_G(s, u) = \infty$ or $d_G(u, t) = \infty$, then return $\mathsf{NextSP}(G[V \setminus \{u\}]; s, t)$
4. Construct G' based on u and initialize $\hat{P} \leftarrow \pi'(\mathsf{NextSP}(G'; s, t))$.
5. For each edge $(x, y) \in E \cap N^{\times}(u)$ with $w(x, y) < w(x, u) + w(u, y)$, do:
   (a) If there exists an $s \to t$ shortest path in G that traverses (x, y), construct a path Q by replacing (x, y) with $(x, u), (u, y)$. (Since u cannot appear on any $s \to t$ shortest path, the resulting path Q is not a shortest path.)
   (b) Otherwise, set $Q \leftarrow \perp$.
   (c) If $w(Q) < w(\hat{P})$, then update $\hat{P} \leftarrow Q$.
6. Return $\hat{P}$.

Lemma 4.3. *We have the following properties:*

(a) If there exists a next-to-shortest path P in G, then either $d_{G'}(s, t) < w'(\pi(P)) = w(P)$, or $\mathsf{NextSP}(G; s, t)$ finds a next-to-shortest path in step 5 of the algorithm.

(b) If there exists a next-to-shortest path P' in G', then $d_G(s, t) < w(\pi'(P')) \leq w'(P')$.

Proof.

Proof of (a). Assume that there is a next-to-shortest path P in graph G. By Lemma 4.2 (c), we know $d_{G'}(s, t) = d_G(s, t)$. In addition, according to the definition of π, if P does not go through u, $w'(\pi(P)) = w(P)$ and $\pi(P)$ is also a not-shortest path in G' since $d_{G'}(x, y) = d_G(x, y)$. Therefore, we can assume that P goes through u, and v^{in} and v^{out} denote, respectively, the predecessor and successor of u on P.

- If $(v^{in}, v^{out}) \notin E$ or $w(v^{in}, v^{out}) \geq w(v^{in}, u) + w(u, v^{out})$, then $w'(\pi(P)) = w(P)$, and $\pi(P)$ is also a not-shortest path in G'.

- If $(v^{in}, v^{out}) \in E$, $w(v^{in}, v^{out}) < w(v^{in}, u) + w(u, v^{out})$, and there is a $s \to t$ shortest path in G that goes through (v^{in}, v^{out}), then we check the edge (v^{in}, v^{out}) in step 5 of the algorithm $\mathsf{NextSP}(G; s, t)$. Thus, we find a path of length $d_G(s, v^{in}) + w(v^{in}, u) + w(u, v^{out}) + d_G(v^{out}, t)$. Because the next-to-shortest path P goes through u, v^{in}, and v^{out}, we know that $w(P) \geq d_G(s, v^{in}) + w(v^{in}, u) + w(u, v^{out}) + d_G(v^{out}, t)$. Since our algorithm finds a path of that weight, our algorithm indeed finds a next-to-shortest path.
- Otherwise, $(v^{in}, v^{out}) \in E$, $w(v^{in}, v^{out}) < w(v^{in}, u) + w(u, v^{out})$, and there is no $s \to t$ shortest path in G that goes through (v^{in}, v^{out}). By replacing P's subpath (v^{in}, u, v^{out}) by the edge (v^{in}, v^{out}), we obtain a not-shortest path with smaller length, contradicting the fact that P is a next-to-shortest path.

Proof of (b). Let P' be a next-to-shortest path in G'. Recall that an edge (v^{in}, v^{out}) is a shortcut edge if $(v^{in}, u), (u, v^{out}) \in E$ and either $(v^{in}, v^{out}) \notin E$ or $w(v^{in}, v^{out}) > w(v^{in}, u) + w(u, v^{out})$.

- If P' does not contain any shortcut edges, then $w(\pi'(P')) = w'(P') > d_{G'}(s, t) = d_G(s, t)$.
- If P' at least one shortcut edge, then $\pi'(P')$ goes through u, which means $w(\pi'(P')) > d_G(s, t)$. In addition, based on Definition 4.4, we can notice that $w'(P') \geq w(\pi'(P'))$.

□

Lemma 4.4. *Assume there exists an algorithm that solves the next-to-shortest path problem on any (s, t)-straight graph G_S in:*

$$O(|V(G_S)|^3 \cdot |E_F(G_S)|^2 \cdot (|V(G_S)||E_F(G_S)| + |E(G_S)|) \cdot \log |V(G_S)|)$$

time, then there exists a $O(|V|^4|E|^3 \log |V|)$-time algorithm that can solve the next-to-shortest path problem on any positively weighted digraph $G = (V, E)$.

Proof. We consider the above algorithm $\mathsf{NextSP}(G; s, t)$, where G is a positively weighted directed graph, and this recursive algorithm eventually call G_S. Regarding the time complexity, we can notice the followings:

- In each recursive step the vertex set decreases by one, i.e., $|V'| = |V| - 1$. Hence, $\mathsf{NextSP}(G; s, t)$ performs at most $O(|V|)$ recursive calls before G becomes (s, t)-straight.
- Each recursive step only adds back-edges, so $|E_F(G_S)| \leq |E_F(G)| \leq |E(G)|$.
- Each step takes $O(|V|^2|E_F(G)|)$ time, since in each step we examine at most $|V|^2$ pairs of vertices, and for each pair we spend $O(1)$ time plus an additional scan over a subset of forward edges.

Consequently, if $\mathsf{NextSP\text{-}Straight}(G_S; s, t)$ runs in

$$O(|V(G_S)|^3 \cdot |E_F(G_S)|^2 \cdot (|V(G_S)||E_F(G_S)| + |E(G_S)|) \cdot \log |V(G_S)|)$$

time, then the time complexity of $\mathsf{NextSP}(G; s, t)$ is:

$$
\begin{aligned}
&O(|V(G_S)|^3 \cdot |E_F(G_S)|^2 \cdot (|V(G_S)||E_F(G_S)| + |E(G_S)|) \cdot \log |V(G_S)|) + O(|V|) \cdot O(|V|^2 |E_F(G)|) \\
&\quad \leq O(|V|^3 |E_F(G)|^2 \cdot (|V||E_F(G)| + |V|^2) \cdot \log |V|) + O(|V|) \cdot O(|V|^2 E_F(G)|) \\
&\quad \leq O(|V|^4 |E|^3 \log |V|)
\end{aligned}
$$

runs in $O(|V|^4|E|^3 \log |V|)$ time.

With regard to the correctness, since the two boundary cases are clear, we only need to discuss the case where G is not (s,t)-straight, and we have a vertex $u \in V$ such that $d_G(s,t) < d_G(s,u) + d_G(u,t) < \infty$. In this case, we construct another graph $G' = (V', E', w')$ based on u, and let the final result of $\mathsf{NextSP}(G; s, t)$ be $\hat{P}$.

- If there exists a next-to-shortest path P in G, then according to Lemma 4.3 (b), either $\mathsf{NextSP}(G'; s, t) = \perp$ or $w(\pi'(\mathsf{NextSP}(G'; s, t))) \geq w(P)$, so $w(\hat{P}) \geq w(P)$. Conversely, according to Lemma 4.3 (a), $w(\hat{P}) \leq w(P)$. Hence, $w(\hat{P}) = w(P)$.
- If there is no next-to-shortest path in G, according to Lemma 4.3 (b), $\mathsf{NextSP}(G'; s, t) = \perp$. Moreover, step 5 of $\mathsf{NextSP}(G; s, t)$ will not find a next-to-shortest path since it only explores paths in G. So $\hat{P} = \perp$.

□

4.2 From (s,t)-Straight Graphs to (s,t)-Layered Graphs

In this subsection, we reduce the next-to-shortest path problem on (s,t)-straight graphs to the problem on (s,t)-layered graphs. In order to measure how close an (s,t)-straight graph to being (s,t)-layered graph, we define a potential function $\varphi(G)$ (G is an (s,t)-straight graph) to estimate how many edges violate the properties of (s,t)-layered graphs.). In particular, for an (s,t)-straight graph G, we define:

$$
\begin{aligned}
\varphi(G) = &\#\{(u,v) \in E \mid d_G(s,u) = d_G(s,v)\} \\
&+ \#\{(u,v) \in E \mid d_G(s,u) < d_G(s,v) \text{ and } \exists x \in V, d_G(s,u) < d_G(s,x) < d_G(s,u) + w(u,v)\}.
\end{aligned}
$$

By the definition of an (s,t)-layered graph, a graph G is (s,t)-layered if and only if G is (s,t)-straight and $\varphi(G) = 0$. Given an algorithm $\mathsf{NextSP\text{-}Layered}(G; s, t)$ that can solve the next-to-shortest path problem when G is a (s,t)-layered graph, we present a recursive algorithm NextSP-Straight:

Basically, we consider an edge that violates the properties of the (s,t)-layered graph. If it is a back-edge, we remove it. If it is a forward-edge we subdivide it so that it doesn't violate the property any more.

Algorithm NextSP-Straight$(G; s, t)$
Input: A (s,t)-straight graph $G = (V, E, w)$, and two vertices $s, t \in V$
Output: A $s \to t$ next-to-shortest path if one exists, or report that none exists.

1. If G is a (s,t)-layered graph, then we return NextSP-Layered$(G; s, t)$. Otherwise, We select an edge $(u, v) \in E$ such that either $d_G(s, u) = d_G(s, v)$, or both $d_G(s, u) < d_G(s, v)$ and there exists a vertex x with $d_G(s, u) < d_G(s, x) < d_G(s, u) + w(u, v)$.

2. If (u, v) is a back-edge, we construct a new graph G' from G by removing the edge (u, v). Since G is an (s,t)-straight graph, there exists a shortest $s \to u$ path $\mathscr{P}_{s\to u}$ and a shortest $v \to t$ path $\mathscr{P}_{v\to t}$. We then compare the weight of the concatenated path $\mathscr{P}_{s\to u} \circ (u, v) \circ \mathscr{P}_{v\to t}$ with NextSP-Straight$(G'; s, t)$, and return the one of smaller weight.

3. If (u, v) is a forward-edge (i.e., $d_G(s, u) + w(u, v) = d_G(s, v)$), we denote

$$\{q_1, q_2, \ldots, q_k\} = \{d_G(s, x) : x \in V, d_G(s, u) < d_G(s, x) < d_G(s, u) + w(u, v)\},$$

where k is the number of such values, and $q_1 < q_2 < \ldots < q_k$. Then we construct a new graph $G' = (V', E', w')$ by replacing (u, v) by a list of k vertices. In particular, we create a series of new vertices $v_1, v_2, \ldots, v_k$. To simplify notation, we let $v_0 = u$, $v_{k+1} = v$, $q_0 = d_G(s, u)$, and $q_{k+1} = d_G(s, v)$). Then, we define $V' = V \cup \{v_1, v_2, \ldots, v_k\}$ and $E' = (E \setminus \{(u, v)\}) \cup \{(v_i, v_{i+1}) : 0 \le i \le k\}$. For the edge weights, we define $w'(v_i, v_{i+1}) = q_{i+1} - q_i$ for $1 \le i \le k$, and for all edges $e \in E \setminus \{(u, v)\}$, we set $w'(e) = w(e)$. Then we return NextSP-Layered$(G'; s, t)$.

Lemma 4.5. *Whenever the algorithm NextSP-Straight$(G; s, t)$ makes a recursive call on a graph $G' = (V', E', w')$, we have:*

(a) For any $z \in V$, $d_G(s, z) = d_{G'}(s, z)$.

(b) $\#\{d_G(s, z) : z \in V\} = \#\{d_{G'}(s, z) : z \in V'\}$

(c) $\varphi(G') = \varphi(G) - 1$.

Proof. In the algorithm, we select an edge (u, v) that violates the defining property of an (s,t)-layered graph.

Proof of (a). If (u, v) is a back-edge, as in step 2, for any $z \in V$, there is no $s \to z$ shortest path that goes through (u, v), so $d_{G'}(s, z) = d_G(s, z)$.

If (u, v) is a forward-edge, as in step 3, then our algorithm replaces the forward-edge by a path of the same length. For any $z \in V$, any $s \to z$ shortest path in G' that goes through any new vertices must go through u and v, and we can obtain a corresponding $s \to z$ path in G by replacing this $u \to v$ subpath by an edge (u, v). Therefore, $d_{G'}(s, z) = d_G(s, z)$.

Proof of (b) According to (a), $d_{G'}(s, z) = d_G(s, z)$ for all $z \in V$, and for new vertices, it is clear that $\{d_{G'}(s, z) : z \in V' \setminus V\} \subseteq \{d_G(s, z) : z \in V\}$.

Proof of (c) If (u, v) is a back-edge, (u, v) is removed in G' and this process does not change other edges. Therefore, $\varphi(G') = \varphi(G) - 1$.

If (u, v) is a forward-edge, (u, v) is removed in G'. With regard to the new edges, for every $0 \leq i \leq k$, there are no vertices $z \in V$ satisfying $d_{G'}(s, v_i) < d_G(s, z) < d_{G'}(s, v_{i+1}) = d_{G'}(s, v_i) + w'(v_i, v_{i+1})$. Therefore, $(v_i, v_{i+1})(0 \leq i \leq k)$ does not violate the property but (u, v) is removed, so $\varphi(G') = \varphi(G) - 1$. □

Lemma 4.6. *Assume that there exists a $O(|V_B(G_L)|^2 \cdot |E_{FP}(G_L)| \cdot (|E(G_L)| + |V(G_L)| \log |V(G_L)|))$-time algorithm that can solve the next-to-shortest path problem on any (s, t)-layered graph G_L. There exists a $O(|V(G_S)|^3 \cdot |E_F(G_S)|^2 \cdot (|V(G_S)||E_F(G_S)| + |E(G_S)|) \cdot \log |V(G_S)|)$-time algorithm that can solve the next-to-shortest path problem on any (s, t)-straight graph G_S.*

Proof. We consider the process NextSP-Straight$(G_S; s, t)$ that finally calls NextSP-Layered$(G_L; s, t)$. According to Lemma 4.5 (b), $\#\{d_{G_S}(s, z) : z \in V(G_S)\}$ does not change during the recursive process, so for each recursive step, we add at most $O(\#\{d_{G_S}(s, z) : z \in V(G_S)\})$ new vertices.

For an arbitrary recursive step with an input $G = (V, E)$ that recursively calls G', we can notice that:

- If this step removes a back-edge, then $|V_B(G')| \leq |V_B(G)|$ since the some vertices may no longer connect to back-edges.
- If this step replaces an edge by a list of vertices, then the new vertices are not connected to back-edges, so $|V_B(G)| \leq |V_B(G')|$

Therefore, $|V_B(G_L)| \leq |V_B(G_S)| \leq |V(G_S)|$

In addition, for every $q \in \{d_G(s, z) : z \in V\}$, $\#\{(u, v) \in E_F(G) : d_G(s, u) = q\} \leq |E_F(G_S)|$. Hence, $|E_{FP}(G_L)| \leq \#\{d_{G_S}(s, z) : z \in V(G_S)\} \cdot |E_F(G_S)|^2$

By Lemma 4.5 (c), our algorithm NextSP-Straight only repeats at most $\varphi(G_S) = O(|E(G_S)|)$ times, and each step takes $O(|E(G)| + |V(G)|) \leq O(|E(G_S)| \cdot |V(G_S)|)$ time. Therefore, the total time complexity is:

$$
\begin{aligned}
&O\big(|V_B(G_L)|^2 \cdot |E_{FP}(G_L)| \cdot (|E(G_L)| + |V(G_L)| \log |V(G_L)|)\big) \\
&\qquad + O(|E(G_S)| \cdot |V(G_S)|) \cdot \#\{d_{G_S}(s, z) : z \in V(G_S)\} \\
&\leq O(|V(G_S)|^2 \cdot |V(G_S)||E_F(G_S)|^2 \cdot (|V(G_S)||E_F(G_S)| + |E(G_S)|) \cdot \log(|V(G_S)| \cdot |E(G_S)|)) \\
&\qquad + O(|E(G_S)| \cdot |V(G_S)|^2) \\
&\leq O(|V(G_S)|^3 \cdot |E_F(G_S)|^2 \cdot (|V(G_S)||E_F(G_S)| + |E(G_S)|) \cdot \log |V(G_S)|)
\end{aligned}
$$

To show the correctness, let (u, v) be the edge selected in step 1. If (u, v) is a back-edge and there is a next-to-shortest path P that goes through (u, v), we consider a walk $\mathscr{P}_{s \to u} \circ (u, v) \circ \mathscr{P}_{v \to t}$. Since $d(u) \leq d(v)$ and along both $\mathscr{P}_{s \to u}$ and $\mathscr{P}_{v \to t}$, the distances from s are monotonically increasing, this walk is a simple path. In addition, we know $\mathscr{P}_{s \to u} \circ (u, v) \circ \mathscr{P}_{v \to t}$ is the shortest among the paths that traverse (u, v). Therefore, $\mathscr{P}_{s \to u} \circ (u, v) \circ \mathscr{P}_{v \to t}$ must be a next-to-shortest path. If (u, v) is a forward-edge, then we have a bijection from $s \to t$ paths in G to $s \to t$ paths in G' by replacing (u, v) with $(v_0, v_1, \ldots, v_{k+1})$ (and conversely). □

Proof of Theorem 4.1. This statement follows from the combination of Lemma 4.4 and Lemma 4.6. □

5 Polynomial-time Algorithm for (s,t)-Layered Digraphs

In the following sections, we restrict our attention to (s,t)-layered graphs. In this section, we provide a polynomial-time algorithm NextSP-Layered that can solve the next-to-shortest path problem on (s,t)-layered graphs. This algorithm is based on the Key Lemma (Lemma 5.3), which will be proved in Section 6. In the remainder of this paper, we fix an (s,t)-layered graph $G = (V, E, w)$. We first define the layer function for the (s,t)-layered graph as follows:

Definition 5.1 (Layer Function)**.** We are given a (s,t)-layered graph $G = (V, E, w)$. Let k be the number of distinct values of $d(x) = d_G(s, x)$ over all $x \in V$, and write

$$\{d(x) : x \in V\} = \{p_1, p_2, \ldots, p_k\}, \quad 0 = p_1 < p_2 < \cdots < p_k = d_G(s,t).$$

We define the **layer function** $\lambda = \lambda_G : V \to [k]$ by setting $\lambda(u) = i \quad$ whenever $d(u) = p_i$.

Based on Definition 4.1 and Definition 5.1, we adopt the following notational conventions:

- For $u \in V$, let $d(u) = d_G(s, u)$ denote the length of a shortest path from s to u.
- For a $u \to v$ path P $(u, v \in V)$, we call P **forward** if it only contains forward-edges. (Equivalently, along P the layer function λ is non-decreasing.)
- For each $u \in V$, since there exists a shortest $s \to t$ path passing through u, there exist an $s \to u$ forward path and a $u \to t$ forward path. We denote by $\mathscr{P}_{s\to u}$ and $\mathscr{P}_{u\to t}$ arbitrary but fixed choices of such forward paths, respectively.

The following lemma introduces some basic properties of λ:

Lemma 5.1. *Let $G = (V, E, w)$ be a (s,t)-layered graph with distance function $d(x) = d_G(s, x)$ for $x \in V$, and layer function $\lambda(x) = \lambda_G(x)$ for $x \in V$. Then the following hold:*

(a) For all $u, v \in V$, $d(u) < d(v) \iff \lambda(u) < \lambda(v)$.

(b) For every edge $(u, v) \in E$:

- *if (u, v) is a back-edge, then $\lambda(v) < \lambda(u)$;*
- *if (u, v) is a forward-edge, then $\lambda(v) = \lambda(u) + 1$.*

Proof. According to the definition, we let

$$\{d(x) : x \in V\} = \{p_1, p_2, \ldots, p_k\}, \quad 0 = p_1 < p_2 < \cdots < p_k = d_G(s,t).$$

Proof of (a). For any $u, v \in V$, we have

$$d(u) < d(v) \iff p_{\lambda(u)} < p_{\lambda(v)} \iff \lambda(u) < \lambda(v).$$

Proof of (b). Consider an edge $(u, v) \in E$. Choose an arbitrary vertex x such that $\lambda(x) = \lambda(u) + 1$. We consider the following two cases.

- If (u, v) is a back-edge, by Definition 4.1, we know $d_G(s, u) \neq d_G(s, v)$ (equivalently, $\lambda(u) \neq \lambda(v)$. Moreover, if $\lambda(u) < \lambda(v)$, then

$$d_G(s, u) < d_G(s, x) \leq d_G(s, v) < d_G(s, u) + w(u, v),$$

  contradicting the definition of (s, t)-layered graphs.

- If (u, v) is a forward-edge, then

$$d(u) + w(u, v) = d(v),$$

  which implies $\lambda(v) > \lambda(u)$. Assume for contradiction that $\lambda(v) > \lambda(u) + 1$. Then

$$d_G(s, u) < d_G(s, x) < d_G(s, v) = d_G(s, u) + w(u, v),$$

  again contradicting the definition.

□

According to Lemma 3.1, we know that every next-to-shortest path must contain at least one back-edge. Based on this observation, we define the Back-Edge Decomposition as follows (see Figure 2):

Definition 5.2 (Back-Edge Decomposition)**.** Given a (s, t)-layered graph $G = (V, E, w)$ with $s, t \in V$, we consider a $s \to t$ not-shortest path P. The **Back-Edge Decomposition** BE-Decomp(P) is defined as a 5-tuple $(A, B; P_1, P_0, P_2)$, where $A, B \in V$, $P_1 = P_{s \to A}, P_0 = P_{A \to B}, P_2 = P_{B \to t}$ are paths in G, and P_0 is the subpath of P starting with the first back-edge of P and ending with the last back-edge.

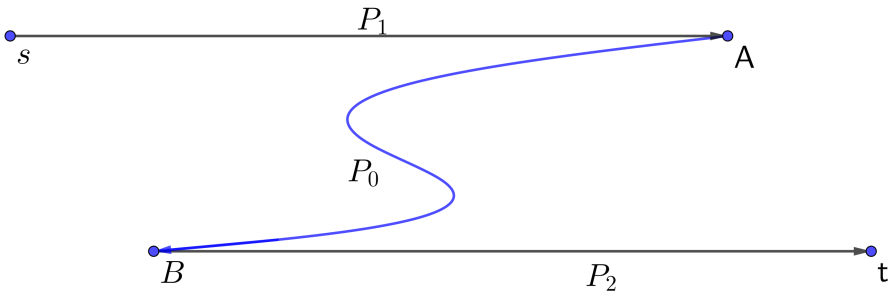


Figure 2: An illustration of Back-Edge Decomposition.

A similar concept "maximal mixed path", which corresponds to P_0 in our Definition 5.2, was used in [WW15]. Lemma 1 in [WW15] implies the following lemma. Here we reprove the lemma using our notation.

Lemma 5.2. *Given a (s, t)-layered graph $G = (V, E, w)$ and any $s \to t$ next-to-shortest path P^*, let* BE-Decomp$(P^*) = (A, B; P_1^*, P_0^*, P_2^*)$. *Then $d(B) < d(A)$, and for all $u \in \mathrm{V}(P_0^*) \setminus \{A, B\}$ (if such u exist), $d(B) < d(u) < d(A)$.*

Proof. If $\mathrm{V}(P_0^*) \setminus \{A, B\}$ is an empty set, then as P_0^* begins with a back edge, $d(B) < d(A)$. In the following, we assume that $\mathrm{V}(P_0^*) \setminus \{A, B\}$ isn't empty.

First we prove that $\forall u \in \mathrm{V}(P_0^*) \setminus \{A, B\}$, $d(B) < d(u)$. We choose $u \in \mathrm{V}(P_0^*) \setminus \{A, B\}$ with a minimum layer index. i.e., $u = \arg\min_{u' \in \mathrm{V}(P_0^*) \setminus \{A,B\}} \lambda(u')$. As P_0^* begins with a back edge, we have $d(u) \leq d(A)$. Suppose, for contradiction, that $d(u) \leq d(B)$. Then the forward path $\mathscr{P}_{s \to u}$ does not intersect with $(P_0^*)_{u \to B} \circ (P_2^*)_{B \to t}$. Let us think about $P' = \mathscr{P}_{s \to u} \circ (P_0^*)_{u \to B} \circ (P_2^*)_{B \to t}$. We will derive a contradiction by showing that P' is a not-shortest path that is shorter than P^*:

- $w(P') = w(\mathscr{P}_{s \to u}) + w((P_0^*)_{u \to B}) + w((P_2^*)_{B \to t}) = d(u) + d(t) - d(B) + w((P_0^*)_{u \to B})$
  $\leq d(A) + d(t) - d(B) + w((P_0^*)_{u \to B}) < w(P^*)$.
- Since the last edge of P_0^* must be a back-edge, due to Lemma 3.1, $w(P') > d(t)$.

Therefore, P' is a $s \to t$ not-shortest path that is shorter than P^*, which contradicts the assumption that P^* is a next-to-shortest path. Hence, we have $\forall u \in \mathrm{V}(P_0^*) \setminus \{A, B\}$, $d(B) < d(u)$.

The rest of the proof is symmetric to the proof above. We can assume that $v \in V(P_0^*) \setminus \{A, B\}$ is the vertex with maximum $d(v)$. Suppose for contradiction that $d(v) \geq d(A)$. In this case, by a symmetric argument, $P_{s \to v}^* \circ \mathscr{P}_{v \to t}$ is a not-shortest path which is shorter than P^*. □

The Key Lemma is as follows (see the accompanying Figure 3):

Lemma 5.3 (Key Lemma)**.** *Given an (s, t)-layered graph $G = (V, E, w)$ and any $s \to t$ next-to-shortest path P^*, let* BE-Decomp$(P^*) = (A, B; P_1^*, P_0^*, P_2^*)$. *There exist two edges $(X', X) \in E, (Y', Y) \in E$ satisfying the following conditions:*

- $\lambda(X') = \lambda(Y') = \lambda(X) - 1 = \lambda(Y) - 1$
- *There exists a pair of forward paths (P_1, P_2) such that*
  - *P_1 is a $s \to A$ shortest path that goes through (X', X)*
  - *P_2 is a $B \to t$ shortest path that goes through (Y', Y)*
  - $\mathrm{V}(P_1) \cap \mathrm{V}(P_2) = \emptyset$*;*

  and each pair of forward paths (P_1, P_2) that satisfies the above conditions also satisfies:

$$(\mathrm{V}(P_0^*) \setminus \{A, B\}) \cap (\mathrm{V}(P_1) \cup \mathrm{V}(P_2)) = \emptyset.$$

The proof of the lemma is deferred to Section 6. Here, based on the Key Lemma, we provide a polynomial-time algorithm NextSP-Layered. The idea of the algorithm is to try every possible A, B, X, X', Y, Y', and for each such tuple find paths P_1, P_2 using Lemma 3.2, then combine these paths with a shortest $A \to B$ path, which is guaranteed to result in a simple path by the Key Lemma.

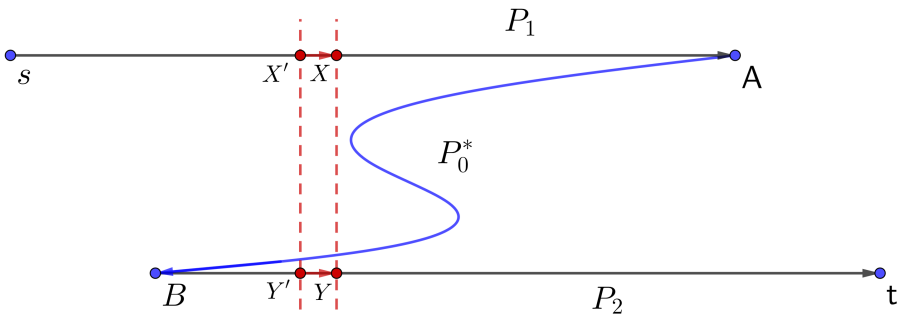


Figure 3: An illustration of $(X', X), (Y', Y)$ and P^* in Lemma 5.3

Algorithm NextSP-Layered$(G; s, t)$
Input: (s, t)-layered graph $G = (V, E, w)$, and two vertices $s, t \in V$
Output: A $s \to t$ next-to-shortest path if one exists, or report $\perp$ otherwise to indicate that no next-to-shortest path exists.

1. Initialize $\hat{P} \leftarrow \perp$
2. Enumerate all 6-tuples $(A, B, X', X, Y', Y) \in V^6$ satisfying $d(A) > d(B)$ (equivalently $\lambda(A) > \lambda(B)$), $A, B \in V_B(G)$,$(X', X) \in E, (Y', Y) \in E$ and $\lambda(X') = \lambda(Y') = \lambda(X) - 1 = \lambda(Y) - 1$. For each such tuple of vertices, proceed as follows:
   - Find two disjoint forward paths P_1, P_2 satisfying the conditions of Lemma 5.3. Since forward paths contain no back-edges, this reduces to finding two vertex-disjoint paths in a directed acyclic graph (DAG). We can first apply Lemma 3.2 to find $(P_1)_{s \to X'}, (P_2)_{B \to Y'}$, and then we call the algorithm in Lemma 3.2 again to find $(P_1)_{X \to A}, (P_2)_{Y \to t}$. After that, we concatenate the output paths with edges $(X', X), (Y', Y)$. In particular, $P_1 = (P_1)_{s \to X'} \circ (X', X) \circ (P_1)_{X \to A}$ and $P_2 = (P_2)_{B \to Y'} \circ (Y', Y) \circ (P_2)_{Y \to t}$. If no such pair (P_1, P_2) exists, proceed to the next iteration with a new tuple (A, B, X', X, Y', Y).
   - In graph $G\Big[\Big(V \setminus (\mathrm{V}(P_1) \cup \mathrm{V}(P_2))\Big) \cup \{A, B\}\Big]$, we find a shortest $A \to B$ path P_0. (If there is no such path, $P_0 \leftarrow \perp$).
   - If $P_0 \neq \perp$ and $w(P_1 \circ P_0 \circ P_2) < w(\hat{P})$, then we set $\hat{P} \leftarrow P_1 \circ P_0 \circ P_2$.
3. Return $\hat{P}$.

Theorem 5.4. *Given an (s, t)-layered graph $G_L = (V(G_L), E(G_L), w)$ with two distinguished vertices $s, t \in V$, there exists a $O(|V_B(G_L)|^2 |E_{FP}(G_L)|(|E(G_L)| + |V(G_L)| \log |V(G_L)|))$-time algorithm that either finds a $s \to t$ next-to-shortest path (if one exists) or correctly reports that no such path exists.*

Proof of Theorem 5.4 (given Lemma 5.3). We consider the above algorithm NextSP-Layered$(G_L; s, t)$. The number of choices for $A, B, (X', X), (Y', Y)$ is $O(|V_B(G_L)|^2 \cdot |E_{FP}(G_L)|)$. Consider one choice. For each choice, by Lemma 3.2, there is a $O(|V(G_L)| + |E(G_L)|)$-time algorithm that can solve 2-VDP in a DAG, so finding (P_1, P_2) takes $O(|V(G_L)| + |E(G_L)|)$ time. To find the shortest from A

to B, we can use dijkstra algorithm, which runs in time $O(|E(G_L)| + |V(G_L)| \log |E(G_L)|)$. Hence, the total time complexity is $O(|V_B(G_L)|^2 \cdot |E_{FP}(G_L)| \cdot (|E(G_L)| + |V(G_L)| \log |V(G_L)|))$

To prove the correctness, we first discuss if there exists a next-to-shortest path. If no one exists, then it is clear that our algorithm will not find a path. Otherwise, we consider an arbitrary $s \to t$ next-to-shortest path P^* with BE-Decomp$(P^*) = (A, B; P_1^*, P_0^*, P_2^*)$.

By Lemma 5.3, our algorithm examines all choices of $A, B \in V(G_L)$ and $(X', X), (Y', Y) \in E(G_L)$. Hence, we must find some feasible $A, B, (X', X), (Y, Y')$. Since every forward $s \to A$ path has the same length, and every forward $B \to t$ path has the same length, and since our algorithm computes a shortest $A \to B$ path on the remaining graph, the length of $\hat{P}$ cannot be greater than that of P^*. Moreover, because $\hat{P}$ is not a shortest path, our algorithm returns a not-shortest path whose length is no greater than $w(P^*)$. □

Now we restate the main theorem:

Theorem 1.1. *Given a directed graph $G = (V, E, w)$, where $w : E \to \mathbb{R}_{>0}$ assigns the edge weights, and two vertices $s, t \in V$, there exists an $O(|V|^4|E|^3 \log |V|)$-time algorithm that determines if there exists a next-to-shortest path from s to t, and if so, outputs such a path.*

Proof of Theorem 1.1 (given Lemma 5.3). This statement follows by combining Theorem 4.1 and Theorem 5.4. □

6 Proof of Lemma 5.3

In this section, we provide a proof for the Key Lemma (Lemma 5.3). We are given a (s, t)-layered graph $G = (V, E, w)$ with vertices $s, t \in V$. To simplify notation, we define the following:

- For $a, b, c, d \in V$, we call a pair of two paths (P_1, P_2) an $(a \to b, c \to d)$-**Pair of Disjoint Forward Paths** ($(a \to b, c \to d)$-PDFP) if P_1 is a $a \to b$ forward path (i.e., no back-edges), P_2 is a $c \to d$ forward path and these two paths are vertex-disjoint (i.e., they have no common vertices). Recall that such a pair can be found or determined not to exist in linear time by solving the 2-VDP problem on the DAG without back-edges, as stated in Lemma 3.2. In addition, if we force these two paths to go through some specific vertices $(u_1, u_2, \ldots, u_p)$ and $(v_1, v_2, \ldots, v_p)$ respectively, where p is a constant and $\lambda(u_i) = \lambda(v_i)$ for all $1 \leq i \leq p$, we abbreviate the pair as $(a \to u_1 \to \ldots \to u_p \to b, c \to v_1 \to \ldots \to v_p \to d)$-PDFP. In this case, we can divide the DAG into $p + 1$ parts according to the layers and solve 2-VDP for each part, so such a pair can also be found or determined not to exist in linear time.

- We call an $A \to B$ path P_0^* an **oracle middle path** if there exists an $s \to t$ next-to-shortest path P^* such that BE-Decomp$(P^*) = (A, B; P_1^*, P_0^*, P_2^*)$ for some other subpaths P_1^*,P_2^*. In addition, we say that such a pair (P_1^*, P_2^*) is **compatible** with P_0^* (or P_0^*-compatible).

- For an oracle middle path P_0^* from A to B, and a $(a \to b, c \to d)$-PDFP (P_1, P_2) $(a, b, c, d \in V)$, we denote $\mathcal{I}(P_0^*; P_1, P_2) = (\mathrm{V}(P_0^*) \setminus \{A, B\}) \cap (\mathrm{V}(P_1) \cup \mathrm{V}(P_2))$ as the set of intersection points between $\mathrm{V}(P_0^*) \setminus \{A, B\}$ and the union of $\mathrm{V}(P_1)$ and $\mathrm{V}(P_2)$.

- Let P_0^* be an oracle middle path from A to B. We say that a 4-tuple $(X', X, Y', Y) \in V^4$ is P_0^*-**isolated** if the following hold:

– $(X', X), (Y', Y) \in E$ and $\lambda(X') = \lambda(Y') = \lambda(X) - 1 = \lambda(Y) - 1$;
– There exists a P_0^*-compatible $(s \to X' \to X \to A, B \to Y' \to Y \to t)$-PDFP;
– For all $(s \to X' \to X \to A, B \to Y' \to Y \to t)$-PDFPs (P_1, P_2), $\mathcal{I}(P_0^*; P_1, P_2) = \emptyset$.

In other words, (X', X, Y', Y) satisfies the conditions in Lemma 5.3.

Now, we restate Lemma 5.3 using the above definitions:

Lemma (Restatement of Lemma 5.3)**.** *Given an* ***oracle middle path*** P_0^* *from A to B, there exists a* P_0^****-isolated*** *4-tuple* (X', X, Y', Y).

Here are some high-level ideas for our proof:

- We prove the Key Lemma by contradiction. During our proof, we always assume that there exists an $(s \to X' \to X \to A, B \to Y' \to Y \to t)$-PDFP (P_1, P_2) that intersects with P_0^* (i.e., $\mathcal{I}(P_0^*; P_1, P_2) \neq \emptyset$).
- First, we consider an $s \to A$ or $B \to t$ path P that may intersect with P_0^*. We discuss the relationship between P and any (P_1^*, P_2^*) that is compatible with P_0^*. We claim that the intersection $(\mathrm{V}(P_0^*) \setminus \{A, B\}) \cap \mathrm{V}(P)$ consists only of a special type of vertex that we call an **Up-Vertex**.
- Secondly, we explicitly construct a pair (X, Y) such that $d(B) \leq d(X) = d(Y) \leq d(A)$, and then further construct X' and Y'. We show that if an $(s \to X' \to X \to A, B \to Y' \to Y \to t)$-PDFP (P_1, P_2) intersects with P_0^*, then we can construct an $(s \to X \to A, B \to Y \to t)$-PDFP (P_1', P_2') that intersects P_0^* in at least two vertices $\alpha, \beta \in \mathrm{V}(P_0^*)$, where $d(\alpha) < d(X)$ and $d(\beta) > d(X)$.
- Finally, based on those two vertices α and β, we construct a next-to-shortest path with smaller length, which contradicts our assumption that the oracle middle path P_0^* corresponds to a next-to-shortest path.

6.1 Relationship between Arbitrary PDFPs and Compatible PDFPs

The following definition decomposes a path P into segments based on the intersection pattern of P with paths (P_1^*, P_2^*). See Figure 4.

Definition 6.1 (Up/Down/Gap Paths and Vertices)**.** Consider an $(s \to A, B \to t)$-PDFP (P_1^*, P_2^*) that is compatible with some oracle middle path P_0^*. For a forward path P with both endpoints in $\mathrm{V}(P_1^*) \cup \mathrm{V}(P_2^*)$, denote $V^{\text{inter}}(P_1^*, P_2^*; P) = (\mathrm{V}(P_1^*) \cup \mathrm{V}(P_2^*)) \cap \mathrm{V}(P)$. According to the order in P, we list the items of $V^{\text{inter}}(P_1^*, P_2^*; P)$: $(u_1, \ldots, u_k)$, where $k = |V^{\text{inter}}(P_1^*, P_2^*; P)|$ is the number of intersection points. Then P can be decomposed into $k-1$ subpaths: the i-th subpath is $P_{u_i \to u_{i+1}}$ for $1 \leq i < k$. We classify each subpath $P_{u_i \to u_{i+1}}$ of P into three types, based on the endpoints' membership in P_1^* and P_2^*:

- **Up-Path**: $u_i \in \mathrm{V}(P_2^*)$, $u_{i+1} \in \mathrm{V}(P_1^*)$;
- **Down-Path**: $u_i \in \mathrm{V}(P_1^*)$, $u_{i+1} \in \mathrm{V}(P_2^*)$;
- **Gap-Path**: both $u_i, u_{i+1} \in \mathrm{V}(P_1^*)$ or both in $\mathrm{V}(P_2^*)$.

For each case, we define the corresponding internal vertices (excluding shared intersection points) $V(P_{u_i \to u_{i+1}}) \setminus V^{\text{inter}}(P_1^*, P_2^*; P)$ as **Up-Vertices**, **Down-Vertices**, or **Gap-Vertices**, respectively.

We denote their union over all such subpaths in P by

$$V^{\text{up}}(P_1^*, P_2^*; P), \quad V^{\text{down}}(P_1^*, P_2^*; P), \quad \text{and } V^{\text{gap}}(P_1^*, P_2^*; P),$$

respectively. For a PDFP (P_1, P_2), we extend the notation by defining:

$$V^{\text{down}}(P_1^*, P_2^*; P_1, P_2) := V^{\text{down}}(P_1^*, P_2^*; P_1) \cup V^{\text{down}}(P_1^*, P_2^*; P_2),$$

and analogously for V^{up} and V^{gap}.

Remark. The notations $\mathcal{I}(P_0^*; P_1, P_2)$ and $V^{\text{inter}}(P_1^*, P_2^*; P)$ are similar but differ slightly. The former is defined in the context of an oracle middle path P_0^*, while the latter arises in the comparison between two PDFPs. Notably, $V^{\text{inter}}(P_1^*, P_2^*; P)$ includes the endpoints of P. As a result, we have the following decomposition:

$$V(P) = V^{\text{up}}(P_1^*, P_2^*; P) \uplus V^{\text{down}}(P_1^*, P_2^*; P) \uplus V^{\text{gap}}(P_1^*, P_2^*; P) \uplus V^{\text{inter}}(P_1^*, P_2^*; P).$$

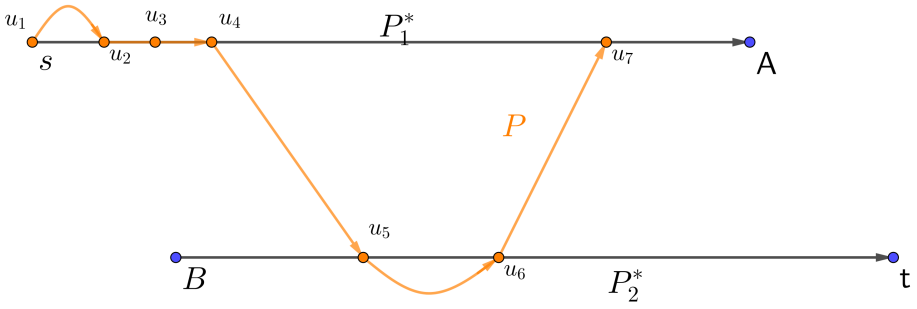


Figure 4: An example of Definition 6.1. P is a path from u_1 to u_7, and $V^{\text{inter}}(P_1^*, P_2^*; P) = \{u_1, u_2, \ldots, u_7\}$. $P_{u_1 \to u_2}, P_{u_2 \to u_3}, P_{u_3 \to u_4}$ and $P_{u_5 \to u_6}$ are gap-paths. However, $P_{u_2 \to u_3}$ and $P_{u_3 \to u_4}$ contain no gap-vertices. $P_{u_4 \to u_5}$ is a down-path. $P_{u_6 \to u_7}$ is an up-path.

Lemma 6.1. *Given an oracle middle path P_0^* from A to B, for any P_0^*-compatible $(s \to A, B \to t)$-PDFP (P_1^*, P_2^*) and any $(s \to A, B \to t)$-PDFP (P_1, P_2), (P_1, P_2) intersects P_0^* only in up-vertices. i.e.,*

$$\mathcal{I}(P_0^*; P_1, P_2) \subseteq V^{\text{up}}(P_1^*, P_2^*; P_1, P_2).$$

Proof. Since $\mathcal{I}(P_0^*; P_1^*, P_2^*) = \emptyset$, for any $u \in \mathcal{I}(P_0^*; P_1, P_2)$, u can be an up-vertex, a down-vertex, or a gap-vertex. We argue by contradiction. Suppose, for the sake of contradiction, that u is either a down-vertex or a gap-vertex. By Definition 6.1, this implies that u lies on either a down-path or a gap-path. Denote p as this down-path or gap-path, and $u_L, u_R \in \mathrm{V}(p)$ are two endpoints of p. We discuss two cases based on whether $u_R \in \mathrm{V}(P_1^*)$. See Figure 5.

Case 1: $u_R \in \mathrm{V}(P_1^*)$**.**
Without loss of generality, we assume that u is the first vertex on p (in the direction from u_L to u_R) that is in P_0^*. That is, the internal vertices of the subpath $p_{u_L \to u}$ are disjoint from P_0^*; formally,

$$(\mathrm{V}(p_{u_L \to u}) \setminus \{u_L, u\}) \cap \mathrm{V}(P_0^*) = \emptyset.$$

In this case, u must be a gap-vertex and $u_L \in \mathrm{V}(P_1^*)$. Therefore, we construct the following walk:

$$W_1 = (P_1^*)_{s \to u_L} \circ (p)_{u_L \to u} \circ (P_0^*)_{u \to B} \circ (P_2^*).$$

- By Definition 6.1, we know that the subpath $(p)_{u_L \to u}$ does not intersect P_1^* or P_2^*, except possibly at its endpoints. Moreover, since P_0^*, P_1^*, and P_2^* together form a path, they do not intersect one another except at their endpoints. Therefore, this walk contains no repeated vertices.
- By Lemma 5.2, we know $d(u) > d(B)$, so $(P_0^*)_{u \to B}$ contains a back-edge. By Lemma 3.1, $w(W_1) > d_G(s,t)$.
- Recall that $d(u) < d(A)$ by Lemma 5.2. Since $w((p)_{u_L \to u}) = d(u) - d(u_L)$ is strictly smaller than $w((P_1^*)_{u_L \to A})$, we have $w(W_1) = d(u) + w\left((P_0^*)_{u \to B} \circ (P_2^*)\right) < w(P_1^* \circ P_0^* \circ P_2^*)$.

Therefore, W_1 is a shorter $s \to t$ not-shortest path, which contradicts the assumption that $P_1^* \circ P_0^* \circ P_2^*$ is a next-to-shortest path.

Case 2: $u_R \in \mathrm{V}(P_2^*)$**.**
Without loss of generality, we let u be the last vertex on p (in the direction from u_L to u_R) that intersects P_0^*. We consider the following walk:

$$W_2 = P_1^* \circ (P_0^*)_{A \to u} \circ p_{u \to u_R} \circ (P_2^*)_{u_R \to t}$$

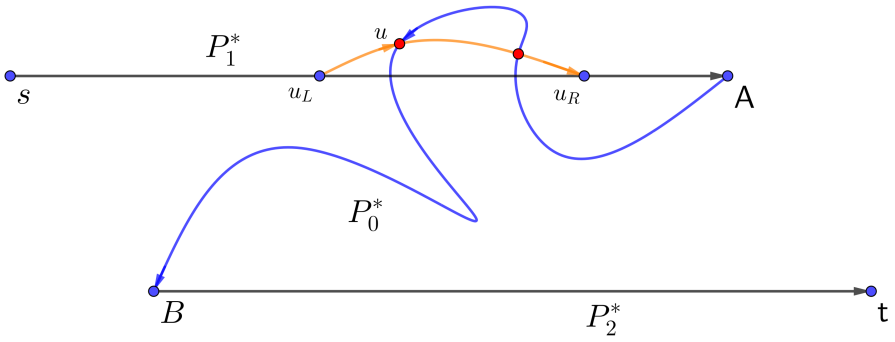
- By our assumption that u is the last vertex on p, $p_{u \to u_R}$ does not intersect P_0^* (except for u). Also, by Definition 6.1, $p_{u \to u_R}$ does not intersect P_0^*, P_1^* or P_2^* except at its endpoints, so W_2 contains no repeated vertices.
- By Lemma 5.2, we know $d(u) < d(A)$, so $(P_0^*)_{A \to u}$ contains a back-edge and $w(W_2) > d_G(s,t)$.
- Similar to case 1, by Lemma 5.2 we have $d(u) > d(B)$. Therefore, $w(W_2) = w(P_1^*) + (w((P_0^*)_{A \to u}) + w(p_{u \to u_R})) + w((P_2^*)_{u_R \to t}) < w(P_1^*) + w(P_0^*) + w(P_2^*)$.

Therefore, W_2 is a shorter $s \to t$ not-shortest path, which contradicts the assumption that $P_1^* \circ P_0^* \circ P_2^*$ is a next-to-shortest path. □

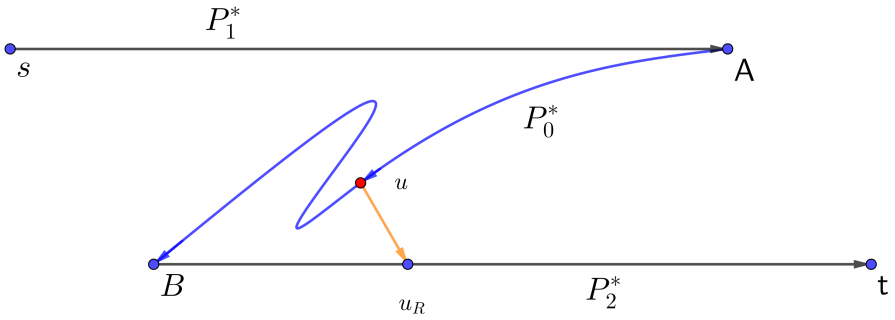
6.2 Construction of the Isolated Tuple

Recall that we are discussing a (s,t)-layered graph $G = (V, E, w)$. In this subsection, for an oracle middle path P_0^*, we provide a construction of the two edges $(X', X), (Y', Y) \in E$, and show some properties of this construction.

Definition 6.2. Given an oracle middle path P_0^* from A to B, we call a pair $(X, Y) \in V \times V$ **critical** if it satisfies the following conditions:



(a) Case 1.



(b) Case 2.

Figure 5: An illustration of proof for Lemma 6.1.

(i) $d(B) \leq d(X) = d(Y) \leq d(A)$

(ii) There exists a P_0^*-compatible $(s{\to}X{\to}A, B{\to}Y{\to}t)$-PDFP.

(iii) There exists a $(s \to X, B \to Y)$-PDFP (P_1', P_2') such that P_1' or P_2' intersects with P_0^* (i.e. $\mathcal{I}(P_0^*; P_1', P_2') \neq \emptyset$).

Lemma 6.2. *Let $G = (V, E, w)$ be a (s,t)-layered graph, and P_0^* be an oracle middle path from A to B. Assume that there is no critical pair. Then for any P_0^*-compatible $(s{\to}A, B{\to}t)$-PDFP (P_1^*, P_2^*), and for any tuple (X', X, Y', Y) such that $(X', X) \in \mathrm{E}(P_1^*), (Y', Y) \in \mathrm{E}(P_2^*)$ and $\lambda(X') = \lambda(Y')$, the 4-tuple (X', X, Y', Y) is P_0^*-isolated .*

Proof. By definition, we have:

- $\lambda(X') = \lambda(Y') = \lambda(X) - 1 = \lambda(Y) - 1$ since P_1^* and P_2^* are forward paths.
- (P_1^*, P_2^*) is a P_0^*-compatible $(s{\to}X'{\to}X{\to}A, B{\to}Y'{\to}Y{\to}t)$-PDFP.
- Consider any $(s{\to}X'{\to}X{\to}A, B{\to}Y'{\to}Y{\to}t)$-PDFP (P_1, P_2). Let A' be the vertex in P_2^* with $d(A') = d(A)$. If $\mathcal{I}(P_0^*; P_1, P_2) \neq \emptyset$, then (A, A') is a critical pair, which contradicts our assumption. Therefore, $\mathcal{I}(P_0^*; P_1, P_2) = \emptyset$.

Hence, (X', X, Y', Y) is P_0^*-isolated. □

Now we provide our construction of (X', X, Y', Y):

- If there are no critical pairs, then by the definition of oracle middle path, there exists a P_0^*-compatible $(s \to A, B \to t)$-PDFP (P_1^*, P_2^*). Thus by Lemma 6.2, the construction of (X', X, Y', Y) is trivial as we can choose any (X', X) and (Y', Y) in P_1^* and P_2^* respectively with the same layer index.
- Otherwise, there exist at least one critical pairs. In this case, we choose this 4-tuple as follows:
  - (X, Y) is an arbitrary leftmost critical pair. i.e., $d(X)$ is minimum among all such pairs.
  - By the definition of critical pair, we know that there exists a P_0^*-compatible $(s \to X \to A, B \to Y \to t)$-PDFP (P_1^*, P_2^*). Let X' be the predecessor of X on P_1^* and Y' be the predecessor of Y on P_2^*, then it is clear that $(X', X), (Y', Y) \in E$ and $\lambda(X') = \lambda(Y') = \lambda(X) - 1 = \lambda(Y) - 1$.

Lemma 6.3. *We are given a (s, t)-layered graph $G = (V, E, w)$ and an oracle middle path P_0^* from A to B. Let (X', X, Y', Y) be as constructed above. Assume that there exists an $(s \to X' \to X \to A, B \to Y' \to Y \to t)$-PDFP (P_1, P_2) such that $\mathcal{I}(P_0^*; P_1, P_2) \neq \emptyset$. Then there exists an $(s \to X \to A, B \to Y \to t)$-PDFP (P_1', P_2') that has at least two intersection vertices $\alpha, \beta \in \mathcal{I}(P_0^*; P_1', P_2')$ with $d(\alpha) < d(X) = d(Y) < d(\beta)$.*

Proof. If there is no critical pair, the statement is trivial: by Lemma 6.2, there exists no $(s \to X' \to X \to A, B \to Y' \to Y \to t)$-PDFP that intersects P_0^*.

In the remainder of the proof, we may therefore assume that a critical pair exists, and (X, Y) is a leftmost one. Let $u \in \mathcal{I}(P_0^*; P_1, P_2)$ be an arbitrary intersection vertex. We discuss the following three cases:

- **Case 1:** $\lambda(u) < \lambda(X)$.
  In this case, (X', Y') is a critical pair and $\lambda(X') < \lambda(X)$. This contradicts the assumption that (X, Y) is leftmost.
- **Case 2:** $\lambda(u) = \lambda(X)$.
  It is impossible because (P_1, P_2) only goes through (X, Y) in layer $\lambda(X)$.
- **Case 3:** $\lambda(u) > \lambda(X)$.
  Since (X, Y) is a critical pair (Definition 6.2), there exists a $(s \to X, B \to Y)$-PDFP (P_1'', P_2'') that intersects P_0^*. In other words, there exists some $\alpha \in \mathcal{I}(P_0^*; P_1'', P_2'')$ such that $d(\alpha) < d(X)$ (equivalently, $\lambda(\alpha) < \lambda(X)$).

  We consider an $(s \to X \to A, B \to Y \to t)$-PDFP $(P_1', P_2') = (P_1'' \circ (P_1)_{X \to A}, P_2'' \circ (P_2)_{Y \to t})$. It contains at least two intersection vertices α and $\beta = u$, where $\lambda(\alpha) < \lambda(X)$ and $\lambda(\beta) > \lambda(Y)$. This completes the proof.

□

6.3 Final Analysis

In this subsection, we will complete the proof of the Key Lemma (Lemma 5.3). We need to show that the choice of (X', X, Y', Y) in Section 6.2 is P_0^*-isolated. According to Lemma 6.3, we only need to consider the case that there exists an $(s \to X \to A, B \to Y \to t)$-PDFP with at least two intersection vertices with P_0^*. To prove this, we focus on the structural relationship between an arbitrary PDFP and a P_0^*-compatible PDFP.

Lemma 6.4. *Given a (s,t)-layered graph $G = (V, E, w)$, an oracle middle path P_0^*, and an arbitrary corresponding critical pair (X, Y), for any $(s{\to}X{\to}A, B{\to}Y{\to}t)$-PDFP (P_1, P_2), there exists a P_0^*-compatible $(s{\to}X{\to}A, B{\to}Y{\to}t)$-PDFP (P_1^{**}, P_2^{**}) such that:*

$$V^{\text{gap}}(P_1^{**}, P_2^{**}; P_1, P_2) = \emptyset$$

Proof. Assume that we have an $(s{\to}X{\to}A, B{\to}Y{\to}t)$-PDFP (P_1, P_2). We choose (P_1^{**}, P_2^{**}) as an arbitrary P_0^*-compatible $(s{\to}X{\to}A, B{\to}Y{\to}t)$-PDFP that maximizes:

$$|\,(\mathrm{E}(P_1^{**}) \cup \mathrm{E}(P_2^{**})) \cap (\mathrm{E}(P_1) \cup \mathrm{E}(P_2))\,|$$

i.e., the number of common edges.

We prove the statement by contradiction. Suppose that $V^{\text{gap}}(P_1^{**}, P_2^{**}; P_1, P_2) \neq \emptyset$, i.e., there exists a vertex $u \in V^{\text{gap}}(P_1^{**}, P_2^{**}; P_1, P_2)$. We know u is in P_1 or P_2. Denote this path as P_i for a specific $i \in \{1, 2\}$, and let $u_L, u_R \in V$ be the two endpoints of this gap-path.

Let $u_L, u_R \in \mathrm{V}(P_j^{**})$ for a specific $j \in \{1, 2\}$. We can replace $(P_j^{**})_{u_L \to u_R}$ by $(P_i)_{u_L \to u_R}$.

- By Lemma 6.1, $(P_i)_{u_L \to u_R}$ does not intersect P_0^*. i.e., $\mathrm{V}((P_i)_{u_L \to u_R}) \cap (\mathrm{V}(P_0^*) \setminus \{A, B\}) = \emptyset$. In addition, according to the definition of gap-path, $(P_i)_{u_L \to u_R}$ does not intersect P_{3-j}^{**}, and neither X nor Y is an internal vertex of $(P_i)_{u_L \to u_R}$. Therefore, after replacing $(P_j^{**})_{u_L \to u_R}$ by $(P_i)_{u_L \to u_R}$, the paths P_1^{**}, P_2^{**} are P_0^*-compatible $(s{\to}X{\to}A, B{\to}Y{\to}t)$-PDFP.
- The replacement provides more common edges:

$$\begin{aligned} |\,\mathrm{E}((P_i)_{u_L \to u_R}) \cap (\mathrm{E}(P_1) \cup \mathrm{E}(P_2))\,| &= \lambda(u_R) - \lambda(u_L) \\ &> |\,\mathrm{E}((P_j^{**})_{u_L \to u_R}) \cap (\mathrm{E}(P_1) \cup \mathrm{E}(P_2))\,| \end{aligned}$$

This contradicts the assumption that P_1^{**}, P_2^{**}) are chosen to maximize the number of common edges with P_1, P_2. □

Based on a critical pair (X, Y) the vertex X, we divide the vertex set V into three parts:

- the left region: $\{u \in V : \lambda(u) < \lambda(X)\}$,
- the middle region: $\{u \in V : \lambda(u) = \lambda(X)\}$,
- the right region: $\{u \in V : \lambda(u) > \lambda(X)\}$.

Lemma 6.5. *Let $G = (V, E, w)$ be an (s,t)-layered graph, and let P_0^* be an oracle middle path with critical pair (X, Y). We partition E into the left region ($\{u \in \mathrm{V} : \lambda(u) < \lambda(X)\}$) and the right region ($\{u \in \mathrm{V} : \lambda(u) > \lambda(X)\}$). For any $(s{\to}X{\to}A, B{\to}Y{\to}t)$-PDFP (P_1, P_2) and P_0^*-compatible $(s{\to}X{\to}A, B{\to}Y{\to}t)$-PDFP (P_1^{**}, P_2^{**}) such that $V^{\text{gap}}(P_1^{**}, P_2^{**}; P_1, P_2) = \emptyset$, the following hold:*

(a) Every up-path p lies entirely in one region (i.e., either $\forall u \in \mathrm{V}(p), \lambda(u) < \lambda(X)$ or $\forall u \in \mathrm{V}(p), \lambda(u) > \lambda(X)$).

(b) If P_1' or P_2' contains an up-path in the right region, then P_1' contains a down-path $u_{1L} \to u_{1R}$ in the right region and P_2' contains an up-path $u_{2L} \to u_{2R}$ in the right region. Moreover, $\lambda(u_{1R}) > \lambda(u_{2L})$ and $\lambda(u_{2R}) > \lambda(u_{1L})$.

Proof.
Proof of (a). Since P_1, P_1^{**} are forward paths and both of them go through X, any up-paths in P_1 must be included in either the left region or the right region. Similarly, since P_2, P_2^{**} are forward paths, and both of them go through Y, up-paths in P_2 are included in either the left region or the right region.

Proof of (b). As P_1' or P_2' contains an up-path in the right region, we have $(P_1')_{X\to A} \neq (P_1^{**})_{X\to A}$ and $(P_2')_{Y\to t} \neq (P_2^{**})_{Y\to t}$ (because otherwise P_1' and P_2' would intersect). Notice that there are no gap paths in P_1' and P_2'. Therefore, in the right region, P_1' contains a down-path, and P_2' contains an up-path.

Let the first down path in P_1' in the right region be from u_{1L} to u_{1R}, and the first up-path of P_2' in the right region be from u_{2L} to u_{2R}. Here "first path" means the one with the leftmost starting vertex. Since P_1' and P_2' are forward paths, this is an intuitive definition. As $u_{1R} \notin P_2', u_{1R} \in P_2^{**}$ and $(P_2')_{Y\to u_{2L}} \subseteq P_2^{**}$, we must have $\lambda(u_{1R}) > \lambda(u_{2L})$ to avoid intersection between P_1' and P_2'. Similarly, we have $\lambda(u_{2R}) > \lambda(u_{1L})$. □

Proof of Lemma 5.3. We prove the Key Lemma by contradiction.

Recall that we are given a (s,t)-layered $G = (V, E, w)$ and an oracle middle path P_0^*. We construct the 4-tuple (X', X, Y', Y) as described in Section 6.2. That is, by Lemma 6.2, we may assume that critical pairs exist, and we fix a leftmost critical pair (X, Y). Then we select a P_0^*-compatible $(s\to X\to A, B\to Y\to t)$-PDFP (P_1^*, P_2^*), and choose edges $(X', X) \in \mathrm{E}(P_1^*)$ and $(Y', Y) \in \mathrm{E}(P_2^*)$.

Suppose for contradiction that there exists an $(s\to X'\to X\to A, B\to Y'\to Y\to t)$-PDFP (P_1, P_2) that intersects P_0^*. Then by Lemma 6.3, there exists an $(s\to X\to A, B\to Y\to t)$-PDFP (P_1', P_2') with two intersection vertices $\alpha, \beta \in \mathcal{I}(P_0^*; P_1', P_2')$ satisfying $\lambda(\alpha) < \lambda(X)$ and $\lambda(\beta) > \lambda(X)$. Applying Lemma 6.4 to (P_1', P_2'), we obtain a P_0^*-compatible $(s\to X\to A, B\to Y\to t)$-PDFP (P_1^{**}, P_2^{**}) such that $V^{\mathrm{gap}}(P_1^{**}, P_2^{**}; P_1', P_2') = \emptyset$.

Let u_{first} be the first vertex in $\mathcal{I}(P_0^*; P_1', P_2')$ with respect to P_0^* (in the direction from A to B). Based on the layer index of u_{first}, we distinguish the following two cases (we know $d(u_{\mathrm{first}}) \neq d(X)$ since u_{first} is either in P_1' or P_2'). An illustration of the paths and vertices introduced in each of the two cases is shown in Figure 6 and Figure 7.

Case 1: $d(u_{\mathbf{first}}) > d(X)$.
In this case, there must exist two consecutive (i.e., there are no intersection vertices in between) intersection vertices in P_0^* such that the first one is in the right region and the second one is in the left region. Let β be the first one, and α be the second one. Then we have: $(\mathrm{V}((P_0^*)_{\beta\to\alpha}) \setminus \{\alpha, \beta\}) \cap (\mathrm{V}(P_1') \cup \mathrm{V}(P_2')) = \emptyset$. By Lemma 6.1, we know both α and β are in up-paths, so we introduce the following notation:

- Let α be in the up-path $P^{(\alpha)}$, and the two endpoints of this up-path are $u_{\alpha L}$ and $u_{\alpha R}$.

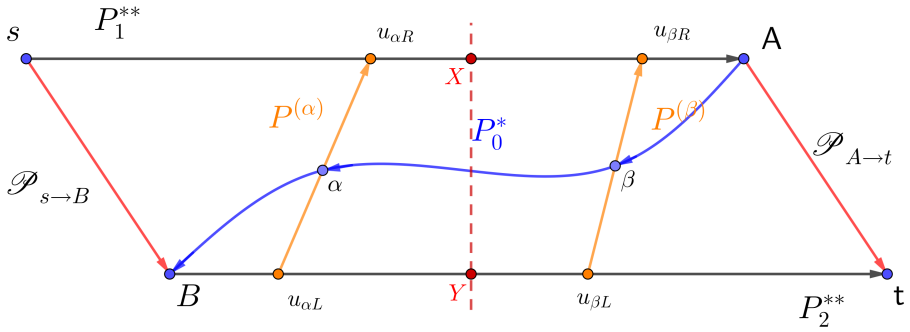


Figure 6: An illustration of Case 1 in the proof of Lemma 5.3.

- Let β be in the up-path $P^{(\beta)}$, and the two endpoints of this up-path are $u_{\beta L}$ and $u_{\beta R}$.

Now we consider the following walk:

$$W_2^* = \mathscr{P}_{s\to B} \circ (P_2^{**})_{B\to u_{\beta L}} \circ (P^{(\beta)})_{u_{\beta L}\to\beta} \circ (P_0^*)_{\beta\to\alpha} \circ (P^{(\alpha)})_{\alpha\to u_{\alpha R}} \circ (P_1^{**})_{u_{\alpha R}\to A} \circ \mathscr{P}_{A\to t}$$

To obtain a contradiction, we will show that W_2^* is an $s\to t$ not-shortest path that is shorter than $P_1^{**} \circ P_0^* \circ P_2^{**}$.

- (No Repeated Vertices) We can notice the followings:
  - By Lemma 5.2, $\mathscr{P}_{s\to B}$ and $\mathscr{P}_{A\to t}$ are disjoint and they do not intersect other fragments.
  - β, α are consecutive intersection vertices, so $(P_0^*)_{\beta\to\alpha}$ does not intersect the fragments that are subpaths of $P^{(\alpha)}, P^{(\beta)}$ except the endpoints $\{\alpha, \beta\}$. In addition, $(P_0^*)_{\beta\to\alpha}$ does not intersect the fragments that are subpaths of P_1^{**}, P_2^{**} because those are compatible with P_0^*. Hence, $(P_0^*)_{\beta\to\alpha}$ does not intersect the rest of W_2^*.
  - $P^{(\alpha)}$ and $P^{(\beta)}$ do not intersect each other because they are different parts of the same PDSP. Also, their internal vertices do not intersect P_1^{**} or P_2^{**} since they are up-paths.
- (Not Shortest Path) $d(\beta) > d(X) > d(\alpha)$, so W_2^* includes a back-edge.
- (Shorter than $P_1^{**} \circ P_0^* \circ P_2^{**}$) We can notice that $d(\alpha) > d(B)$ and $d(\beta) < d(A)$ (Lemma 5.2). Therefore, since W_2^* consists of a forward path from s to β, a subpath of P_0^*, and a forward path from α to t,

$$\begin{aligned} w(W_2^*) &\le d(\beta) + w(P_0^*) + d(t) - d(\alpha) \\ &< d(A) + w(P_0^*) + d(t) - d(B) = w(P_1^{**} \circ P_0^* \circ P_2^{**}) \end{aligned}$$

Case 2: $d(u_{\text{first}}) < d(X)$.
In this case, we cannot apply the same reasoning as in Case 1. Specifically, if P_0^* traverses the layers in a "squiggly" manner such that all intersection points in the left region appear before those in the right region along P_0^*, then the fragment $(P_0^*)_{\beta\to\alpha}$ no longer exists.

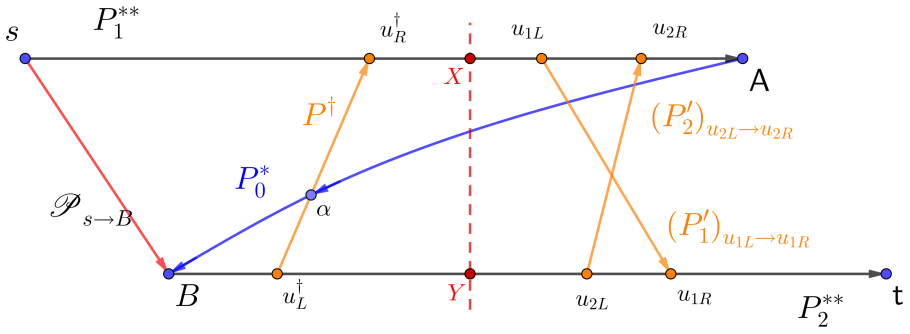


Figure 7: An illustration of Case 2 in the proof of Lemma 5.3.

Here we let $\alpha = u_{\text{first}}$, and let β be an arbitrary intersection vertex in the right region (i.e., $d(\beta) > d(X)$). By Lemma 6.1, β must be a up-vertex (i.e., $\beta \in V^{\text{up}}(P_1^{**}, P_2^{**}; P_1', P_2')$). Therefore, by Lemma 6.5, there exists an up-path that entirely lies in the right region.

By Lemma 6.5, there exist a down-path of P_1' in the right region from u_{1L} to u_{1R}, and an up-path of P_2' in the right region from u_{2L} to u_{2R}. Similarly, according to Lemma 6.1, we know that α is in an up-path. We denote this up-path as $P^\dagger$, and let it be from $u_L^\dagger$ to $u_R^\dagger$. Recall that $\mathscr{P}_{s\to B}$ is an arbitrary but fixed forward path from s to B. Then we consider the following walk:

$$\begin{aligned} W_1^* = \mathscr{P}_{s\to B} \circ (P_2^{**})_{B\to u_{2L}} \circ (P_2')_{u_{2L}\to u_{2R}} \circ (P_1^{**})_{u_{2R}\to A} \circ (P_0^*)_{A\to\alpha} \\ \circ (P^\dagger)_{\alpha\to u_R^\dagger} \circ (P_1^{**})_{u_R^\dagger\to u_{1L}} \circ (P_1')_{u_{1L}\to u_{1R}} \circ (P_2^{**})_{u_{1R}\to t} \end{aligned}$$

and for a concatenation $W = W_1 \circ \ldots \circ W_k$, we call $W_1, \ldots, W_k$ **fragments** of W. To obtain a contradiction, we will show that W_1^* is an $s\to t$ not-shortest path that is shorter than $P_1^{**} \circ P_0^* \circ P_2^{**}$.

- (No Repeated Vertices) According to Lemma 5.2, $\mathscr{P}_{s\to B}$ doesn't intersect the other fragments of W_1^* because all vertices in other fragments are of layer index not smaller than B. In addition, since $\alpha = u_{\text{first}}$ is the first intersection vertex, $(P_0^*)_{A\to\alpha}$ does not intersect the remaining fragments. The fragments that are subpaths of P_1^{**} and P_2^{**} are disjoint from each other by the fact that u_{1L} comes before u_{2R} on P_1^{**}, and that u_{2L} comes before u_{1R} on P_2^{**}. These fragments are also disjoint from the internal vertices of the fragments that are subpaths of P_1', P_2', since the latter are up-paths and down-paths. So, it remains to show that subpaths of P_1', P_2' are disjoint from each other. $(P^\dagger)_{\alpha\to u_R^\dagger}$ is disjoint from $(P_1')_{u_{1L}\to u_{1R}}$ and $(P_2')_{u_{2L}\to u_{2R}}$ since the former consists only of vertices in the left region, and the latter consists only of vertices in the right region. Lastly, $(P_1')_{u_{1L}\to u_{1R}}$ and $(P_2')_{u_{2L}\to u_{2R}}$ are disjoint since P_1', P_2' are disjoint by definition.
- (Not Shortest Path) By Lemma 5.2, $d(u_{\text{first}}) < d(A)$, so W_1^* includes a back-edge.
- (Shorter than $P_1^{**} \circ P_0^* \circ P_2^{**}$) We can notice that $d(\alpha) > d(u_L^\dagger) \geq d(B)$. Since W_1^* consists of a forward path from s to A, a subpath of P_0^*, and a forward path from u_{first} to t, we have:

$$\begin{aligned} w(W_1^*) &\leq d(A) + w(P_0^*) + d(t) - d(u_{\text{first}}) \\ &< d(A) + w(P_0^*) + d(t) - d(B) = w(P_1^{**} \circ P_0^* \circ P_2^{**}). \end{aligned}$$

To summarize, in both **Case 1** and **Case 2**, we derived a contradiction by constructing a not-shortest path that is shorter than $P_1^{**} \circ P_0^* \circ P_2^{**}$, contradicting the assumption that P_0^* is an oracle middle path. This completes the proof of Lemma 5.3 by showing that (X', X, Y', Y), as constructed in Section 6.2, is P_0^*-isolated. □

Acknowledgments

We would like to thank Luke Schaeffer and Fabrizio Grandoni for making us aware of the problem. We would like to thank the Bertinoro 2019 Fine-Grained Approximation Algorithms & Complexity Workshop for initial discussions about the problem, including discussions with Andreas Björklund, Thore Husfeldt, Vijaya Ramachandran, and Virginia Vassilevska Williams. We are also very grateful to Shyan Akmal, Jacob Holm, Eva Rotenberg, and Virginia Vassilevska Williams for extended discussions about the problem in 2020.

More recently, we would like to thank the participants of the open problem sessions of Nicole Wein's *Graph Algorithms* course for additional discussions about the problem. In particular, we would like to thank participants Zixi Cai, Shengquan Du, Bing Han, Teo Miklethun, Benyu Wang, and Qingyue Wu. We also thank Erik Demaine for use of his "supercollaboration" method and accompanying *Coauthor* software, for the open problem sessions. Finally, we would like to thank Yang Hu and Junkai Zhang for additional discussions.

References

[AHW25] Sepehr Assadi, Gary Hoppenworth, and Nicole Wein. Covering approximate shortest paths with dags. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC)*, pages 2269–2280, 2025.

[AVWWX23] Shyan Akmal, Virginia Vassilevska Williams, Ryan Williams, and Zixuan Xu. Faster detours in undirected graphs. In *31st Annual European Symposium on Algorithms (ESA)*, volume 274, page 7. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023.

[AW23] Shyan Akmal and Nicole Wein. A local-to-global theorem for congested shortest paths. In *31st Annual European Symposium on Algorithms (ESA)*, pages 8–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023.

[BCDF19] Ivona Bezáková, Radu Curticapean, Holger Dell, and Fedor V Fomin. Finding detours is fixed-parameter tractable. *SIAM Journal on Discrete Mathematics*, 33(4):2326–2345, 2019.

[BMP09] Sambhu Charan Barman, Sukumar Mondal, and Madhumangal Pal. An efficient algorithm to find next-to-shortest path on permutation graphs. *Journal of Applied Mathematics and Computing*, 31(1):369–384, 2009.

[BSS21] Eli Berger, Paul Seymour, and Sophie Spirkl. Finding an induced path that is not a shortest path. *Discrete Mathematics*, 344(7):112398, 2021.

[Chi23] Rajesh Chitnis. A tight lower bound for edge-disjoint paths on planar dags. *SIAM Journal on Discrete Mathematics*, 37(2):556–572, 2023.

[CMPP13] Marek Cygan, Daniel Marx, Marcin Pilipczuk, and Michal Pilipczuk. The planar directed k-vertex-disjoint paths problem is fixed-parameter tractable. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 197–206, 2013.

[COO23] Kyungjin Cho, Eunjin Oh, and Seunghyeok Oh. Parameterized algorithm for the disjoint path problem on planar graphs: exponential in k 2 and linear in n. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3734–3758. SIAM, 2023.

[DHLS14] Erik D Demaine, Yamming Huang, Chung-Shou Liao, and Kunihiko Sadakane. Canadians should travel randomly. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 380–391. Springer, 2014.

[DKQ18] Alexandre Dolgui, Mikhail Y Kovalyov, and Alain Quilliot. Simple paths with exact and forbidden lengths. *Naval Research Logistics (NRL)*, 65(1):78–85, 2018.

[FGL+23] Fedor V Fomin, Petr A Golovach, William Lochet, Danil Sagunov, Saket Saurabh, and Kirill Simonov. Detours in directed graphs. *Journal of Computer and System Sciences*, 137:66–86, 2023.

[FGLZ07] Rudolf Fleischer, Qi Ge, Jian Li, and Hong Zhu. Efficient algorithms for k-disjoint paths problems on dags. In *International Conference on Algorithmic Applications in Management*, pages 134–143. Springer, 2007.

[FHW80] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

[HMPS23] Meike Hatzel, Konrad Majewski, Michał Pilipczuk, and Marek Sokołowski. Simpler and faster algorithms for detours in planar digraphs. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 156–165. SIAM, 2023.

[KCWJ11] Kuo-Hua Kao, Jou-Ming Chang, Yue-Li Wang, and Justie Su-Tzu Juan. A quadratic algorithm for finding next-to-shortest paths in graphs. *Algorithmica*, 61(2):402–418, 2011.

[KKR12] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012.

[KN04] Ilia Krasikov and Steven D Noble. Finding next-to-shortest paths in a graph. *Information processing letters*, 92(3):117–119, 2004.

[LMP+20] Daniel Lokshtanov, Pranabendu Misra, Michał Pilipczuk, Saket Saurabh, and Meirav Zehavi. An exponential time parameterized algorithm for planar disjoint paths. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1307–1316, 2020.

[LP97] Kumar N Lalgudi and Marios C Papaefthymiou. Computing strictly-second shortest paths. *Information processing letters*, 63(4):177–181, 1997.

[LPP96] Kumar N Lalgudi, Marios C Papaefthymiou, and Miodrag Potkonjak. Optimizing systems for effective block-processing: The k-delay problem. In *Proceedings of the 33rd annual Design Automation Conference*, pages 714–719, 1996.

[LPP00] Kumar N Lalgudi, Marios C Papaefthymiou, and Miodrag Potkonjak. Optimizing computations for effective block-processing. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 5(3):604–630, 2000.

[LSC06] Shisheng Li, Guangzhong Sun, and Guoliang Chen. Improved algorithm for finding next-to-shortest paths. *Information processing letters*, 99(5):192–194, 2006.

[Lyn75] James F Lynch. The equivalence of theorem proving and the interconnection problem. *ACM Sigda Newsletter*, 5(3):31–36, 1975.

[MP93] Matthias Middendorf and Frank Pfeiffer. On the complexity of the disjoint paths problem. *Combinatorica*, 13(1):97–107, 1993.

[MP06] S Mondal and M Pal. A sequential algorithm to solve next-to-shortest path problem on circular-arc graphs. *Journal of Physical Sciences*, 10:201–217, 2006.

[NU02] Matti Nykänen and Esko Ukkonen. The exact path length problem. *Journal of Algorithms*, 42(1):41–53, 2002.

[RS95] Neil Robertson and Paul D Seymour. Graph minors. xiii. the disjoint paths problem. *Journal of combinatorial theory, Series B*, 63(1):65–110, 1995.

[Sch94] Alexander Schrijver. Finding k disjoint paths in a directed planar graph. *SIAM Journal on Computing*, 23(4):780–788, 1994.

[Sli10] Aleksandrs Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. *SIAM Journal on Discrete Mathematics*, 24(1):146–157, 2010.

[SSMT16] Leena Salmela, Kristoffer Sahlin, Veli Mäkinen, and Alexandru I Tomescu. Gap filling as exact path length problem. *Journal of Computational Biology*, 23(5):347–361, 2016.

[Tho12] Torsten Tholey. Linear time algorithms for two disjoint paths problems on directed acyclic graphs. *Theoretical Computer Science*, 465:35–48, 2012.

[WGW12] Bang Ye Wu, Jun-Lin Guo, and Yue-Li Wang. A linear time algorithm for the next-to-shortest path problem on undirected graphs with nonnegative edge lengths. *arXiv preprint arXiv:1203.5235*, 2012.

[Wu13] Bang Ye Wu. A simpler and more efficient algorithm for the next-to-shortest path problem. *Algorithmica*, 65(2):467–479, 2013.

[WW15] Bang Ye Wu and Hung-Lung Wang. The next-to-shortest path problem on directed graphs with positive edge weights. *Networks*, 65(3):205–211, 2015.

[WZ23] Michał Włodarczyk and Meirav Zehavi. Planar disjoint paths, treewidth, and kernels. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 649–662, 2023.

[ZN12] Cong Zhang and Hiroshi Nagamochi. The next-to-shortest path in undirected graphs with nonnegative weights. In *Proceedings of the Eighteenth Computing: The Australasian Theory Symposium-Volume 128*, pages 13–20, 2012.