Efficient Algorithms for the Disjoint Shortest Paths Problem and Its Extensions

Keerti Choudhary
Department of Computer Science and Engineering, IIT Delhi, India

Amit Kumar
Department of Computer Science and Engineering, IIT Delhi, India

Lakshay Saggi
Department of Computer Science and Engineering, IIT Delhi, India

Abstract

We study the *2-Disjoint Shortest Paths (2-DSP)* problem: given a directed weighted graph and two terminal pairs (s_1, t_1) and (s_2, t_2), decide whether there exist vertex-disjoint shortest paths between each pair.

Building on recent advances in disjoint shortest paths for DAGs and undirected graphs (Akmal et al. 2024), we present an $O(mn \log n)$-time algorithm for this problem in weighted directed graphs that do not contain negative or zero weight cycles. This algorithm presents a significant improvement over the previously known $O(m^5 n)$-time bound (Berczi et al. 2017). Our approach exploits the algebraic structure of polynomials that enumerate shortest paths between terminal pairs. A key insight is that these polynomials admit a recursive decomposition, enabling efficient evaluation via dynamic programming over fields of characteristic two. Furthermore, we demonstrate how to report the corresponding paths in $O(mn^2 \log n)$-time.

In addition, we extend our techniques to a more general setting: given two terminal pairs (s_1, t_1) and (s_2, t_2) in a directed graph, find the minimum possible number of vertex intersections between any shortest path from s_1 to t_1 and s_2 to t_2. We call this the *Minimum 2-Disjoint Shortest Paths (Min-2-DSP)* problem. We provide in this paper the first efficient algorithm for this problem, including an $O(m^2 n^3)$-time algorithm for directed graphs with positive edge weights, and an $O(m + n)$-time algorithm for DAGs and undirected graphs. Moreover, if the number of intersecting vertices is at least one, we show that it is possible to report the paths in the same $O(m+n)$-time. This is somewhat surprising, as there is no known $o(mn)$ time algorithm for explicitly reporting the paths if they are vertex-disjoint, and is left as an open problem in (Akmal et al. 2024).

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Disjoint paths, Disjoint shortest paths, Algebraic graph algorithms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2026.39

Related Version *Full Version*: `https://arxiv.org/abs/2509.14588` [11]

Funding *Lakshay Saggi*: Supported by Prime Minister's Research Fellowship (PMRF).

Acknowledgements The authors are thankful to Shyan Akmal for his detailed and valuable feedback.

1 Introduction

The *2-Disjoint Shortest Paths (2-DSP)* problem seeks to determine whether a graph contains two vertex-disjoint paths such that each path is a shortest path between its designated source and target vertices. More formally, given a weighted graph $G = (V, E)$ with edge weights $\ell : E \to \mathbb{R}$ and two source-target pairs (s_1, t_1) and (s_2, t_2), the goal is to determine whether there exist vertex-disjoint paths P_1 and P_2, where P_1 is a shortest (s_1, t_1)-path and P_2 is a shortest (s_2, t_2)-path.

The 2-DSP problem is a fundamental problem in combinatorial optimization with applications in network flows, routing, and circuit design [20, 22, 23]. Beyond its practical relevance, the problem has attracted significant attention because its complexity varies across graph classes, giving rise to a rich algorithmic landscape.

The 2-DSP problem has a rich history of algorithmic results. Eilam-Tzoreff [12] gave the first polynomial-time algorithm for 2-DSP in undirected graphs with running time $O(n^8)$. This was improved by Akhmedov [1] to $O(n^7)$ for the weighted case and $O(n^6)$ for the unweighted case. Bentert et al. [6] obtained an $O(mn)$ algorithm for unweighted undirected graphs. More recently, Akmal, Vassilevska Williams, and Wein [4] achieved an optimal $O(m+n)$ algorithm for weighted undirected graphs, and extended their techniques to obtain the same bound for weighted DAGs. Their algorithm employs an algebraic framework based on path-enumerating polynomials, where cancellations over a field of characteristic two are used to detect disjoint shortest paths.

The situation is markedly different for general directed graphs. Here, the first polynomial-time algorithm was given only recently by Bérczi and Kobayashi [7], whose method requires strictly positive edge weights and runs in $O(m^5 n)$ time.[1] Thus, despite decades of progress in the undirected setting, the directed case has remained far less understood, with a substantial gap between known upper bounds and what might be achievable.

Our work addresses the central question: **Can 2-DSP be solved efficiently in general directed weighted graphs?** We give an affirmative answer that significantly improves the running time of 2-DSP problem. Our work builds on the framework of [4] by employing algebraic techniques for suitable enumerating polynomials. However, the complexity of path intersections in general directed graphs poses new challenges and we require significantly new ideas to obtain a recursive decomposition of these polynomials.

We also introduce and study a natural relaxation of 2-DSP. When no pair of vertex-disjoint shortest paths exists, one may instead ask for two shortest paths whose overlap is as small as possible. This motivates the *Minimum 2-Disjoint Shortest Paths (Min-2-DSP)* problem: given two source–target pairs (x_1, y_1) and (x_2, y_2), find shortest paths P_1 and P_2 minimizing $|V(P_1) \cap V(P_2)|$, where $V(P)$ denotes the vertex set of path P. Our techniques extend naturally to this more general problem, yielding the first polynomial-time algorithms for Min-2-DSP across several graph classes. The problem is inherently more intricate than 2-DSP, since it requires reasoning not only about the existence of disjoint paths but also about quantifying and optimizing the extent of their intersections.

1.1 Our Contributions

Our first result presents a polynomial-time algorithm for 2-DSP in directed weighted graphs without negative or zero-weight cycles, providing the first efficient algorithm in this general setting. Further, it significantly improves the previously best-known $O(m^5 n)$-time bound even for the special case of directed graphs with strictly positive edge weights.

▶ **Theorem 1.1.** *The 2-DSP problem in directed weighted graphs having no cycles of negative or zero weight is solvable in $O(mn \log n)$ time. Moreover, we can report the explicit paths, if they exist, in $O(mn^2 \log n)$ time.*

[1] This running time is obtained by reducing to the edge disjoint version of the 2-DSP problem. However, we feel that a direct implementation of their approach to vertex-disjoint setting would also require cycling over subsets of size four from the set of vertices, and hence, would incur $\Omega(mn^4)$ time.

It is crucial to note that the graph considered in the theorem above must not contain zero-weight cycles. This is because if zero weight cycles are allowed, then the 2-Disjoint Paths (2-DP) problem which is NP-hard in directed graphs [13] becomes a special case of the 2-DSP problem.

We next consider the Min-2-DSP problem in directed graphs with positive edge weights. This result provides the first algorithmic framework for minimizing path intersections in the directed setting, showing that even this more general objective remains polynomial-time solvable.

▶ **Theorem 1.2.** *The Min-2-DSP problem in directed graphs with positive edge weights is solvable in $O(m^2n^3)$ time. Moreover, we can report the explicit paths in $O(m^2n^4)$ time bound.*

For structured graph classes, we obtain significantly faster algorithms. In particular, for DAGs we achieve an optimal linear-time bound.

▶ **Theorem 1.3.** *The Min-2-DSP problem in weighted DAGs is solvable in $O(m+n)$ time. Moreover, if we are guaranteed that no disjoint paths exist, then there exists an algorithm which also reports the explicit paths in the same time.*

Finally, we extend these results to undirected graphs with positive edge weights, again achieving optimal complexity.

▶ **Theorem 1.4.** *The Min-2-DSP problem in undirected graphs with positive edge weights is solvable in $O(m+n)$ time. Moreover, if we are guaranteed that no disjoint paths exist, then there exists an algorithm which also reports the explicit paths in the same time.*

1.2 Related Work

For $k \geqslant 3$, the k-DSP problem complexity landscape becomes dramatically more challenging, and the complexity landscape diverges sharply between undirected and directed graphs. In undirected graphs, Lochet [18] provided the first polynomial-time algorithm for k-DSP (for constant k), though with a running time of $n^{O(k^{5^k})}$, which was later improved to $n^{O(k \cdot k!)}$ by Bentert et al. [6]. It is worth noting that $n^{\Theta(k)}$ is the best we can hope for in undirected graphs, due to $W[1]$-Hardness and fine-grained lower bounds established in [6, 18, 4]. Hence, reducing the exponent from $\Theta(k!)$ to $\Theta(\text{poly}(k))$ remains an important open problem for k-DSP in undirected graphs. In an interesting breakthrough, Pilipczuk, Stamoulis and Wlodarczyk [21] recently claimed an FPT algorithm for k-DSP in undirected planar graphs, providing the first non-trivial graph class to admit such an algorithm.

In contrast, the situation is far less understood for directed graphs with $k \geqslant 3$: no polynomial-time algorithms are known, and no hardness results have been established either. The complexity of k-DSP in directed graphs therefore remains wide open. Nevertheless, $n^{O(k)}$-time algorithms are known for certain restricted classes, including DAGs [4, 7, 13] and planar digraphs [7].

A different line of work considers the objective of finding vertex-disjoint paths that minimize the *total length* of the paths. For undirected graphs, Björklund and Husfeldt [8] developed an algebraic algorithm for the case of two paths with running time $O(n^{11})$. This running time was subsequently improved to $\widetilde{O}(n^{3+\omega})$ by Björklund, Husfeldt, and Kaski [9]. For $k \geqslant 3$, the problem remains open even in undirected graphs, and since this objective generalizes k-DSP, it inherits all of its hardness barriers [6, 18]. More recently, Mari et al. [19] obtained an FPT algorithm for rectangular grid graphs for this objective.

Another natural objective extending the k-DSP problem is to maximize the number of source–target pairs that can be connected by vertex-disjoint shortest paths. Clearly this problem inherits the lower-bound barriers of k-DSP, making it natural to ask whether efficient approximation algorithms are possible. However, recent hardness results by Chitnis et al. [10] and Bentert et al. [5] rule out even such approximation approaches, underscoring the inherent difficulty of the problem.

To the best of our knowledge, the Min-2-DSP objective has not been studied previously. A related direction appears in the work of Funayama et al. [14], who consider the problem of finding k shortest s–t paths that minimize the total number of pairwise vertex intersections among them. We leave the Min-k-DSP as an open problem, for $k \geqslant 3$, for undirected as well as directed graphs.

1.3 Organization of the Paper

We introduce the necessary notations, definitions, and algebraic tools in Section 2. An overview of our techniques is provided in Section 3. Our $O(mn \log n)$-time algorithm for the 2-DSP problem in general directed graphs is presented in Section 4. The results on the Min-2-DSP problem for directed graphs are discussed in Section 5, while Min-2-DSP results for DAGs and undirected graphs are deferred to the full version [11]. Proofs and technical details omitted from the main exposition are also included in the full version [11].

2 Preliminaries

Let $G = (V, E)$ be a directed or undirected graph. For any vertex $v \in V$, let $\Gamma^{\text{IN}}(v)$ and $\mathcal{E}^{\text{IN}}(v)$ denote the set of in-neighbors and in-edges of v, respectively, and let $\text{indeg}(v)$ denote the size of these sets. Define $\Gamma^{\text{OUT}}(v)$, $\mathcal{E}^{\text{OUT}}(v)$, and $\text{outdeg}(v)$ analogously. For any two vertices $x, y \in V$, let $\text{dist}(x, y)$ denote the length of the shortest path from x to y in G, and let $\Pi(x, y)$ denote the set of all shortest paths from x to y. We say that a vertex w is a '*distance-critical vertex*' for the pair (x, y) if its removal increases the distance from x to y, i.e., $\text{dist}(x, y, G \setminus \{w\}) > \text{dist}(x, y, G)$. Let $\Delta(x, y)$ denote the set of all distance-critical vertices for pair (x, y), including the endpoints x and y; and define $\delta(x, y) = |\Delta(x, y)|$. Observe that $\delta(x, x) = 1$.

Given a path P, we denote by $V(P)$ and $E(P)$ the sets of vertices and edges that appear on P respectively. Given any two vertices $x, y \in P$, we say x ***precedes*** y on P, denoted by $x \preceq_P y$, if either $x = y$ or x appears before y on P. For vertices $x, y \in V$ and a path P where x precedes y on P, let $P[x, y]$ denote the sub-path of P from x to y. Similarly, for a vertex $x \in V$, an edge $e = (a, b) \in E$, and path P where x precedes a, let $P[x, e]$ denote the subpath $P[x, a]$. Define $P[e, x]$ for a suitable e, x and P analogously. Given two paths P and Q, their concatenation is denoted by $P \cdot Q$, provided that the last vertex of P coincides with the first vertex of Q. Given vertices x and y, let $V(x, y)$ and $E(x, y)$ denote the sets of all vertices and edges that lie on at least one shortest path between x and y in G respectively. Formally,

$$V(x, y) = \bigcup_{P \in \Pi(x,y)} V(P), \quad \text{and} \quad E(x, y) = \bigcup_{P \in \Pi(x,y)} E(P).$$

▶ **Definition 2.1** (Internally vertex-disjoint paths). *Let (x_1, y_1) and (x_2, y_2) be two pairs of vertices. A pair of distinct paths, one from x_1 to y_1 and the other from x_2 to y_2, are said to be internally vertex-disjoint if they do not share any vertices except those in the set $\{x_1, y_1\} \cap \{x_2, y_2\}$.*

In general, when two directed paths intersect at several vertices, the order in which these vertices appear in the two paths respectively could be different. The following lemma shows that there is more structure when these are shortest paths with a common destination.

▶ **Lemma 2.2.** *Let x_1, x_2, v be vertices and $P_1 \in \Pi(x_1, v)$ and $P_2 \in \Pi(x_2, v)$. Suppose P_1 and P_2 intersect internally and let S be the set of vertices in $P_1 \cap P_2$. Then the relative order of the vertices in S is the same in both P_1 and P_2.*

Proof. Suppose for the sake of contradiction that there are vertices $w_1, w_2 \in S$ such that w_1 appears before w_2 in P_1, but after w_2 in P_2. Since each sub-path of P_1 is also a shortest path, we see that $\text{dist}(w_1, v) > \text{dist}(w_2, v)$, but the reverse inequality is implied by the position of w_1 and w_2 in P_2. This leads to a contradiction. ◀

▶ **Definition 2.3** (Twin-crossing paths)**.** *Two paths P_1 and P_2 are said to be **twin-crossing** if there exist **distinct** vertices $a, b \in V$ such that a precedes b in both paths, and the sub-paths $P_1[a, b]$ and $P_2[a, b]$ are not identical to each other. Here, (a, b) is referred to as **twin-crossing pair** for paths P_1 and P_2.*

▶ **Lemma 2.4** (Schwartz-Zippel lemma)**.** *Let $P(x_1, x_2, \ldots, x_N)$ be a non-zero polynomial of degree d over a finite field $\mathbb{F}$. If a point is selected uniformly at random from $\mathbb{F}^N$, then the probability that P evaluates to a non-zero value is at least $1 - d/|\mathbb{F}|$.*

In this paper, we will work with fields of characteristic two. Thus $|\mathbb{F}|$ will be 2^q, for some integer $q \geqslant 1$. We will be working with q which is $O(\log n)$. This would support addition and multiplication over $\mathbb{F}$ in constant time in the Word-RAM model.

Enumerating Polynomials

For each edge $e = (u, v) \in E$, we introduce an indeterminate z_{uv} (also denoted by z_e). For any simple path P consisting of edges $e_1, \ldots, e_\ell$, we associate with it a monomial

$$f(P) := \prod_{i=1}^{\ell} z_{e_i}.$$

If P is an empty-path (that is, has zero edges), then $f(P)$ is defined to be 1. Note that the monomials corresponding to any two distinct simple paths are distinct. We now proceed to define the concept of enumerating polynomials.

▶ **Definition 2.5** (Enumerating polynomial)**.** *Given a collection $\mathcal{W}$ of paths, the enumerating polynomial $F_{\mathcal{W}}$ for $\mathcal{W}$ is defined as $F_{\mathcal{W}} := \sum_{P \in \mathcal{W}} f(P)$. Similarly, for a collection $\mathcal{W}$ of pairs of paths, the enumerating polynomial for the collection $\mathcal{W}$ is defined as $F_{\mathcal{W}} := \sum_{(P_1, P_2) \in \mathcal{W}} f(P_1) f(P_2)$.*

For any $x, y \in V$, we use $F(x, y)$ to denote the enumerating polynomial $F_{\mathcal{W}}$, where $\mathcal{W} = \Pi(x, y)$, i.e., the set of all shortest paths from x to y in G. That is,

$$F(x, y) := \sum_{P \in \Pi(x,y)} f(P).$$

[4] showed that $F(x, y)$ can be evaluated efficiently, we give a proof below for the sake of completeness.

▶ **Lemma 2.6.** *Let $G = (V, E)$ be a directed weighted graph with no negative weight cycles. For any assignment $\{\bar{z}_e\}_{e \in E}$, the polynomial $F(x, y)$, for all $x, y \in V$, can be evaluated in $O(mn + n^2 \log n)$ time.*

Proof. We first compute the all-pairs distance matrix using Johnson's algorithm [16] in $O(mn + n^2 \log n)$ time. Using this matrix, we evaluate $F(x, y)$ for all $x, y \in V$ as follows:

$$F(x, x) = 1, \quad F(x, y) = \sum_{(a,y) \in E(x,y)} F(x, a) \cdot z_{ay}, \tag{1}$$

where the membership $(a, y) \in E(x, y)$ can be verified in constant time using the distance matrix (recall that $E(x, y)$ is the set of edges that appear on a shortest path from x to y). To compute $F(x, y)$ efficiently for all vertex pairs, we fix a source vertex x and construct the DAG $G_x = (V, \bigcup_{v \in V} E(x, v))$. We then process the vertices of G_x in topological order, ensuring that for each vertex y, all required values $F(x, a)$ in Eq. 1 have already been computed prior to evaluating $F(x, y)$. Since each edge in E is processed once per source x, the computation of $F(x, y)$ values for all pairs can be performed in $O(mn)$ time, given the distance matrix of G. Therefore, the overall computation takes $O(mn + n^2 \log n)$ time. ◀

▶ **Definition 2.7** (Swap operation)**.** *For any pair of paths $(P_1, P_2) \in \Pi(x_1, y_1) \times \Pi(x_2, y_2)$ and any $a, b \in V$ such that a precedes b in both the paths, we define $\phi_{a,b}(P_1, P_2)$ to be the pair of paths obtained by swapping segments between a and b in P_1, P_2. That is, if P_i has endpoints x_i, y_i, then $Q_i = P_i[x_i, a] \cdot P_{3-i}[a, b] \cdot P_i[b, y_i]$, for $i = 1, 2$. We refer to $\phi_{a,b}$ as a swap operation at (a, b).*

The following claim proves that the swap operation is an involution.

▷ Claim 2.8. Let (P_1, P_2) be a pair of paths in $\Pi(x_1, y_1) \times \Pi(x_2, y_2)$ with (a, b) as a twin-crossing pair, and let $(Q_1, Q_2) = \phi_{a,b}(P_1, P_2)$. Then the pair $(Q_1, Q_2) \in \Pi(x_1, y_1) \times \Pi(x_2, y_2)$ also has (a, b) as twin-crossing. Further, $\phi_{a,b}(Q_1, Q_2) = (P_1, P_2)$ and the pair (Q_1, Q_2) is distinct from (P_1, P_2).

Proof. Let P_1 and P_2 be shortest paths between pairs (x_1, y_1) and (x_2, y_2) with (a, b) as a twin-crossing pair, i.e., $P_1[a, b] \neq P_2[a, b]$. Define $(Q_1, Q_2) = \phi_{a,b}(P_1, P_2)$. Since both P_1 and P_2 are shortest paths, exchanging $P_1[a, b]$ and $P_2[a, b]$ between them yields Q_1 and Q_2 that are also shortest paths from x_1 to y_1 and x_2 to y_2, respectively.

Moreover, applying $\phi_{a,b}$ again to (Q_1, Q_2) restores the original pair (P_1, P_2), since the swapped segments are simply exchanged back. As $P_1[a, b] \neq P_2[a, b]$, (Q_1, Q_2) is distinct from (P_1, P_2), and both share (a, b) as a twin-crossing pair because $Q_1[a, b] = P_1[a, b] \neq P_2[a, b] = Q_2[a, b]$. Thus, the claim holds. ◁

3 Technical Overview

In this section, we outline the technical novelties of our contribution.

3.1 2-DSP: From DAGs to Directed Graphs

[4] studied the 2-DSP problem for DAGs and undirected graphs, achieving optimal linear-time algorithms. The main idea behind their approach was to compute enumerating polynomials whose monomials are products of edge variables for each choice of vertex-disjoint (s_1, t_1) and (s_2, t_2) shortest paths. The polynomial $F(s_1, t_1)F(s_2, t_2)$ enumerates all shortest path pairs with no disjointness constraints.

For solving 2-DSP, they showed that it suffices to cancel the effect of intersecting paths from this total enumeration. Their key insight was that twin-crossing paths contribute zero in a field of characteristic 2: if P_1, P_2 are twin-crossing at vertices w, v, then swapping the segments between w and v results in paths Q_1, Q_2 with the same monomial contribution as P_1, P_2, and these contributions cancel each other in a characteristic 2 field. For the remaining case, namely, the scenario where non-twin crossing paths intersect for the first time at some vertex v, they gave techniques that, after precomputing all relevant $F(x, y)$ values, require only $O(\text{indeg}(v))$ time per vertex v. Summing over all vertices v resulted in the $O(m + n)$ time algorithm.

Challenge in Directed graphs. In directed graphs, the main challenge is that paths may intersect in more complicated ways, and there is no universally consistent "first common intersection" between P_1 and P_2 due to the absence of a global topological ordering. See Figure 1. Unlike DAGs where intersection patterns follow predictable structures, directed shortest paths can intersect and separate multiple times in a complex manner.

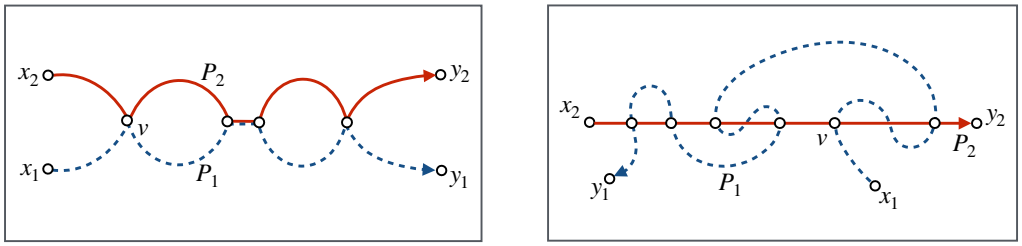


Figure 1 Illustration of interaction of shortest-paths in (a) DAGs, and (b) directed graphs having no cycles of negative or zero weight.

Our Approach. To tackle this complexity, we solve the 2-DSP problem by studying intersection patterns with respect to the first intersection point on a reference path $P_1 \in \Pi(s_1, t_1)$. Observe that in DAGs, a vertex v is the unique first intersection point for P_1 and P_2 if and only if prefixes of P_1 and P_2 up to v are internally vertex-disjoint. In comparison, in general directed graph, a vertex v is the first vertex on P_1 that is common between P_1 and P_2 if and only if the prefixes of P_1 and P_2 up to v, and the suffix of P_2 from v to t_2, *all these three segments*, are internally vertex-disjoint. See Figure 1.

We define $D_v(s_1, t_1, s_2, t_2)$ to be the enumerating polynomial of pairs $(P_1, P_2) \in \Pi(s_1, t_2) \times \Pi(s_2, t_2)$ such that

$$P_1[s_1, v], \ P_2[s_2, v], \text{ and } P_2[v, t_2] \tag{2}$$

are internally vertex-disjoint, or equivalently, v is the first vertex on P_1 that is common between P_1, P_2. Thus, if $F_{\text{disj}}(s_1, t_1, s_2, t_2)$ denotes the enumerating polynomial of all disjoint pairs of paths $(P_1, P_2) \in \Pi_1(s_1, t_1) \times \Pi_2(s_2, t_2)$, then

$$F_{\text{disj}}(s_1, t_1, s_2, t_2) := F(s_1, t_1)F(s_2, t_2) - \sum_v D_v(s_1, t_1, s_2, t_2).$$

Indeed, the second term in the right-hand side captures all intersecting pairs $(P_1, P_2) \in \Pi_1(s_1, t_1) \times \Pi_2(s_2, t_2)$ by cycling over all choices of the *first* vertex on P_1 where this intersection occurs. Thus, it suffices to check if the above polynomial is non-zero. Now, the disjointness conditions in (2) can be relaxed as follows: instead of requiring that $P_1[s_1, v]$ and $P_2[s_2, v]$

are disjoint, we just require that the incoming edges of v in these two paths respectively are distinct. Indeed, if $P_1[s_1, v]$ and $P_2[s_2, v]$ intersect while the incoming edges of v are distinct in these two prefixes, then there exist a common vertex $w \in P_1[s_1, v] \cap P_2[s_2, v]$ such that (w, v) forms a twin crossing pair for P_1, P_2. But again, a standard swapping argument shows the contribution of such pairs (P_1, P_2) in the sum $D_v(s_1, t_1, s_2, t_2)$ gets canceled in a field of characteristic 2.

In order to exploit the above observation, we define two additional polynomials. For any v, define $H_v(s_1, t_1, s_2, t_2)$ to be the enumerating polynomial of pairs $(P_1, P_2) \in \Pi(s_1, t_2) \times \Pi(s_2, t_2)$ containing v that satisfy

$$P_1[s_1, v] \text{ and } P_2[v, t_2]$$

are internally vertex-disjoint. Further, for any in-edge (a, v) of v, define H_{av} analogously with an additional requirement that the paths P_1, P_2 must contain the edge (a, v). The discussion above shows that $D_v(s_1, t_1, s_2, t_2)$, for any v, is given by (see Section 4 for a more formal proof):

$$H_v(s_1, t_1, s_2, t_2) - \sum_{a \in \Gamma^{\text{IN}}(v)} H_{av}(s_1, t_1, s_2, t_2).$$

The definition of $H_v(s_1, t_1, s_2, t_2)$ shows that it can be expressed in terms of $F_{\text{disj}}(s_1, v, v, t_2)$, and similarly for $H_{av}(s_1, t_1, s_2, t_2)$ (see Lemma 4.1 and 4.2). Thus, we are able to recursively express $F_{\text{disj}}(s_1, t_1, s_2, t_2)$ in terms of polynomials F_{disj} over a different set of quadruples. Finally, we show that overall only $O(m)$ quadruples are encountered in such manner, and their computation (i.e., evaluation over randomly chosen values) can be done efficiently using dynamic programming.

3.2 Solving Min-2-DSP in general directed graphs

The Minimum-2-Disjoint Shortest Paths (Min-2-DSP) problem is significantly more challenging than the standard 2-DSP problem. Let (s_1, t_1) and (s_2, t_2) be the input source-destination pairs, and let k^* denote the minimum possible overlap between any pair $(P_1, P_2) \in \Pi_1(s_1, t_1) \times \Pi_2(s_2, t_2)$ (we call such a pair an *optimal pair*).

One could define a suitable enumerating polynomial of all such optimal pairs. The first challenge in computing k^* arises when every optimal pair (P_1, P_2) is twin-crossing. Indeed, the standard swap operation would result in another optimal pair (Q_1, Q_2) such that the contributions of (P_1, P_2) and (Q_1, Q_2) would cancel out in the enumerating polynomial. While this cancellation property was crucial in our algorithm for the standard 2-DSP problem, it inadvertently causes the cancellation of contributions from optimal solutions in the Min-2-DSP setting. To overcome this limitation, we introduce the notion of a concordant pair.

▶ **Definition 3.1** (Concordant pair). *Let P_1 and P_2 be two paths with endpoints x_1, y_1 and x_2, y_2, respectively. A pair of vertices (a, b), not contained in the set $\{x_1, y_1\} \cap \{x_2, y_2\}$, is said to be a concordant pair for P_1 and P_2 if either $a = b$, or a appears before b on P_1 and P_2; and the subpaths $P_1[x_1, a], P_2[x_2, a]$ are internally vertex-disjoint, and likewise $P_1[b, y_1], P_2[b, y_2]$ are internally vertex-disjoint.*

Note that for general directed graphs, there can be multiple concordant pairs with respect to any pair of intersecting paths (see Figure 2). We denote by $\mathcal{C}(P_1, P_2)$ the set of all concordant pairs with respect to paths P_1, P_2.

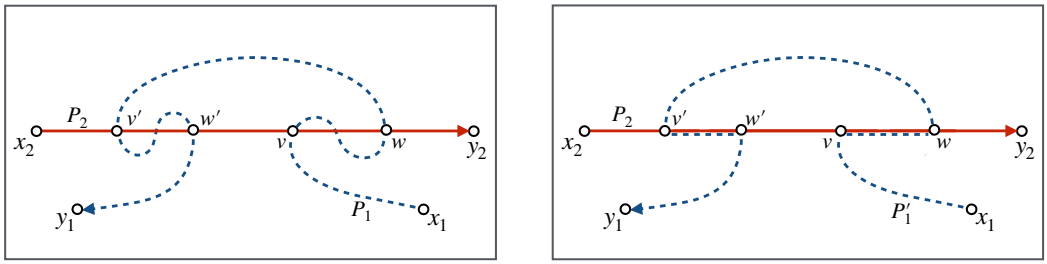


Figure 2 Illustration of paths P_1, P_2 with two concordant pairs. Observe that $\mathcal{C}(P_1, P_2) = \mathcal{C}(P_1', P_2)$.

▶ **Lemma 3.2.** *Let $(P_1, P_2) \in \Pi(s_1, t_1) \times \Pi(s_2, t_2)$ be a pair of paths, and $(v_1, w_1), \ldots, (v_r, w_r)$ be the concordant pairs with respect to P_1, P_2. Then, the following holds.*

1. (Vertex-disjointness) *For $i \neq j$, $V(P_1[v_i, w_i]) \cap V(P_1[v_j, w_j])$ and $V(P_2[v_i, w_i]) \cap V(P_2[v_j, w_j])$ are empty sets; that is, the subpaths corresponding to concordant pairs on P_1 (resp. P_2) are vertex-disjoint.*
2. (Order Reversal) *If the concordant pairs are ordered along P_1 as $(v_1, w_1), \ldots, (v_r, w_r)$, then on P_2 this ordering is exactly reversed.*

Intuitively, the concordant pairs capture the structural overlap for any pair of paths having minimum overlap. In order to explain this we present the following observation.

▶ **Observation 3.3.** *For any $v, w \in V$, there exist two shortest paths from v to w in G (possibly identical) that intersect only at (v, w)-distance critical vertices.*

Proof. Define a subgraph G_v^{out} of G by including only those edges $(a, b) \in E(G)$ that satisfy

$$\text{dist}(v, a) + wt(a, b) = \text{dist}(v, b).$$

That is, G_v^{out} contains exactly those edges that participate in some shortest path from v to b, for $b \in V$. It follows that G_v^{out} is acyclic, and for any $w \in V$, there is a bijection between (v, w)-shortest-paths in G and (v, w)-paths in G_v^{out}. This implies that the cut vertices for pair (v, w) in G_v^{out} correspond exactly to the (v, w)-distance critical vertices in G. By applying the Menger's theorem to G_v^{out}, we obtain two paths from v to w in G_v^{out}, intersecting at only (v, w)-cut vertices. These paths correspond to two (v, w)-shortest paths in G that intersect only at (v, w)-distance critical vertices. ◀

As a corollary we get the following.

▶ **Lemma 3.4.** *For any (s_1, t_1) and (s_2, t_2) shortest paths P_1, P_2 intersecting at exactly k^* internal vertices and any $(u, v) \in \mathcal{C}(P_1, P_2)$, the subpaths of P_1, P_2 from u to v have $\delta(u, v)$ vertices in common. Furthermore, these common vertices are precisely the (u, v)-distance-critical vertices.*

This suggests a shift in strategy: rather than directly searching for an optimal path pair, we can search for a pair of paths such that the total count of distance-critical vertices across all concordant pairs is minimized. This leads us to the following definition.

▶ **Definition 3.5.** *The interaction complexity of two paths P_1 and P_2 is defined as*

$$\gamma(P_1, P_2) = \sum_{(v,w) \in \mathcal{C}(P_1, P_2)} \delta(v, w).$$

Note that if P_1, P_2 is an optimal solution to Min-2-DSP then $\gamma(P_1, P_2) = k^*$. We define $F_{\text{disj},k}(s_1, t_1, s_2, t_2)$ as the enumerating polynomial of all pair of paths in $\Pi(s_1, t_1) \times \Pi(s_2, t_2)$ that have at most k interaction complexity. Our goal is to return the smallest k for which $F_{\text{disj},k}(s_1, t_1, s_2, t_2)$ is non-zero.

In order to compute $F_{\text{disj},k}$, we introduce several helper polynomials and present relations between them. These are more complicated in nature, so we defer the discussion to Section 5.

3.3 Solving Min-2-DSP in DAGs and Undirected graphs

For DAGs and undirected graphs, the Min-2-DSP problem admits a simpler and more efficient $O(m + n)$ time algorithm. First assume that given pairs (s_1, t_1) and (s_2, t_2), the desired minimum number of intersections k^* between any pair of paths $(P_1, P_2) \in \Pi(s_1, t_1) \times \Pi(s_2, t_2)$ is strictly positive (otherwise, we can employ the results in [4]). For a vertex v, let $\alpha(v)$ denote the minimum number of intersections of any pair of paths $(P_1, P_2) \in \Pi(s_1, t_1) \times \Pi(s_2, t_2)$ such that v lies on both of these paths. Since the optimal pair of paths must intersect at some vertex, it is not hard to show that k^* is equal to $\min_v \alpha(v)$ (if v does not lie on any pair of paths in $\Pi(s_1, t_1) \times \Pi(s_2, t_2)$, we can assume $\alpha(v)$ to be infinity).

A naive method for calculating $\alpha(v)$ would be following: (i) find the $(v, t_1), (v, t_2)$ shortest paths with minimum number of intersections (call this number $\alpha_1(v)$) and (ii) find the $(s_1, v), (s_2, v)$ shortest paths with minimum number of intersections (call this number $\alpha_2(v)$). If G is DAG, or if G is undirected and $k^* \geqslant 2$, then it can be seen that $\alpha(v) = \alpha_1(v) + \alpha_2(v)$. For each vertex v, $\alpha_1(v)$ and $\alpha_2(v)$ can be computed in $O(m+n)$ time by building dominator trees [17, 15] in appropriate subgraphs of G. This results in overall running time of $O(mn)$.

While it may seem that computing $\alpha(v)$ for any fixed v takes $O(m)$ time, our main insight is that this computation can be done more efficiently, such that computing $\alpha(v)$ over all $v \in V$ takes $O(m + n)$ total time. For doing so, we take inspiration from the notion of *bicoloured components* defined by Lochet [18] and partitioning the vertex set V into smaller components where Min-2-DSP reduces to Min-2-DP.

4 2-DSP in Directed graphs

For any vertex pairs (x_1, y_1) and (x_2, y_2), let $F_{\text{disj}}(x_1, y_1, x_2, y_2)$ be the enumerating polynomial of the collection of all pairs $(P_1, P_2) \in \Pi(x_1, y_1) \times \Pi(x_2, y_2)$ such that P_1 and P_2 are internally vertex-disjoint.

Observe that the polynomial $F_{\text{disj}}(x_1, y_1, x_2, y_2)$ is non-zero (in a field of characteristic 2) if and only if there exist two shortest paths, one from x_1 to y_1 and the other from x_2 to y_2, that are internally vertex-disjoint. This follows from the fact that each such pair contributes a distinct monomial to $F_{\text{disj}}(x_1, y_1, x_2, y_2)$, determined by the set of edges used along both paths. Indeed, if (P_1, P_2) is a pair of internally vertex-disjoint shortest paths for (x_1, y_1) and (x_2, y_2), then for each $i = 1, 2$, the path from x_i to y_i remains unique in the subgraph $G' = (V, E(P_1) \cup E(P_2))$, unless the two vertex pairs coincide.

4.1 Helper polynomials

In this subsection, we will provide construction of various helper polynomials that will be crucial in solving 2-DSP in directed graphs. The helper polynomials introduced in this section will be helpful in providing a recursive formulation for the polynomial $F_{\text{disj}}(x_1, y_1, x_2, y_2)$.

I. Disjoint Prefix-Suffix linkages up to a Shared Vertex/Edge

For any $v \in V$, let $H_v(x_1, y_1, x_2, y_2)$ be the enumerating polynomial of all pairs $(P_1, P_2) \in \Pi(x_1, y_1) \times \Pi(x_2, y_2)$ such that
1. $v \in V(P_1) \cap V(P_2)$, and
2. Prefix $P_1[x_1, v]$ and suffix $P_2[v, y_2]$ are internally vertex-disjoint.

Similarly, for any edge $e = (a, v) \in E$, let $H_{av}(x_1, y_1, x_2, y_2)$ be the enumerating polynomial of all all pairs $(P_1, P_2) \in \Pi(x_1, y_1) \times \Pi(x_2, y_2)$ such that
1. $e \in E(P_1) \cap E(P_2)$, and
2. Prefix $P_1[x_1, e]$ and suffix $P_2[e, y_2]$ are internally vertex-disjoint.

The next two lemmas present relations between the polynomials H_v, H_{av}, and F_{disj}, using a reasoning similar to those in Lemma 33 of [3] for solving 2-DSP in DAGs.

▶ **Lemma 4.1.** *For any $v \in V$ satisfying $\text{dist}(x_i, y_i) = \text{dist}(x_i, v) + \text{dist}(v, y_i)$, for $i = 1, 2$, we have*

$$H_v(x_1, y_1, x_2, y_2) = F(x_2, v) \cdot F_{\text{disj}}(x_1, v, v, y_2) \cdot F(v, y_1).$$

Proof. Consider a pair of shortest paths $(P_1, P_2) \in \Pi(x_1, y_1) \times \Pi(x_2, y_2)$ contributing to $H_v(x_1, y_1, x_2, y_2)$. For $i = 1, 2$, decompose $P_i = Q_i \cdot R_i$, where $Q_i = P_i[x_i, v]$ and $R_i = P_i[v, y_i]$ are shortest paths. Then,

$$f(P_1)f(P_2) = \prod_{i=1}^{2} \big(f(Q_i) \cdot f(R_i)\big).$$

Thus, $H_v(x_1, y_1, x_2, y_2)$ enumerates all such tuples (Q_1, R_1, Q_2, R_2) of shortest paths satisfying the following.
- For each $i = 1, 2$, Q_i is an (x_i, v)-shortest path and R_i is a (v, y_i)-shortest path.
- The paths Q_1 and R_2 are internally vertex-disjoint.

The contribution of all such internally disjoint path pairs (Q_1, R_2) is captured by the polynomial $F_{\text{disj}}(x_1, v, v, y_2)$. Meanwhile, the contributions of the unconstrained paths Q_2 and R_1, which are only required to be shortest paths between their endpoints, are given by $F(x_2, v)$ and $F(v, y_1)$, respectively. Multiplying these three components yields the desired expression. ◀

▶ **Lemma 4.2.** *For any $e = (a, v) \in E$, satisfying $\text{dist}(x_i, y_i) = \text{dist}(x_i, a) + wt(a, v) + \text{dist}(v, y_i)$, for $i = 1, 2$, we have*

$$H_{av}(x_1, y_1, x_2, y_2) = F(x_2, a) \cdot z_{av}^2 \cdot F_{\text{disj}}(x_1, a, v, y_2) \cdot F(v, y_1).$$

Proof. Consider a pair of shortest paths (P_1, P_2) contributing to $H_{av}(x_1, y_1, x_2, y_2)$. For $i = 1, 2$, decompose each $P_i = Q_i \cdot e \cdot R_i$, where $Q_i = P_i[x_i, a]$ and $R_i = P_i[v, y_i]$ are shortest paths. Then,

$$f(P_1)f(P_2) = \prod_{i=1}^{2} \big(f(Q_i) \cdot z_{av} \cdot f(R_i)\big) = z_{av}^2 \cdot \prod_{i=1}^{2} f(Q_i)f(R_i).$$

Thus, $H_{av}(x_1, y_1, x_2, y_2)/z_{av}^2$ enumerates all such tuples (Q_1, R_1, Q_2, R_2) of shortest paths satisfying the following.
- For each $i = 1, 2$, Q_i is an (x_i, a)-shortest path and R_i is a (v, y_i)-shortest path.
- The paths Q_1 and R_2 are internally vertex-disjoint.

The contribution of all such internally disjoint path pairs (Q_1, R_2) is captured by the polynomial $F_{\text{disj}}(x_1, a, v, y_2)$. Meanwhile, the contributions of the unconstrained paths Q_2 and R_1, which are only required to be shortest paths between their endpoints, are given by $F(x_2, a)$ and $F(v, y_1)$, respectively. Multiplying these components together, along with the z_{av}^2 term, yields the desired expression. ◀

II. Paths with First Intersection at a Given Vertex along a Reference Path

We define $D_v(x_1, y_1, x_2, y_2)$ to be the enumerating polynomial of the collection of pairs $(P_1, P_2) \in \Pi(x_1, y_1) \times \Pi(x_2, y_2)$ whose first intersection point with respect to P_1 is vertex v.[2] More formally, the paths P_1 and P_2 satisfy the following.

1. $v \in V(P_1) \cap V(P_2)$, and
2. Prefix $P_1[x_1, v]$ is internally vertex-disjoint with $P_2[x_2, v]$ as well as $P_2[v, y_2]$.

It is important to observe that if $v \notin V(x_1, y_1) \cap V(x_2, y_2)$, then $D_v(x_1, y_1, x_2, y_2)$ will be a zero polynomial. We next present a relation between polynomials H_v, H_{av}, and D_v. The lemma below crucially uses the fact that the underlying field has characteristic two.

▶ **Lemma 4.3.** *For any $v \in V$ satisfying $\text{dist}(x_i, y_i) = \text{dist}(x_i, v) + \text{dist}(v, y_i)$ for $i = 1, 2$, we have*

$$D_v(x_1, y_1, x_2, y_2) \;=\; H_v(x_1, y_1, x_2, y_2) - \sum_{a \in \Gamma^{\text{IN}}(v)} H_{av}(x_1, y_1, x_2, y_2).$$

Proof. Let $\mathcal{A}$ be a collection of all pairs of paths $(P_1, P_2) \in \Pi(x_1, y_1) \times \Pi(x_2, y_2)$ such that

1. $v \in V(P_1) \cap V(P_2)$,
2. prefix $P_1[x_1, v]$ and suffix $P_2[v, y_2]$ are internally vertex-disjoint, and
3. if a_1 and a_2 are the vertices preceding v in P_1 and P_2 respectively (assuming these vertices exist, otherwise this condition holds vacuously), then $a_1 \neq a_2$.

The definitions of the polynomials H_v and H_{av} imply that the enumerating polynomial for the pairs in $\mathcal{A}$ is given by

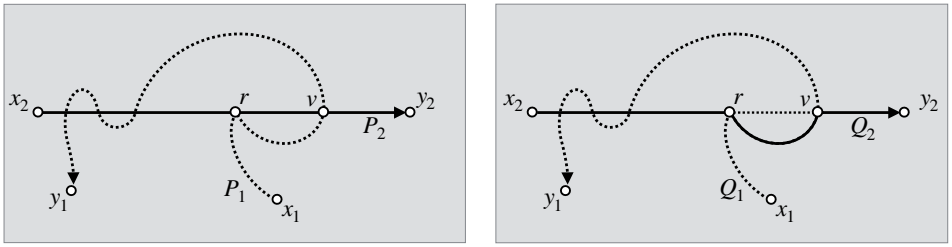
$$H_v(x_1, y_1, x_2, y_2) - \sum_{a \in \Gamma^{\text{IN}}(v)} H_{av}(x_1, y_1, x_2, y_2).$$

Let $\mathcal{B}$ be the set of all pairs of paths (P_1, P_2) contributing to $D_v(x_1, y_1, x_2, y_2)$. In order to prove the claim it suffices to prove that the enumerating polynomials of $\mathcal{A}$ and $\mathcal{B}$ are identical (mod 2). First, observe that $\mathcal{B} \subseteq \mathcal{A}$. Indeed, if $(P_1, P_2) \in \mathcal{B}$, then $P_1[x_1, v]$ is internally vertex-disjoint from both $P_2[x_2, v]$ and $P_2[v, y_2]$, which in particular ensures that the in-edges of v in P_1 and P_2 (if they exist) must be distinct.

Now consider any pair $(P_1, P_2) \in \mathcal{A} \setminus \mathcal{B}$. The definitions of $\mathcal{A}$ and $\mathcal{B}$ imply that the following two conditions must hold: (i) the sub-paths $P_1[x_1, v]$ and $P_2[x_2, v]$ have a common internal vertex, (ii) the edges preceding v in these two sub-paths respectively are distinct. Let r be the **immediate** predecessor of v common to both the paths. It follows that (r, v) is a twin-crossing pair for (P_1, P_2). Let $(Q_1, Q_2) = \phi_{r,v}(P_1, P_2)$. Claim 2.8 shows that (Q_1, Q_2) has (r, v) as twin-crossing pair and hence lie in $\mathcal{A} \setminus \mathcal{B}$ (see Figure 3). Further, it follows that r must be the immediate predecessor of v on Q_1, Q_2 and $(P_1, P_2) = \phi_{r,v}(Q_1, Q_2)$.

[2] Note that v may not be the first/last intersection point with respect to path P_2.

Thus, the set $\mathcal{A} \setminus \mathcal{B}$ can be partitioned into pairs $\{(P_1, P_2), (Q_1, Q_2)\}$, where $(Q_1, Q_2) = \phi_{r,v}(P_1, P_2)$ and r is the immediate predecessor of v on P_1, P_2 (as well as, on Q_1, Q_2). Since $f(P_1)f(P_2) = f(Q_1)f(Q_2)$, we see that each such pair contributes two identical monomials to the enumerating polynomial of $\mathcal{A} \setminus \mathcal{B}$. Thus, the total contribution of $\mathcal{A} \setminus \mathcal{B}$ is a multiple of two, and hence zero in a field of characteristic two. This establishes the claim. ◀



(a) (b)

Figure 3 Illustration of the swap operation on pair $(P_1, P_2) \in \mathcal{A} \setminus \mathcal{B}$.

III. Intersecting and Disjoint Shortest paths

We define $F_\cap(x_1, y_1, x_2, y_2)$ to be the enumerating polynomial for all pairs of shortest paths between pairs (x_1, y_1) and (x_2, y_2) that are not internally vertex-disjoint. Observe that

$$F_\cap(x_1, y_1, x_2, y_2) = \sum_{v \in V \setminus (\{x_1, y_1\} \cap \{x_2, y_2\})} D_v(x_1, y_1, x_2, y_2).$$

Indeed, if two paths P_1, P_2 are intersecting (at an internal vertex), there must exist a unique earliest intersection point v on P_1 (that lies outside the set $\{x_1, y_1\} \cap \{x_2, y_2\}$). Since every pair of (x_1, y_1) and (x_2, y_2) shortest paths is either internally vertex-disjoint or else intersects at some common vertex outside the set $\{x_1, y_1\} \cap \{x_2, y_2\}$, we obtain the following result.

▶ **Lemma 4.4.** *The enumerating polynomial $F_{\text{disj}}(x_1, y_1, x_2, y_2)$ is given by*

$$F_{\text{disj}}(x_1, y_1, x_2, y_2) = F(x_1, y_1)F(x_2, y_2) - \sum_{v \in V \setminus (\{x_1, y_1\} \cap \{x_2, y_2\})} D_v(x_1, y_1, x_2, y_2).$$

4.2 Solving 2-DSP in $O(m^2 \log n)$ time

In this subsection, we present an algorithm for solving the 2-DSP problem in weighted directed graphs that takes $O(m^2)$ time. Let (s_1, t_1) and (s_2, t_2) be the input terminal pairs. By Schwartz-Zippel lemma (Lemma 2.4), in order to check if $F_{\text{disj}}(s_1, t_1, s_2, t_2)$ is non-zero it suffices to evaluate this polynomial on a random assignment of the variables $\{z_e\}_{e \in E}$ drawn from a sufficiently large finite field $\mathbb{F}$. The algorithm for evaluating $F_{\text{disj}}(s_1, t_1, s_2, t_2)$ at these random values is given in Algorithm 1.

We begin by assigning random values $\bar{z}_e$ from $\mathbb{F}$ to each of the edge variables z_e respectively. For a polynomial R involving the variables $z_e, e \in E$, we shall use $\bar{R}$ to denote the value in $\mathbb{F}$ obtained by evaluating R at $\bar{z}_e, e \in E$. Using Lemma 2.6, we compute $\bar{F}(x, y)$ for all pairs of vertices in $O(mn \log n)$ time. In order to compute $\bar{F}_{\text{disj}}(s_1, t_1, s_2, t_2)$, we would need to compute $\bar{F}_{\text{disj}}(x_1, y_1, x_2, y_2)$ for several tuples (x_1, y_1, x_2, y_2). We store all such tuples in a list $\mathcal{L}$. In particular, $\mathcal{L}$ contains the following: (i) tuples (s_1, v, v, t_2), for each $v \in V$; (ii) tuples (s_1, a, v, t_2), for each $(a, v) \in E$; and (iii) the query tuple (s_1, t_1, s_2, t_2).

Algorithm 1 2-DSP-DIRECTED(G, s_1, t_1, s_2, t_2).

Assign each edge $(u, v) \in E$ an independent and uniformly random value $\bar{z}_{uv}$ from $\mathbb{F}$;
Use Lemma 2.6 to compute $\bar{F}(x, y)$ for all $x, y \in V$;
Compute the ordered list $\mathcal{L}^*$ using Lemma 4.6;
foreach (x_1, y_1, x_2, y_2) *in the ordered list* $\mathcal{L}^*$ **do**
  if $x_1 = y_1$ and $x_2 = y_2$ **then** Set $\bar{F}_{\text{disj}}(x_1, y_1, x_2, y_2) = 1$ and **continue** for-loop;
  foreach $v \in V \setminus (\{x_1, y_1\} \cap \{x_2, y_2\})$ **do**
    if $v \in V(x_1, y_1) \cap V(x_2, y_2)$ **then**
      Compute $\bar{D}_v(x_1, y_1, x_2, y_2)$ as

$$\bar{F}(x_2, v) \cdot \bar{F}_{\text{disj}}(x_1, v, v, y_2) \cdot \bar{F}(v, y_1) - \sum_{\substack{(a,v) \in \\ E(x_1,y_1), \\ E(x_2,y_2)}} z_{av}^2 \cdot F(x_2, a) \cdot \bar{F}_{\text{disj}}(x_1, a, v, y_2) \cdot \bar{F}(v, y_1);$$

    else
      Set $\bar{D}_v(x_1, y_1, x_2, y_2) = 0$;
  Compute

$$\bar{F}_{\text{disj}}(x_1, y_1, x_2, y_2) = \bar{F}(x_1, y_1) \cdot \bar{F}(x_2, y_2) - \sum_{v \in V \setminus (\{x_1, y_1\} \cap \{x_2, y_2\})} \bar{D}_v(x_1, y_1, x_2, y_2);$$

Return True if $\bar{F}_{\text{disj}}(s_1, t_1, s_2, t_2) \neq 0$, else False;

Now we evaluate $\bar{F}_{\text{disj}}(x_1, y_1, x_2, y_2)$ for each such tuple in $\mathcal{L}$ using dynamic programming. For this, we define a partial order $\prec$ on the set of vertices V of the graph G as follows: for any distinct $x, y \in V$, we say $x \prec y$ if there exists an s_1-to-y shortest path in G that contains x (as an internal vertex). This is well-defined, as G contains no cycles of zero or negative weight.

▶ **Definition 4.5.** *Given the partial order $\prec$ on V, let $\mathcal{L}^*$ denote an ordered list such that tuples of the form (s_1, b, u, t_2), (s_1, u, u, t_2) precede tuples like (s_1, a, v, t_2), (s_1, v, v, t_2) whenever $u \prec v$, i.e., whenever u lies on an (s_1, v) shortest path. The tuple (s_1, t_1, s_2, t_2) appears at the end of the list.*

Computing F_{disj} for tuples in the order given by $\mathcal{L}^*$ ensures that that while evaluating $\bar{F}_{\text{disj}}(x_1, y_1, x_2, y_2)$, all the values that are required for this computation are available. In the base case, when $x_1 = y_1$ and $x_2 = y_2$, we set $\bar{F}_{\text{disj}}(x_1, y_1, x_2, y_2) = 1$.

To evaluate $\bar{F}_{\text{disj}}(x_1, y_1, x_2, y_2)$, we use the identity (see Lemma 4.4):

$$\bar{F}_{\text{disj}}(x_1, y_1, x_2, y_2) = \bar{F}(x_1, y_1)\bar{F}(x_2, y_2) - \sum_{v \in V \setminus (\{x_1, y_1\} \cap \{x_2, y_2\})} \bar{D}_v(x_1, y_1, x_2, y_2).$$

By Lemma 4.3 ,$\bar{D}_v$ can be expressed as

$$\bar{D}_v(x_1, y_1, x_2, y_2) = \bar{H}_v(x_1, y_1, x_2, y_2) - \sum_{(a,v) \in E} \bar{H}_{av}(x_1, y_1, x_2, y_2),$$

for each $v \in V(x_1, y_1) \cap (x_2, y_2)$. Now, by Lemma 4.1, we have

$$\bar{H}_v(x_1, y_1, x_2, y_2) = \bar{F}(x_2, v) \cdot \bar{F}_{\text{disj}}(x_1, v, v, y_2) \cdot \bar{F}(v, y_1).$$

We shall show in Lemma 4.6 (for proof refer to the full version [11]) that the ordering induced by $\mathcal{L}^*$ ensures that the right-hand side above has already been computed by the dynamic program. Similarly, by Lemma 4.2, for any edge (a, v),

$$\bar{H}_{av}(x_1, y_1, x_2, y_2) = \bar{F}(x_2, a) \cdot z_{av}^2 \cdot \bar{F}_{\text{disj}}(x_1, a, v, y_2) \cdot \bar{F}(v, y_1).$$

Finally, we output that there exists two disjoint shortest paths between (s_1, t_1) and (s_2, t_2) if and only if $\bar{F}_{\text{disj}}(s_1, t_1, s_2, t_2)$ evaluates to non-zero.

▶ **Lemma 4.6.** *We can compute in $O(mn)$ time the ordered list $\mathcal{L}^*$ so that, when invoking pairs according to this ordering in our dynamic program, all subproblems required are already computed.*

Correctness and Running Time. The correctness of the procedure follows from the recursive expressions in Lemma 4.1, Lemma 4.2, Lemma 4.3, and Lemma 4.4. We now analyze the running time. First observe that $\mathcal{L}$ has $O(m)$ tuples: there are at most n choices for the tuples (s_1, v, v, t_2) and at most m choices for (s_1, a, v, t_2). The description of the algorithm shows that we can evaluate $\bar{D}_v(x_1, y_1, x_2, y_2)$ in $O(\text{indeg}(v))$ time for any fixed v, and hence we can compute $\bar{F}_{\text{disj}}(x_1, y_1, x_2, y_2)$ in $O(m)$ total time per quadruple. Since we evaluate these polynomials for $O(m)$ quadruples in $\mathcal{L}$, the overall running time is bounded by $O(m^2)$, after an initial preprocessing step of $O(mn \log n)$. This yields a randomized $O(m^2 \log n)$-time algorithm for 2-DSP with high success probability.

4.3 An $O(mn \log n)$ time algorithm for 2-DSP

We now present an improved algorithm for solving the 2-DSP problem in $O(mn \log n)$ time. Recall from the previous subsection that Algorithm 1 computes $F_{\text{disj}}(s_1, t_1, s_2, t_2)$ by evaluating F_{disj} for all quadruples in a list $\mathcal{L}$ of size $O(m)$, where each evaluation incurs $O(m)$ time. Note that we can afford $O(m)$ time computation for evaluating $\bar{F}_{\text{disj}}(s_1, v, v, t_2)$ for each v – since there are only $O(n)$ such vertices, this would incur a total of $O(mn)$ time. Therefore, our focus will be on improving the computation of $\bar{F}_{\text{disj}}(s_1, a, v, t_2)$, where (a, v) is an edge in G.

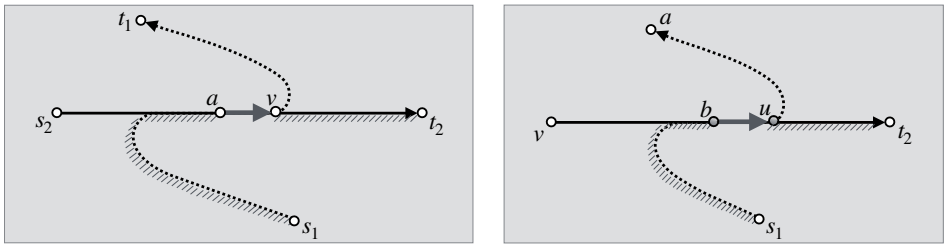


Figure 4 Illustration of interaction between shortest paths contributing to the polynomials (i) $H_{av}(s_1, t_1, s_2, t_2)$, and (ii) its building block $H_{bu}(s_1, a, v, t_2)$. (Note that only the shaded prefix/suffix segments need to be vertex-disjoint.)

Consider a vertex $v \in V(s_1, t_1) \cap V(s_2, t_2)$. We show how to compute $\bar{F}_{\text{disj}}(s_1, a, v, t_2)$, for all relevant edges $(a, v) \in \mathcal{E}^{\text{IN}}(v)$ in $O(m)$ total time.

Let $a \in \Gamma^{\text{IN}}(v)$ be such that $(a, v) \in E(s_1, t_1) \cap E(s_2, t_2)$. Now, Lemma 4.4 shows that computation of $\bar{F}_{\text{disj}}(s_1, a, v, t_2)$ requires us to evaluate $\bar{D}_u(s_1, a, v, t_2)$ for each $u \in V(s_1, a) \cap V(v, t_2)$ (see Figure 4). Now, line 8 in Algorithm 1 shows that

$$\bar{D}_u(s_1,a,v,t_2) = \bar{F}(v,u)\cdot\bar{F}_{\text{disj}}(s_1,u,u,t_2)\cdot\bar{F}(u,a) - \sum_{\substack{(b,u)\ \in \\ E(s_1,a),\\ E(v,t_2)}} z_{bu}^2\cdot\bar{F}(v,b)\cdot\bar{F}_{\text{disj}}(s_1,b,u,t_2)\cdot\bar{F}(u,a)$$

$$= \bar{F}(v,u)\cdot\bar{F}_{\text{disj}}(s_1,u,u,t_2)\cdot\bar{F}(u,a) - \underbrace{\left(\sum_{\substack{(b,u)\ \in \\ E(s_1,\boldsymbol{u}),\\ E(v,t_2)}} z_{bu}^2\cdot\bar{F}(v,b)\cdot\bar{F}_{\text{disj}}(s_1,b,u,t_2)\right)}_{\text{independent of choice of edge } (a,v)}\cdot\bar{F}(u,a)\ .$$

The last inequality holds because $\mathcal{E}^{\text{IN}}(u)\cap E(s_1,a) = \mathcal{E}^{\text{IN}}(u)\cap E(s_1,u)$, for any $u\in V(s_1,a)$. Specifically, (i) if $(b,u)\in E(s_1,a)$, then $(b,u)\in E(s_1,u)$ by the substructure property of shortest paths; (ii) conversely, if $(b,u)\in E(s_1,u)$ and u lies on an (s_1,a)-shortest path P, then $(b,u)\in E(s_1,a)$, since the prefix of P up to u can be replaced by an (s_1,u)-shortest path passing through (b,u). We emphasize that the sum in the parenthesis above depends only on u and v, and not on the specific choice of $a\in\Gamma^{\text{IN}}(v)$. Hence, we can pre-evaluate this expression once per u in $O(\text{indeg}(u))$ time and reuse it for all relevant in-neighbors a of v. Since $\sum_u \text{indeg}(u) = O(m)$, the total time across all such computations for a fixed vertex v is $O(m)$.

Summing over all $v\in V(s_1,t_1)\cap V(s_2,t_2)$ gives a total time of $O(mn)$ to evaluate all required instances of F_{disj}.

Finally, using Lemma 2.6, we can evaluate all $\bar{F}(x,y)$ values in $O(mn\log n)$ time. Hence, the overall running time of our improved algorithm for detecting two disjoint shortest paths becomes $O(mn\log n)$. Thus, we can conclude with the following theorem.

▶ **Theorem 4.7.** *The 2-DSP problem in directed weighted graphs having no cycles of negative or zero weight is solvable in $O(mn\log n)$ time.*

4.4 Reporting the Paths

We now show how to compute a pair of internally vertex-disjoint shortest paths from s_1 to t_1 and from s_2 to t_2 in $O(mn^2\log n)$-time, if such paths exist.

Recall that the proof of Theorem 4.7 provides an $O(mn\log n)$ time algorithm for solving the 2-DSP problem for a collection of $O(m)$ quadruples lying in a list $\mathcal{L}^*$. In order to report the paths, we show how to compute an out-neighbor w^* of s_2 that appears in some solution to the 2-DSP problem in the same time bound. Recall that the list $\mathcal{L}^*$ comprises of (i) tuples (s_1,v,v,t_2), for each $v\in V$; (ii) tuples (s_1,a,v,t_2), for each $(a,v)\in E$; and (iii) the query tuple (s_1,t_1,s_2,t_2).

To compute the out-neighbor w^* of s_2, we remove the tuple (s_1,t_1,s_2,t_2) from list $\mathcal{L}^*$, and append the tuples (s_1,t_1,w,t_2), for $w\in\Gamma^{\text{OUT}}(s_2)\cap V(s_2,t_2)$. Now, we can solve the 2-DSP problem for all pairs in $\mathcal{L}^*$ in the graph $G\setminus\{s_2\}$, in the same time bound of $O(mn\log n)$. By Schwartz-Zippel lemma (Lemma 2.4), with high probability, the out-neighbors of s_2 that participates in solution to the 2-DSP problem are precisely those $w^*\in\Gamma^{\text{OUT}}(s_2)\cap V(s_2,t_2)$ for which $\bar{F}_{\text{disj}}(s_1,t_1,w^*,t_2)$ evaluates to non-zero in the graph $G\setminus\{s_2\}$. After finding the node w^*, we **delete** vertex s_2 from G, and consider a smaller instance of the 2-DSP problem, where source s_2 is replaced with w^*. This process is repeated at most $O(n)$ times, helping us recover an (s_2,t_2)-shortest path P_2, disjoint from some (s_1,t_1)-shortest path in G.

Finally, we delete the vertices of P_2 from G, and find an (s_1,t_1)-shortest path P_1 in the resulting graph in $O(mn)$ time using the Bellman-Ford algorithm. This provides us an algorithm for reporting a pair of disjoint (s_1,t_1) and (s_2,t_2) shortest paths, if they exists, in $O(mn^2\log n)$ time.

Theorem 4.7 together with discussion above proves the following result.

▶ **Theorem 1.1.** *The 2-DSP problem in directed weighted graphs having no cycles of negative or zero weight is solvable in $O(mn \log n)$ time. Moreover, we can report the explicit paths, if they exist, in $O(mn^2 \log n)$ time.*

5 Min-2-DSP in Directed graphs

In this section, we consider the Min-2-DSP problem in directed graphs with positive edge weights and show that it can be solved in $O(m^2n^3)$ time. For any vertex pairs (x_1, y_1) and (x_2, y_2), and any integer k, let $F_{\text{disj},k}(x_1, y_1, x_2, y_2)$ be the enumerating polynomial of all pairs of shortest paths P_1 from x_1 to y_1 and P_2 from x_2 to y_2 such that the *interaction complexity*

$$\gamma(P_1, P_2) = \sum_{(v,w)\in\mathcal{C}(P_1,P_2)} \delta(v, w)$$

is at most k. In order to solve Min-2-DSP it suffices to determine the smallest integer k for which polynomial $F_{\text{disj},k}(s_1, t_1, s_2, t_2)$ is non-zero, where (s_1, t_1) and (s_2, t_2) are input source-destination pairs.

For any vertex pairs (x_1, y_1) and (x_2, y_2), any integer k, and vertices $v, w \in V$, define

$$D_{v,w,k}(x_1, y_1, x_2, y_2)$$

to be the enumerating polynomial of all pairs $(P_1, P_2) \in \Pi(x_1, y_1) \times \Pi(x_2, y_2)$, having interaction complexity at most k and such that (v, w) is the *first (with respect to P_1)* concordant pair in $\mathcal{C}(P_1, P_2)$.

The following lemma provides relation between polynomials $F_{\text{disj},k}$ and $D_{v,w,k}$.

▶ **Lemma 5.1.** *Let (x_1, y_1) and (x_2, y_2) be pairs satisfying the property that any pair of paths $(P_1, P_2) \in \Pi(x_1, y_1) \times \Pi(x_2, y_2)$ intersect internally. Then, for any integer $k \geqslant 1$, we have*

$$F_{\text{disj},k}(x_1, y_1, x_2, y_2) = \sum_{v,w\in V} D_{v,w,k}(x_1, y_1, x_2, y_2)$$

Proof. Let $(P_1, P_2) \in \Pi(x_1, y_1) \times \Pi(x_2, y_2)$ be a pair of paths with interaction complexity k. Furthermore, let $(v, w) \in \mathcal{C}(P_1, P_2)$ be the first concordant pair appearing on P_1. Then, the enumerating polynomial for such pairs is given by $D_{v,w,k}(x_1, y_1, x_2, y_2)$. To obtain $F_{\text{disj},k}(x_1, y_1, x_2, y_2)$, we sum over all choices of concordant pairs. This proves the desired claim. ◀

To efficiently determine whether $D_{v,w,k}(x_1, y_1, x_2, y_2)$ is a non-zero polynomial, we define next some helper polynomials.

5.1 Additional Helper Polynomials

For any vertex pairs (x_1, y_1) and (x_2, y_2), any integer k, and any $v, w \in V$, define

$$H_{v,w,k}(x_1, y_1, x_2, y_2)$$

to be the enumerating polynomial for all pairs of shortest paths (P_1, P_2) between (x_1, y_1) and (x_2, y_2) such that:

1. the subpaths $P_1[x_1, v]$ and $P_2[w, y_2]$ are internally vertex-disjoint; and
2. the interaction complexity of $P_1[v, y_1]$ and $P_2[x_2, w]$ is at most k.

Similarly, we define the polynomials $H_{av,w,k}$, $H_{v,wb,k}$, and $H_{av,wb,k}$, which satisfy the above constraints with the additional requirement that the paths P_1, P_2 must contain the edges (a, v), (w, b), and both (a, v) and (w, b), respectively.

Note that in the definitions of $H_{v,w,k}$, $H_{av,w,k}$, $H_{v,wb,k}$, and $H_{av,wb,k}$ any in-neighbor of v or out-neighbor of w that appears in both P_1 and P_2 is not counted in the overlap budget k as the overlap constraint in condition two applies exclusively to the number of common vertices between the suffix of P_1 starting from v and prefix of P_2 up to w.

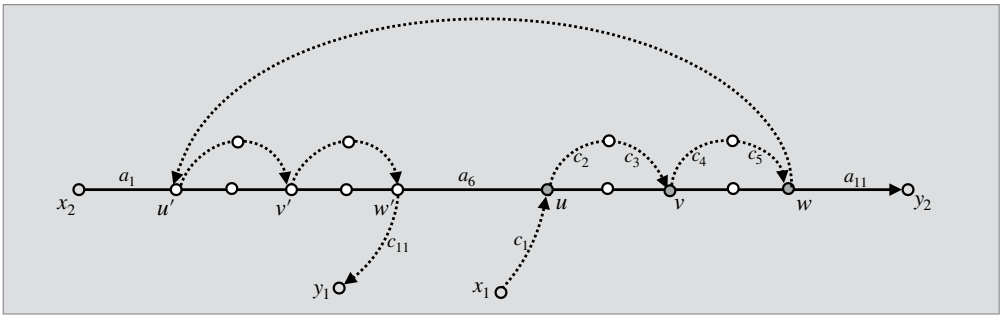


Figure 5 The directed graph above shows multiple shortest paths between (x_1, y_1) (eg. dashed edges $c_1, \ldots, c_{11}$) and (x_2, y_2) (eg. solid edges $a_1, \ldots, a_{11}$); For any choice of (x_1, y_1) and (x_2, y_2) shortest paths (P_1, P_2), we have $\mathcal{C}(P_1, P_2) = \{(u, w), (u', w')\}$, thereby implying $F_{\text{disj},6}(x_1, y_1, x_2, y_2)/(F(u, w)^2 \cdot F(u', w')^2) = z_{a_1} z_{a_6} z_{a_{11}}\, z_{c_1} z_{c_6} z_{c_{11}}$.

5.2 Properties of Helper Polynomials

We first present a relation between $D_{v,w,k}$ and polynomials $H_{v,w,k}$, $H_{av,w,k}$, $H_{v,wb,k}$, and $H_{av,wb,k}$ (for proof see the full version [11]).

▶ **Lemma 5.2.** *For any quadruple $\sigma = (x_1, y_1, x_2, y_2) \in V^4$, integer $k \geqslant 1$, and any $v, w \in V$ such that v precedes w on some (x_1, y_1) and (x_2, y_2) shortest paths, the following holds:*

$$D_{v,w,k}(\sigma) \;=\; H_{v,w,k}(\sigma) - \sum_{(a,v)\in E} H_{av,w,k}(\sigma) - \sum_{(w,b)\in E} H_{v,wb,k}(\sigma) + \sum_{\substack{(a,v),(w,b)\\ \in E}} H_{av,wb,k}(\sigma)$$

Next we provide construction of other helper polynomials.

▶ **Lemma 5.3.** *For any integer k, and any $v, w \in V$ such that v precedes w on some (x_1, y_1) and (x_2, y_2) shortest paths, the following holds:*

$$H_{v,w,k}(x_1, y_1, x_2, y_2) = F_{\text{disj}}(x_1, v, w, y_2) \cdot F(v, w)^2 \cdot F_{\text{disj},k-\delta(v,w)}(w, y_1, x_2, v)$$

Proof. Consider a pair of shortest paths (P_1, P_2) contributing to $H_{v,w,k}(x_1, y_1, x_2, y_2)$. For $i = 1, 2$, decompose $P_i = Q_i \cdot L_i \cdot R_i$, where $Q_i = P_i[x_i, v]$, $L_i = P_i[v, w]$, and $R_i = P_i[w, y_i]$ are shortest paths. The definition of $H_{v,w,k}$, together with the observation that L_1 and L_2 coincide at a minimum of $\delta(v, w)$ vertices, implies that the interaction complexity of Q_2, R_1 is bounded above by $k - \delta(v, w)$. The contributions of the paths Q_2 and R_1 is captured by $F_{\text{disj},k-\delta(v,w)}(w, y_1, x_2, v)$. Meanwhile, the contribution of all the internally disjoint path pairs Q_1, R_2 is given by the polynomial $F_{\text{disj}}(x_1, v, w, y_2)$. Finally, observe that the contribution of L_1, L_2 is given by $F(v, w)^2$ since if $L_1 \neq L_2$, then the pair $(P_1', P_2') = \phi_{v,w}(P_1, P_2)$ will have the same monomial contribution as that of (P_1, P_2), which will cancel modulo 2. Multiplying these components yields the desired expression. ◀

Similar to the above lemma, we can establish the following.

▶ **Lemma 5.4.** *For any integer k, and any $(a, v) \in E$, $w \in V$ such that (a, v) precedes w on some (x_1, y_1) and (x_2, y_2) shortest paths, the following holds:*

$$H_{av,w,k}(x_1, y_1, x_2, y_2) = z_{av}^2 \cdot F_{\text{disj}}(x_1, a, w, y_2) \cdot F(v, w)^2 \cdot F_{\text{disj},k-\delta(v,w)}(w, y_1, x_2, a)$$

▶ **Lemma 5.5.** *For any integer k, and any $v \in V$, $(w, b) \in E$ such that v precedes (w, b) on some (x_1, y_1) and (x_2, y_2) shortest paths, the following holds:*

$$H_{v,wb,k}(x_1, y_1, x_2, y_2) = z_{wb}^2 \cdot F_{\text{disj}}(x_1, v, b, y_2) \cdot F(v, w)^2 \cdot F_{\text{disj},k-\delta(v,w)}(b, y_1, x_2, v)$$

▶ **Lemma 5.6.** *For any integer k, and any $(a, v), (w, b) \in E$ such that (a, v) precedes (w, b) on some (x_1, y_1) and (x_2, y_2) shortest paths, the following holds:*

$$H_{av,wb,k}(x_1, y_1, x_2, y_2) = z_{av}^2 \cdot z_{wb}^2 \cdot F_{\text{disj}}(x_1, a, b, y_2) \cdot F(v, w)^2 \cdot F_{\text{disj},k-\delta(v,w)}(b, y_1, x_2, a)$$

5.3 An $O(m^2n^3)$ time algorithm for Min-2-DSP

In this subsection, we present an algorithm for solving the Min-2-DSP problem in weighted directed graphs that takes $O(m^2n^3)$ time. Let (s_1, t_1) and (s_2, t_2) be the input terminal pairs. By Schwartz-Zippel lemma (Lemma 2.4), in order to check if $\bar{F}_{\text{disj},k}(s_1, t_1, s_2, t_2)$ is non-zero it suffices to evaluate this polynomial on a random assignment of the variables $\{z_e\}_{e \in E}$ drawn from a sufficiently large finite field $\mathbb{F}$. The algorithm outputs the smallest integer k for which $\bar{F}_{\text{disj},k}(s_1, t_1, s_2, t_2)$ evaluates to non-zero at a random assignment of edge variables.

We begin by assigning random values $\bar{z}_e$ from $\mathbb{F}$ to each of the edge variables z_e respectively. As before, for a polynomial R involving the variables $z_e, e \in E$, we shall use $\bar{R}$ to denote the value in $\mathbb{F}$ obtained by evaluating R at $\bar{z}_e, e \in E$. Using Lemma 2.6, we compute $\bar{F}(x, y)$ for all pairs of vertices in $O(mn \log n)$ time, and, in addition, we compute $\delta(x, y)$ for all vertex-pairs using the dominator-tree algorithm of Lengauer and Tarjan [17, 15].

We next evaluate the polynomial $\bar{F}_{\text{disj}}(x_1, x_2, y_1, y_2)$, for each quadruple $(x_1, x_2, y_1, y_2) \in V^4$, in $O(mn^4)$ total time. This is accomplished by leveraging a dynamic programming approach similar to Algorithm 1. We process the quadruples in the non-decreasing order of the sum total distance between the endpoints (i.e., $\text{dist}(x_1, y_1) + \text{dist}(x_2, y_2)$).

At each step, we use recurrence identities established in Lemmas 4.1–4.4. For a fixed (x_1, y_1, x_2, y_2), we compute $\bar{D}_v(x_1, y_1, x_2, y_2)$ for each valid v in $O(\text{indeg}(v))$ time, and combine them to evaluate $\bar{F}_{\text{disj}}(x_1, y_1, x_2, y_2)$ in $O(m)$ time. This yields an overall runtime of $O(mn^4)$. See the detailed in the full version [11].

In order to compute $\bar{F}_{\text{disj},k}(s_1, t_1, s_2, t_2)$, we would need to evaluate $\bar{F}_{\text{disj},k}(x_1, y_1, x_2, y_2)$ for several values of tuples (x_1, y_1, x_2, y_2) and integers k. We compute a list $\mathcal{L}$ containing tuples of the form (v, t_1, s_2, w) for all $v, w \in V$. For each such tuple in $\mathcal{L}$, we evaluate $\bar{F}_{\text{disj},k}(x_1, y_1, x_2, y_2)$ using dynamic programming. These computations are performed in the non-decreasing order of the sum total distance between the endpoints, i.e., $\text{dist}(x_1, y_1) + \text{dist}(x_2, y_2)$, which ensures that all necessary values for the recurrence are available when needed. For base cases, when either $x_1 = y_1$ or $x_2 = y_2$, the values simplify as follows:

1. $\bar{F}_{\text{disj},k}(x_1, x_1, x_2, y_2) = \bar{F}_{\text{disj},k}(x_2, y_2)$, and
2. $\bar{F}_{\text{disj}}(x_1, y_1, x_2, x_2) = \bar{F}(x_1, y_1)$.

To evaluate $\bar{F}_{\text{disj},k}(x_1, y_1, x_2, y_2)$, we use the identity (see Lemma 5.1):

$$\bar{F}_{\text{disj},k}(x_1, y_1, x_2, y_2) = \sum_{v,w \in V} \bar{D}_{v,w,k}(x_1, y_1, x_2, y_2)$$

By Lemma 5.2, for any $\sigma = (x_1, y_1, x_2, y_2)$ and any $v, w \in V$ satisfying $\text{dist}(x_i, y_i) = \text{dist}(x_i, v) + \text{dist}(v, w) + \text{dist}(w, y_i)$, for $i = 1, 2$, $\bar{D}_{v,w,k}$ can be expressed as

$$\bar{D}_{v,w,k}(\sigma) \;=\; \bar{H}_{v,w,k}(\sigma) - \sum_{(a,v)\in E} \bar{H}_{av,w,k}(\sigma) - \sum_{(w,b)\in E} \bar{H}_{v,wb,k}(\sigma) + \sum_{\substack{(a,v),(w,b)\\ \in E}} \bar{H}_{av,wb,k}(\sigma).$$

Now, by Lemmas 5.3–5.6, we have

$$\begin{aligned}
\bar{D}_{v,w,k}(x_1, y_1, x_2, y_2) \;=\;& \bar{F}_{\text{disj}}(x_1, v, w, y_2) \cdot \bar{F}(v, w)^2 \cdot \bar{F}_{\text{disj},k-\delta(v,w)}(w, y_1, x_2, v) \\
&- \sum_{\substack{(a,v)\ \in\\ E(x_1,y_1),\\ E(x_2,y_2)}} z_{av}^2 \cdot \bar{F}_{\text{disj}}(x_1, a, w, y_2) \cdot \bar{F}(v, w)^2 \cdot \bar{F}_{\text{disj},k-\delta(v,w)}(w, y_1, x_2, a) \\
&- \sum_{\substack{(w,b)\ \in\\ E(x_1,y_1),\\ E(x_2,y_2)}} z_{wb}^2 \cdot \bar{F}_{\text{disj}}(x_1, v, b, y_2) \cdot \bar{F}(v, w)^2 \cdot \bar{F}_{\text{disj},k-\delta(v,w)}(b, y_1, x_2, v) \\
&+ \sum_{\substack{(a,v),(w,b)\\ \in E(x_1,y_1),\\ E(x_2,y_2)}} (z_{av} z_{wb})^2 \cdot \bar{F}_{\text{disj}}(x_1, a, b, y_2) \cdot \bar{F}(v, w)^2 \cdot \bar{F}_{\text{disj},k-\delta(v,w)}(b, y_1, x_2, a).
\end{aligned}$$

Note that $\text{dist}(w, y_1) + \text{dist}(x_2, v) < \text{dist}(x_1, y_1) + \text{dist}(x_2, y_2)$ and hence, the right-hand side has been computed already. Finally, we return the smallest integer k for which $\bar{F}_{\text{disj},k}(s_1, t_1, s_2, t_2)$ evaluates to non-zero. See the detailed algorithm in the full version [11].

Correctness and Running Time. The correctness of the procedure follows from the recursive expressions in Lemmas 5.1–5.6.

We now analyze the running time. From the identities above, observe that each polynomial $\bar{D}_{v,w,k}(x_1, y_1, x_2, y_2)$ can be evaluated in $O(\text{indeg}(v) \cdot \text{outdeg}(w))$ time for any fixed v, w, and integer k. Consequently, we can compute $\bar{F}_{\text{disj},k}(x_1, y_1, x_2, y_2)$ in $O(m^2)$ time for any fixed quadruple (x_1, y_1, x_2, y_2) and any fixed k. Since we evaluate these polynomials for $O(n^2)$ such quadruples in the set $\mathcal{L}$ and for n distinct values of k, the total evaluation time is bounded by $O(m^2n^3)$. The algorithm has an initial preprocessing phase, where we first solve All-Pairs 2-DSP in $O(mn^4)$ time, and then compute the number of distance-critical vertices $\delta(x, y)$ for all pairs (x, y) in $O(mn)$ time. The latter uses the dominator-tree algorithm of Lengauer and Tarjan [17, 15], which identifies all cut vertices for pairs in $\{s\} \times V$ in $O(m+n)$ time per source. Altogether, this yields a randomized algorithm for Min-2-DSP with a total running time of $O(m^2n^3)$ and high success probability.

The details of our procedure for reporting the actual (s_1, t_1) and (s_2, t_2) shortest paths with minimum intersection are deferred to the full version, as it is closely related to the procedure outlined in Section 4.4. We conclude with the following theorem.

▶ **Theorem 1.2.** *The Min-2-DSP problem in directed graphs with positive edge weights is solvable in $O(m^2n^3)$ time. Moreover, we can report the explicit paths in $O(m^2n^4)$ time bound.*

6 Open Problems

Our work, together with recent advances on the *2-Disjoint Shortest Paths* problem [4, 6, 7], opens up a plethora of exciting new directions for the 2-DSP problem (see also Section 8.7 of [2] for further related open problems). A first challenge is to improve upon the current $O(mn \log n)$ time algorithm for 2-DSP on general directed graphs? If not, can one obtain conditional lower bounds ruling out such improvements?

▶ **Open Problem 1.** *Is it possible to solve 2-DSP in $o(mn)$ time on general directed weighted graphs with no cycles of negative or zero weight? If not, can one prove fine-grained lower bounds ruling out such an improvement?*

It is important to note that due to the algebraic nature of the results of [4] and our work, the time complexities primarily discussed are for *detecting* the existence of disjoint shortest paths. Returning an explicit pair of such paths, when they exist, incurs an additional multiplicative factor of n in the running time [4]. An interesting question to answer is whether the two disjoint shortest paths can be reported in the same time as detecting such paths.

▶ **Open Problem 2** ([4]). *In DAGs and undirected graphs, does there exist an $O(m + n)$ time algorithm to report the two disjoint shortest paths, if they exist? For general directed graphs, does there exist an $O(mn \log n)$ algorithm to report such paths?*

Due to recent breakthroughs of [6, 18], it has been established that for any constant $k \geqslant 1$, k-DSP can be solved in polynomial time in undirected graphs. It is not too hard to obtain such guarantees for DAGs either [4, 7, 13]. However, for general directed graphs, it is not even known whether 3-DSP can be solved in polynomial time (no hardness results are known either). Hence, it is worthwhile to ask if our techniques can be used to obtain an efficient algorithm for 3-DSP.

▶ **Open Problem 3.** *In general directed graphs, can 3-DSP be solved in polynomial time? Alternatively, can one establish meaningful hardness results for 3-DSP in this setting?*

Another promising direction is to extend the k-DSP to the newly introduced Min-2-DSP problem. In particular, can one solve Min-k-DSP in undirected graphs in polynomial time for every constant $k \geqslant 3$? An affirmative answer would yield a natural counterpart to the existing k-DSP results on undirected graphs, whereas a negative answer would provide an interesting separation between the k-DSP and Min-k-DSP problems.

▶ **Open Problem 4.** *In undirected graphs, can Min-k-DSP be solved in polynomial time for every fixed $k \geqslant 3$? Alternatively, can one prove hardness results for Min-k-DSP in this regime?*

Lastly, the algebraic algorithms described in [4] and this work inherently make use of randomization, as they invoke the Schwartz-Zippel lemma. It be would be interesting to obtain deterministic algorithms for 2-DSP and Min-2-DSP, with running time comparable to the randomized algorithms of [4] and this work. We believe this will provide us with deeper structural insights into the problems.

▶ **Open Problem 5.** *Can one obtain deterministic algorithms for 2-DSP and Min-2-DSP, which match the time bounds guaranteed by the current fastest randomized algorithms?*

References

1. Maxim Akhmedov. Faster 2-disjoint-shortest-paths algorithm. In Henning Fernau, editor, *Computer Science - Theory and Applications - 15th International Computer Science Symposium in Russia, CSR 2020, Yekaterinburg, Russia, June 29 - July 3, 2020, Proceedings*, volume 12159 of *Lecture Notes in Computer Science*, pages 103–116. Springer, 2020. doi:10.1007/978-3-030-50026-9_7.
2. Shyan Akmal. *Parameterized Relaxations for Circuits and Graphs*. Ph.d. thesis, Massachusetts Institute of Technology, Cambridge, MA, May 2024. URL: https://dspace.mit.edu/handle/1721.1/156299.
3. Shyan Akmal, Virginia Vassilevska Williams, and Nicole Wein. Detecting disjoint shortest paths in linear time and more. *CoRR*, abs/2404.15916, 2024. doi:10.48550/arXiv.2404.15916.
4. Shyan Akmal, Virginia Vassilevska Williams, and Nicole Wein. Detecting Disjoint Shortest Paths in Linear Time and More. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2024.9.
5. Matthias Bentert, Fedor V. Fomin, and Petr A. Golovach. Tight Approximation and Kernelization Bounds for Vertex-Disjoint Shortest Paths. In Olaf Beyersdorff, Michał Pilipczuk, Elaine Pimentel, and Nguyen Kim Thang, editors, *42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025)*, volume 327 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:17, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.STACS.2025.17.
6. Matthias Bentert, André Nichterlein, Malte Renken, and Philipp Zschoche. Using a geometric lens to find k disjoint shortest paths. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 26:1–26:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.26.
7. Kristof Berczi and Yusuke Kobayashi. The Directed Disjoint Shortest Paths Problem. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms (ESA 2017)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2017.13.
8. Andreas Björklund and Thore Husfeldt. Shortest two disjoint paths in polynomial time. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming*, pages 211–222, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. doi:10.1007/978-3-662-43948-7_18.
9. Andreas Björklund, Thore Husfeldt, and Petteri Kaski. The shortest even cycle problem is tractable. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 117–130. ACM, 2022. doi:10.1145/3519935.3520030.
10. Rajesh Chitnis, Samuel Thomas, and Anthony Wirth. Lower bounds for approximate (& exact) k-disjoint-shortest-paths, 2024. doi:10.48550/arXiv.2408.03933.
11. Keerti Choudhary, Amit Kumar, and Lakshay Saggi. Efficient algorithms for disjoint shortest paths problem and its extensions, 2025. doi:10.48550/arXiv.2509.14588.
12. Tali Eilam-Tzoreff. The disjoint shortest paths problem. *Discrete Applied Mathematics*, 85(2):113–138, June 1998. doi:10.1016/s0166-218x(97)00121-2.
13. Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980. doi:10.1016/0304-3975(80)90003-9.

14 Ryo Funayama, Yasuaki Kobayashi, and Takeaki Uno. Parameterized complexity of finding dissimilar shortest paths, 2024. doi:10.48550/arXiv.2402.14376.

15 Loukas Georgiadis and Robert Endre Tarjan. Dominator tree verification and vertex-disjoint paths. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 433–442. SIAM, 2005. URL: http://dl.acm.org/citation.cfm?id=1070432.1070492.

16 Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1):1–13, January 1977. doi:10.1145/321992.321993.

17 Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Trans. Program. Lang. Syst.*, 1(1):121–141, 1979. doi:10.1145/357062.357071.

18 William Lochet. A polynomial time algorithm for the k-disjoint shortest paths problem. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 169–178. SIAM, 2021. doi:10.1137/1.9781611976465.12.

19 Mathieu Mari, Anish Mukherjee, Michał Pilipczuk, and Piotr Sankowski. *Shortest Disjoint Paths on a Grid*, pages 346–365. SIAM, 2024. doi:10.1137/1.9781611977912.14.

20 R.G. Ogier, V. Rutenburg, and N. Shacham. Distributed algorithms for computing shortest pairs of disjoint paths. *IEEE Transactions on Information Theory*, 39(2):443–455, 1993. doi:10.1109/18.212275.

21 Michał Pilipczuk, Giannos Stamoulis, and Michał Włodarczyk. Planar disjoint shortest paths is fixed-parameter tractable, 2025. doi:10.48550/arXiv.2505.03353.

22 Anand Srinivas and Eytan Modiano. Finding minimum energy disjoint paths in wireless ad-hoc networks. *Wireless Networks*, 11(4):401–417, July 2005. doi:10.1007/s11276-005-1765-0.

23 Clark David Thompson. *A complexity theory for VLSI*. PhD thesis, Carnegie Mellon University, USA, 1980. AAI8100621.