A Local-To-Global Theorem for Congested Shortest Paths

Shyan Akmal
MIT, EECS and CSAIL, Cambridge, MA, USA

Nicole Wein
DIMACS, Rutgers University, Piscataway, NJ, USA

Abstract

Amiri and Wargalla proved the following local-to-global theorem about shortest paths in directed acyclic graphs (DAGs): if G is a weighted DAG with the property that for each subset S of 3 nodes there is a shortest path containing every node in S, then there exists a pair (s, t) of nodes such that there is a shortest st-path containing every node in G. We extend this theorem to general graphs. For undirected graphs, we prove that the same theorem holds (up to a difference in the constant 3). For directed graphs, we provide a counterexample to the theorem (for any constant). However, we prove a *roundtrip* analogue of the theorem which guarantees there exists a pair (s, t) of nodes such that every node in G is contained in the union of a shortest st-path and a shortest ts-path.

The original local-to-global theorem for DAGs has an application to the k-Shortest Paths with Congestion c ((k, c)-SPC) problem. In this problem, we are given a weighted graph G, together with k node pairs $(s_1, t_1), \ldots, (s_k, t_k)$, and a positive integer $c \leq k$, and tasked with finding a collection of paths $P_1, \ldots, P_k$ such that each P_i is a shortest path from s_i to t_i, and every node in the graph is on at most c paths P_i, or reporting that no such collection of paths exists. When $c = k$, there are no congestion constraints, and the problem can be solved easily by running a shortest path algorithm for each pair (s_i, t_i) independently. At the other extreme, when $c = 1$, the (k, c)-SPC problem is equivalent to the k-Disjoint Shortest Paths (k-DSP) problem, where the collection of shortest paths must be node-disjoint. For fixed k, k-DSP is polynomial-time solvable on DAGs and undirected graphs. Amiri and Wargalla interpolated between these two extreme values of c, to obtain an algorithm for (k, c)-SPC on DAGs that runs in polynomial time when $k - c$ is constant.

In the same way, we prove that (k, c)-SPC can be solved in polynomial time on undirected graphs whenever $k - c$ is constant. For directed graphs, because of our counterexample to the original theorem statement, our roundtrip local-to-global result does not imply such an algorithm (k, c)-SPC. Even without an algorithmic application, our proof for directed graphs may be of broader interest because it characterizes intriguing structural properties of shortest paths in directed graphs.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases disjoint paths, shortest paths, congestion, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.8

Related Version *Full Version*: https://arxiv.org/abs/2211.07042

Funding *Shyan Akmal*: Supported in part by NSF grant CCF-2129139.
Nicole Wein: Supported by a grant to DIMACS from the Simons Foundation (820931).

Acknowledgements We thank Virginia Vassilevska Williams for helpful discussions about this work.

1 Introduction

An intriguing question in graph theory and algorithms is: "can we understand the structure of shortest paths in (directed and undirected) graphs?" More specifically: "can we understand the structure of the *interactions* between shortest paths in graphs?" This question has been approached from various angles in the literature. For instance, Bodwin [12] characterizes which sets of nodes can be realized as *unique* shortest paths in weighted graphs, and Cizma

and Linial investigate the properties of graphs whose shortest paths satisfy or violate certain geometric properties [19, 20]. Additionally, there is a large body of work on distance preservers, where the goal is to construct a subgraph that preserves distances in the original graph (see [11] and the references therein). There are also numerous computational tasks in which structural results about shortest paths inform the creation and analysis of algorithms, including distance oracle construction [15], the k-disjoint shortest paths problem, [8], and the next-to-shortest path problem [35] (the cited papers are the most recent publications in their respective topics).

The angle of our work is inspired by the following local-to-global theorem[1] of Amiri and Wargalla [3] concerning the structure of shortest paths in directed acyclic graphs (DAGs):

▶ **Theorem 1** ([3]). *Let G be a weighted DAG with the property that, for each set S of 3 nodes, there is a shortest path containing every node in S. Then, there exists a pair (s, t) of nodes such that there is a shortest st-path containing every node in G.*

Note that in the statement of Theorem 1, the condition that "for each set S of 3 nodes, there is a shortest path containing every node in S" is equivalent to the condition that one of the 3 nodes is on a shortest path between the other two.

Theorem 1 is "local-to-global" in the sense that from a highly congested local structure (every small subset of nodes is contained in a shortest path) we deduce a global structure (all nodes in the graph live on a single shortest path).

At first glance, Theorem 1 may appear rather specialized, since the existence of shortest paths through *all* triples of nodes is a rather strong condition. However, Amiri and Wargalla [3] show that Theorem 1 has applications to the k-Shortest Paths with Congestion c ((k, c)-SPC) on problem on DAGs: in this problem we are given a DAG, and are tasked with finding a collection of shortest paths between k given source/sink pairs, such that each node in the graph is on at most c of the paths. We discuss this application in detail in Section 1.2.

Amiri and Wargalla [3] raised the question of whether their results can be extended to general graphs, both undirected and directed. Our work answers this question.

1.1 Structural Results

We ask the following question:

Is Theorem 1 true for general (undirected and directed) graphs?

Our first result answers this question affirmatively for undirected graphs (with constant 4 instead of 3).

▶ **Theorem 2** (Undirected graphs). *Let G be a weighted undirected graph with the property that, for each set S of 4 nodes, there is a shortest path containing every node in S. Then, there exists a pair (s, t) of nodes such that some shortest st-path contains every node in G.*

The constant 4 in the statement of Theorem 2 cannot be replaced with 3, as seen by considering an undirected cycle on four vertices.

Theorem 2 implies a faster algorithm for (k, c)-SPC on *undirected* graphs, answering an open question raised in [3, Section 4]. We discuss the background of this problem and our improvement in Section 1.2.

[1] This is slightly different from the statement of the theorem in [3], where the authors write present their result in the context of the Shortest Paths with Congestion problem, which we discuss in detail later.

Our second result is for directed graphs. First, we observe that there is actually a *counterexample* to Theorem 1 for general directed graphs: let a be the constant for which we desire a counterexample (i.e., the constant that is 3 in Theorem 1, and 4 in Theorem 2). Let G be a cycle with bidirectional edges, where all clockwise-pointing edges have weight 1 and all counterclockwise-pointing edges have weight a. One can verify that the precondition of the theorem holds: for each set S of a nodes there is a shortest path containing every node in S, just by taking the shortest clockwise path through all nodes in S. However, no single shortest path contains every node in G, so the theorem does not hold.

Even though a direct attempt at generalizing Theorem 1 to all directed graphs fails, one might hope for some analogue of Theorem 1 that does hold. One interpretation of the above counterexample is that the exact statement of Theorem 1 is not the "right" framework for getting a local-to-global shortest path phenomenon in general directed graphs. To that end, we consider the *roundtrip* analogue of Theorem 1, where the final path through every node is a shortest roundtrip path, i.e., the union of a shortest st-path and a shortest ts-path (roundtrip distances are a common object of study in directed graphs, with there being much research, for example, in roundtrip routing [21], roundtrip spanners [33], and roundtrip diameter computation [1]). Note that the above counterexample does not apply to the roundtrip analogue of Theorem 1 since there exists a pair (s, t) of nodes such that the union of a shortest st-path and a shortest ts-path are both in the clockwise direction and thus contain all nodes in the graph.

For our second result, we present a roundtrip analogue of Theorem 1 which holds true for general directed graphs (with the constant 11 instead of 3):

▶ **Theorem 3** (Directed graphs). *Let G be a weighted directed graph with the property that, for each set S of 11 nodes, there is a shortest path containing every node in S. Then, there exists a pair (s, t) of nodes such that the union of a shortest st-path and a shortest ts-path contains every node in G.*

Proving Theorem 3 requires overcoming a number of technical challenges involving the c omplex structure of shortest paths in directed graphs. Due to its roundtrip nature, unlike Theorem 2, Theorem 3 does not appear to have any immediate algorithmic applications.

1.2 Disjoint and Congested Shortest Path Problems

In this section we introduce the k-Shortest Paths with Congestion c ((k, c)-SPC) problem and state the implications of our work for this problem.

1.2.1 Background

We begin by discussing the related k-Disjoint Shortest Paths (k-DSP) problem. For more related work on disjoint path problems in general, see the full version.

Disjoint Shortest Paths

Formally, the k-Disjoint Shortest Paths (k-DSP) problem is defined as follows:

> k-Disjoint Shortest Paths (k-DSP): Given a graph G and k node pairs $(s_1, t_1), \ldots, (s_k, t_k)$, find a collection of node-disjoint paths $P_1, \ldots, P_k$ such that each P_i is a shortest path from s_i to t_i, or report that no such collection of paths exists.

The k-DSP problem was introduced in the 90s by Eilam-Tzoreff [22], who gave a polynomial-time algorithm for undirected graphs when $k = 2$, and conjectured that there is a polynomial-time algorithm for any fixed k in both undirected and directed graphs. Recently, Lochet [30] proved Eilam-Tzoreff's conjecture for undirected graphs by showing that k-DSP can be solved in polynomial time for any fixed k. Subsequently, the dependence on k in the running time was improved by Bentert, Nichterlein, Renken, and Zschoche [8]. It is known that k-DSP on undirected graphs is W[1]-hard [8, Proposition 36], so this problem is unlikely to be fixed-parameter tractable.

For directed graphs, Bérczi and Kobayashi [9] showed that 2-DSP can be solved in polynomial time. For $k \geq 3$ however, determining the complexity of k-DSP on directed graphs remains a major open problem. This problem is only known to be polynomial-time solvable for special classes of directed graphs, such as DAGs and planar graphs [9].

Shortest Paths with Congestion

The k-Shortest Paths with Congestion c ((k, c)-SPC) problem is the variant of k-DSP where some amount of *congestion* (paths overlapping at nodes) is allowed. In general, problems of finding paths with limited congestion are well-studied studied in both theory and practice. For instance, there is much work on the problem in undirected graphs where the goal is to find a maximum cardinality subset of node pairs (s_i, t_i) that admit (not necessarily shortest) paths with congestion at most c [32, 28, 7, 6, 5, 17, 4, 26, 16, 18]. As another example, [27] provides, for fixed k, a polynomial-time algorithm for the problem on directed graphs of determining that either there is no set of disjoint paths between the node pairs (s_i, t_i), or finding a set of such paths with congestion at most 4. Another example for directed graphs is the problem of finding paths between the node pairs (s_i, t_i) where only some nodes in the graph have a congestion constraint [31].

Formally, the k-Shortest Paths with Congestion c ((k, c)-SPC) problem is defined as follows:

> k-Shortest Paths with Congestion c ((k, c)-SPC): Given a graph G, along with k node pairs $(s_1, t_1), \ldots, (s_k, t_k)$, and a positive integer $c \leq k$, find a collection of paths $P_1, \ldots, P_k$ such that each P_i is a shortest path from s_i to t_i, and every node in $V(G)$ is on at most c paths P_i, or report that no such collection of paths exists.

The (k, c)-SPC problem was introduced by Amiri and Wargalla [3]. Before that, the version of (k, c)-SPC where the paths are not restricted to be shortest paths was studied by Amiri, Kreutzer, Marx, and Rabinovich [2].

When $c = 1$, the (k, c)-SPC problem is equivalent to the k-DSP problem. At the other extreme, when $c = k$, there are no congestion constraints, so the problem can be easily solved in polynomial time by simply finding a shortest path for each pair (s_i, t_i) independently. Amiri and Wargalla [3] asked the following question: *can we interpolate between these two extremes?* In particular, can we get algorithms for (k, c)-SPC where the exponential dependence on k for k-DSP can be replaced with some dependence on $O(k - c)$ instead?

Amiri and Wargalla [3] achieved this goal for DAGs. In particular, they gave a reduction from (k, c)-SPC on DAGs to k-DSP on DAGs of the following form: letting $d = k - c$, if k-DSP on DAGs can be solved in time $f(n, k)$, then (k, c)-SPC on DAGs can be solved in time $O\left(\binom{k}{3d} \cdot f(2dn, 3d)\right)$. The essential aspect of this running time is that the second input to the function f is not k but rather an $O(d)$ term. A key tool in their reduction is Theorem 1 (stated in a different way).

Since k-DSP can be solved in $n^{O(k)}$ time on DAGs [9], the above result implies that (k, c)-SPC can be solved in $\binom{k}{3d} \cdot (2dn)^{O(d)}$ time on DAGs. That is, (k, c)-SPC on DAGs is polynomial-time solvable for arbitrary k whenever d is constant. We note that for for every c, the (k, c)-SPC problem on DAGs is W[1]-hard with respect to d, so the problem is unlikely to be fixed-parameter tractable with respect to d [3, Proof of Theorem 3].

1.2.2 Algorithmic Results

Similarly to Amiri and Wargalla's result for DAGs, our result for undirected graphs, Theorem 2, implies a reduction from (k, c)-SPC to k-DSP on undirected graphs.

▶ **Lemma 4.** *If k-DSP can be solved in $f(n, k)$ time on undirected graphs, then (k, c)-SPC can be solved in $O\left(\binom{k}{4d} \cdot f(3dn, 4d)\right)$ time on undirected graphs.*

Lemma 4 follows from Theorem 2 using an argument nearly identical to the one presented for DAGs in [3] (up to a difference in constants). For completeness, we include a full proof of this result in the full version.

Since it is known that k-DSP can be solved in undirected graphs in time $n^{O(k \cdot k!)}$ [8], applying Theorem 2 together with Lemma 4, we deduce the following result.

▶ **Corollary 5.** *(k, c)-SPC on undirected graphs can be solved in $\binom{k}{4d} \cdot (3dn)^{O(d \cdot (4d)!)}$ time.*

Thus (k, c)-SPC on undirected graphs is in polynomial time whenever $d = k - c$ is constant; that is, it is in the complexity class XP with respect to the parameter d. Prior to our work, no polynomial-time algorithm for this problem appears to have been known in this regime, even for simple cases such as $(k, k-1)$-SPC on undirected graphs.

In contrast, our structural result for directed graphs, Theorem 3, does not imply a faster algorithm for (k, c)-SPC in directed graphs. This is because Theorem 3 does not appear to imply a reduction from (k, c)-SPC to k-DSP in the manner of Lemma 4. Moreover, even if such a reduction did exist, this would not imply an algorithm for directed graphs analogous to Corollary 5. This is because while k-DSP is polynomial-time solvable for constant k in undirected graphs, it remains open whether even 3-DSP over directed graphs can be solved in polynomial time.

1.3 The Structure of Shortest Paths in Directed Graphs

Although our result for directed graphs, Theorem 3, does not appear to have immediate algorithmic applications, we believe it is still interesting from a graph theoretic perspective, especially in light of the current scarcity of results for shortest disjoint path problems in directed graphs. In this section, we expand upon this idea with some remarks, and then state a lemma from our proof concerning the structure of shortest paths in directed graphs.

We currently have a poor understanding of the complexity of the k-DSP problem in directed graphs: for fixed $k \geq 3$, it is still not known if this problem is either polynomial-time solvable or NP-hard. In fact, even the complexity of the seemingly easier $(3, 2)$-SPC problem on directed graphs is open. In this context, Theorem 3 is compelling because it presents an example of interesting behavior which holds for collections of shortest paths in DAGs, and then continues to hold, under suitable generalization, for systems of shortest paths in general directed graphs. This sort of characterization appears to be rare in the literature.

More generally, the methods we use to establish Theorem 3 involve combinatorial observations about the structure of shortest paths in directed graphs, and the interactions between them. We believe our analysis could offer more insight into resolving other problems that

concern systems of shortest paths in directed graphs. There are many such problems, where the undirected case is well-understood, but in the directed case not much is known. This barrier is in-part due to the relatively complex patterns of shortest paths which can appear in directed graphs. We hope that our analysis of directed shortest paths may shed light on problems for which the structural complexity of directed shortest paths is the bottleneck towards progress.

One example of a problem where the undirected case is well-understood while the directed case remains poorly understood is, as discussed previously, the k-DSP problem. Another curious example is the Not-Shortest Path problem, where the goal is to find an st-path that is not a shortest path. Although Not-Shortest Path can be solved over undirected graphs in polynomial time [29], no polynomial time algorithm is known for this problem in directed graphs. A third example is the problem of approximating the diameter of a graph. For undirected graphs there is an infinite hierarchy of algorithms that trade off between time and accuracy [14], while only two points on the hierarchy are known for directed graphs. Additionally, the roundtrip variant of diameter is the least understood of any studied variant of the diameter problem [34]. Another example is the construction of approximate hopsets: for directed graphs there is there a polynomial gap between upper and lower bounds [25, 10, 13], while for undirected graphs the gap is subpolynomial [23, 24]. The preponderance of such examples motivates proving results like Theorem 3, which characterize interesting behavior of shortest paths in directed graphs.

Structural Lemma for Directed Shortest Paths

One of the lemmas we establish on the way to proving Theorem 3 is a general statement about the structure of shortest paths in directed graphs. It can be stated independently of the context of the proof of Theorem 3 and we highlight it here.

We categorize any shortest path P into one of two main types, based on the ways that other shortest paths intersect with it. The following simple definition will be useful for defining our path types.

▶ **Definition 6.** *For any nonnegative integer ℓ and set of ℓ nodes, $v_1, v_2, \ldots, v_\ell$, we say that the order $v_1 \to v_2 \to \cdots \to v_\ell$ is a* shortest-path ordering *if there is a shortest path containing all of the nodes $v_1, v_2, \ldots, v_\ell$ in that order.*

In addition to our two main path types, there is a third path type which we call a *trivial path* because it is easy to handle:

▶ **Definition 7** (Trivial Path)**.** *Given a directed graph, nodes a and b, and a shortest path P from a to b, we say P is a* trivial path *if P contains at least one node w such that $a \to w \to b$ is the* only *shortest-path ordering of a, w, b.*

Now we are ready to state our two main types of shortest paths. The first type is a *reversing path*:

▶ **Definition 8** (Reversing path)**.** *Given a directed graph, nodes a and b, and a non-trivial shortest path P from a to b, P is* reversing *if P contains at least one node w such that w falls on some shortest path from b to a. A* non-reversing path *is a non-trivial path that is not reversing.*

We prove a lemma that characterizes the structure of reversing and non-reversing paths in terms of the possible shortest-path orderings of each node on the path and the endpoints of the path. See Figures 1a and 1b for a depiction of the structure enforced by the lemma.

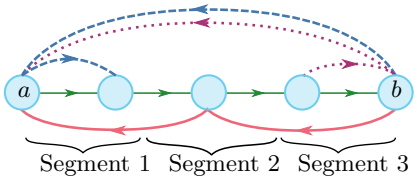
▶ **Lemma 9** (Reversing/Non-Reversing Lemma). *Let P be a non-trivial shortest path and let a and b be the first and last nodes of P respectively. Then P can be partitioned into three contiguous ordered segments with the following properties (where a is defined to be in Segment 1, b is defined to be in Segment 3, and Segment 2 could be empty).*

Segment 1 consists of nodes w such that the shortest-path orderings of a, w, b are precisely $a \to w \to b$, and $b \to a \to w$.

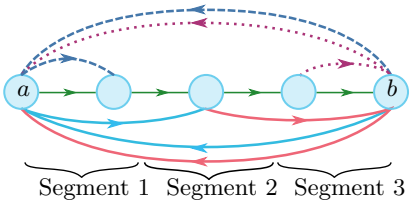
Segment 2 consists of nodes w such that the shortest-path orderings of a, w, b are precisely

$$\begin{cases} a \to w \to b, \text{ and } b \to w \to a & \text{if } P \text{ is a reversing path} \\ a \to w \to b, \text{ and } b \to a \to w, \text{ and } w \to b \to a & \text{if } P \text{ is a non-reversing path.} \end{cases}$$

Segment 3 consists nodes w such that the shortest-path orderings of a, w, b are precisely $a \to w \to b$, and $w \to b \to a$.



(a) Reversing Path. The structure of a reversing path, as given by Lemma 9. The blue dashed path, pink solid path, and purple dotted path are examples of the allowed orderings for nodes in segments 1, 2, and 3 respectively.



(b) Non-Reversing Path. The structure of a non-reversing path, as given by Lemma 9. The possible shortest-path orderings for nodes in segment 2 are represented by pink and light blue solid paths, while the orderings allowed for nodes in segments 1 and 3 are represented by blue dashed and purple dotted segments respectively.

Figure 1 Possible orderings of vertices on shortest paths in the reversing and non-reversing cases. The blue circles are representative examples of the types of nodes on the path from a to b (in general this path will contain more than just five nodes).

In the proof of Theorem 3 we employ the strategy of categorizing shortest paths as reversing or non-reversing (or trivial), and applying Lemma 9 to glean some structure. We note, however, that Lemma 9 itself is not the main technical piece of the proof.

2 Preliminaries

All graphs are assumed to have positive edge weights. Graphs are either undirected or directed, depending on the section. For any pair of nodes (u, v), we use $\mathsf{dist}(u, v)$ to denote the shortest path distance from u to v. Given a path P and two nodes u and v occurring on P in that order, we let $P[u, v]$ denote the subpath of P with u and v as endpoints.

For the (k, c)-SPC problem, we always let d denote the difference $d = k - c$. When considering a particular solution to a (k, c)-SPC instance, we refer to the paths $P_1, \ldots, P_k$ between $(s_1, t_1), \ldots, (s_k, t_k)$ respectively as *solution paths*. Any node in the graph which lies in c of the solution paths is referred to as a *max-congestion* node.

2.1 Subpath Swapping

In our proofs, we will frequently modify collections of shortest paths by "swapping subpaths" between intersecting paths. This procedure is depicted in Figure 2, and we formally describe it below.

▶ **Definition 10** (Subpath Swap). *Let $\mathcal{R}$ be a collection of shortest paths in a directed graph. Let P and Q be two paths in $\mathcal{R}$. Let $a, b \in P \cap Q$ be nodes in these paths, such that a appears before b in both P and Q. We define* swapping *the subpaths of P and Q between a and b to be updating the set of paths $\mathcal{R}$ by simultaneously replacing P with $(P \setminus P[a,b]) \cup Q[a,b]$ and Q with $(Q \setminus Q[a,b]) \cup P[a,b]$. We often refer to this process simply as "swapping $P[a,b]$ and $Q[a,b]$."*

Figure 2 A simple subpath swap, where the subpaths from a to b of the green path (from s_i to t_i) and pink path (from s_j to t_j) are switched.

▶ **Observation 11** (Subpath Swap). *Let $\mathcal{R}$ be the solution to some (k,c)-SPC problem. Then swapping subpaths in $\mathcal{R}$ produces a new solution to the same (k,c)-SPC instance with the same set of max-congestion nodes.*

Proof. This observation holds because swapping subpaths does not change the number of solution paths any given node is contained in, does not change the endpoints of any solution path, and all solution paths remain shortest paths. ◀

2.2 Correspondence Between Our Results and (k, c)-SPC

In this section we detail some nuances regarding the correspondence between the statement of our results and the (k,c)-SPC problem. For the sake of generality and simplicity, we stated Theorems 1-3 independently of the (k,c)-SPC problem. In contrast, the original result of Amiri and Wargalla, corresponding to Theorem 1, was stated as follows:

▶ **Lemma 12** ([3]). *If $k > 3d$, then any instance of (k,c)-SPC on DAGs either has no solution, or has a solution where some solution path P_i passes through all max-congestion nodes.*

Although the statement of Lemma 12 may initially seem unrelated to the statement of Theorem 1, their correspondence becomes clearer with the following observation, which is a simple generalization of an observation from [3]:

▶ **Observation 13.** *Let N be a positive integer. Suppose $k > Nd$, and let $\mathcal{R}$ be a solution to an arbitrary (k,c)-SPC instance. Then for any set S of N max-congestion nodes in $\mathcal{R}$, there exists some solution path in $\mathcal{R}$ which contains every node in S.*

We defer the proof of Observation 13 to the appendix.

To prove our results for the (k,c)-SPC problem in undirected graphs (Lemma 4 and Corollary 5), we need to prove a lemma analogous to Lemma 12 but for undirected graphs:

▶ **Lemma 14.** *If $k > 4d$, then any instance of (k, c)-SPC either has no solution, or has a solution where some solution path P_i passes through all max-congestion nodes.*

The following generalizes Theorem 2 and Lemma 14.

▶ **Lemma 15** (General Undirected Result)**.** *Given an undirected graph and subset W of nodes, let $\mathcal{R}$ be a collection of shortest paths with the following property:*

$$\text{for every set } S \subseteq W \text{ of 4 nodes, some path in } \mathcal{R} \text{ contains every node in } S. \qquad (\star)$$

Further suppose that applying any sequence of $O(n^3)$ subpath swaps to $\mathcal{R}$ yields a collection of shortest paths that still has property $(\star)$. Then starting from $\mathcal{R}$, there exists a sequence of subpath swaps that results in a collection of shortest paths in which some path P passes through all nodes in W.

We note that the quantity $O(n^3)$ is an unimportant technicality used in the proof of the following observation, and is chosen as a loose upper bound on the number of subpath swaps we will perform.

▶ **Observation 16.** *Lemma 15 generalizes both Theorem 2 and Lemma 14.*

We defer the proof of Observation 16 to the appendix.

We also prove an analogue of Lemma 12 and Lemma 15 for directed graphs:

▶ **Lemma 17** (General Directed Result)**.** *Given a directed graph and subset W of nodes, let $\mathcal{R}$ be a collection of shortest paths with the following property:*

$$\text{for every set } S \subseteq W \text{ of 11 nodes, some path in } \mathcal{R} \text{ contains every node in } S. \qquad (\dagger)$$

Further suppose that applying any sequence of $O(n^3)$ subpath swaps to $\mathcal{R}$ yields a collection of paths that still has property $(\dagger)$. Then starting from $\mathcal{R}$, there exists a sequence of subpath swaps that results in a collection of shortest paths in which either:

1. *some path P passes through all nodes in W, or*
2. *the union of two paths P, P' contain all nodes in W, and the first and last nodes on P from W are the same as the last and first nodes on P' from W, respectively.*

Lemma 17 generalizes Theorem 3, in exactly the same way as Lemma 15 generalizes Theorem 2 for undirected graphs. In the same way that Lemma 15 generalizes Lemma 14 for undirected graphs, Lemma 17 implies the following lemma:

▶ **Lemma 18.** *If $k > 11d$, then any instance of (k, c)-SPC on directed graphs either has no solution, or has a solution where the union of some two solution paths P_i and P_j contains all max-congestion nodes.*

Although Lemma 18 does not immediately lead to an algorithm for (k, c)-SPC on directed graphs, it specifies some structure which may be useful for future work on (k, c)-SPC and related problems.

One might wonder whether Lemma 18 can be modified to have only one solution path P_i that contains all max-congestion nodes, like for DAGs (Lemma 12) and undirected graphs (Lemma 14), since this would imply interesting algorithms for (k, c)-SPC on directed graphs. Unfortunately, the answer to this question turns out to be no. Similar to the counterexample against extending Theorem 2 to directed graphs, we present a counterexample in the full version which rules out replacing two solution paths with a single solution path in Lemma 18.

3 Technical Overview

3.1 Prior Work on DAGs

As a starting point for our proofs for general graphs, we use Amiri and Wargalla's proof of Theorem 1 for DAGs [3]. We describe their proof differently than they do for the sake of comparison to our work. We actually describe their proof of the following lemma (which they do not explicitly state), which is analogous to the statements of our general results (Lemmas 15 and 17).

▶ **Lemma 19** ([3]). *Given a DAG and subset W of nodes, let $\mathcal{R}$ be a collection of shortest paths with the following property:*

$$\text{for every set } S \subseteq W \text{ of 3 nodes, some path in } \mathcal{R} \text{ contains every node in } S. \qquad (*)$$

Further suppose that applying any sequence of $O(n^3)$ subpath swaps to $\mathcal{R}$ yields a collection of paths that still has property $(*)$. Then starting from $\mathcal{R}$, there exists a sequence of subpath swaps that results in a collection of shortest paths in which some path P passes through all nodes in W.

The proof of Lemma 19 is as follows. Let a and b be the first and last nodes in W (respectively) in a topological ordering of the DAG. Let $v_1, \dots v_{|W|-2}$ be the remaining nodes in W in order topologically. For ease of notation, we consider $\mathcal{R}$ to be changing over time via subpath swaps, and we will let $\mathcal{R}$ denote the current value of $\mathcal{R}$.

The argument is inductive. For the base case, by property $(*)$ there is a path in $\mathcal{R}$ that contains a and b. Suppose inductively that a path $P \in \mathcal{R}$ currently contains a, b, and $v_1, \dots, v_\ell$ for some ℓ. We would like to perform a subpath swap to augment P by adding $v_{\ell+1}$ to P. To do so, we consider a path $P' \in \mathcal{R}$ that contains v_ℓ, $v_{\ell+1}$, and b, where such a path exists by property $(*)$. Importantly, because the graph is a DAG, v_ℓ, $v_{\ell+1}$, and b appear in that order on both P and P'. Thus, according to the definition of a subpath swap (Definition 10) we can swap $P[v_\ell, b]$ with $P'[v_\ell, b]$, as shown in Figure 3. As a result of this subpath swap, P now contains $v_{\ell+1}$ in addition to all of the nodes in W that P originally contained. By induction, this completes the proof.

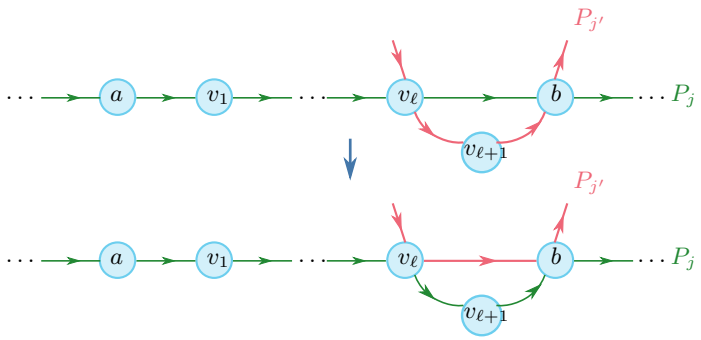


Figure 3 In a DAG, the topological ordering of the nodes allows us to perform a sequence of subpath swaps, each adding the next node in W in order to a path P.

3.2 Undirected Graphs

Recall that our goal for undirected graphs is to prove the following theorem:

▶ **Lemma 15** (General Undirected Result). *Given an undirected graph and subset W of nodes, let $\mathcal{R}$ be a collection of shortest paths with the following property:*

$$\text{for every set } S \subseteq W \text{ of 4 nodes, some path in } \mathcal{R} \text{ contains every node in } S. \qquad (\star)$$

Further suppose that applying any sequence of $O(n^3)$ subpath swaps to $\mathcal{R}$ yields a collection of shortest paths that still has property $(\star)$. Then starting from $\mathcal{R}$, there exists a sequence of subpath swaps that results in a collection of shortest paths in which some path P passes through all nodes in W.

The essential property that enables the subpath swapping in the above argument for DAGs is the fact that for any triple of nodes in a DAG, there is *only one* possible order this triple can appear on any path. This property is not exactly true for general undirected graphs, but we observe that a similar "consistent ordering" property is true: if there is a *shortest* path containing nodes u, v, w in that order, then any shortest path containing these nodes, has them in that order (or in the reverse order w, v, u, but since the graph is undirected we can without loss of generality assume they are in the order u, v, w). This property is true simply because $\mathsf{dist}(u, w)$ is larger than both $\mathsf{dist}(u, v)$ and $\mathsf{dist}(v, w)$, so v must appear between u and w on any shortest path.

To perform subpath swapping on undirected graphs, we need an initial pair of nodes in W such that the rest of the nodes in W will be inserted between this initial pair (in the DAG algorithm, this initial pair a, b was the first and last nodes in W in the topological order). For undirected graphs, our initial pair is the pair a, b of nodes in W whose distance is maximum. We order the rest of the nodes in W by their distance from a, to form $v_1, \ldots, v_{|W|-2}$.

Now, our consistent ordering property from above implies the following: for any shortest path P containing a, all nodes in $W \cap P$ are ordered as a subsequence of $a, v_1, v_2, \ldots, v_m, b$ on P. As a result, we can perform the same type of iterative subpath swapping argument as the DAG algorithm.

3.3 Roundtrip Paths in Directed Graphs

The situation for directed graphs is significantly more involved than the previous cases. There are several challenges that are present for directed graphs that were not present for either undirected graphs or DAGs. These difficulties stem from the fact that the *interactions* between shortest paths is much more complicated in directed graphs than in DAGs or undirected graphs.

We first outline these challenges, and then provide an overview of how we address them. Our techniques for addressing these issues exemplify that despite the possibly complex arrangement of shortest paths in directed graphs, there still exists an underlying structure to extract. We hope that our methods might illuminate some structural properties of shortest paths in directed graphs in a way that could apply to other directed-shortest-path problems.

Recall that our goal is to prove the following theorem:

▶ **Lemma 17** (General Directed Result). *Given a directed graph and subset W of nodes, let $\mathcal{R}$ be a collection of shortest paths with the following property:*

$$\text{for every set } S \subseteq W \text{ of 11 nodes, some path in } \mathcal{R} \text{ contains every node in } S. \qquad (\dagger)$$

Further suppose that applying any sequence of $O(n^3)$ subpath swaps to $\mathcal{R}$ yields a collection of paths that still has property (†). Then starting from $\mathcal{R}$, there exists a sequence of subpath swaps that results in a collection of shortest paths in which either:

1. *some path P passes through all nodes in W, or*
2. *the union of two paths P, P' contain all nodes in W, and the first and last nodes on P from W are the same as the last and first nodes on P' from W, respectively.*

Challenges for directed graphs

Challenge 1: No "extremal" nodes

In the proofs for DAGs and undirected graphs, to begin building the path P containing all nodes in W, we chose two initial extremal nodes $a, b \in W$ and added the rest of the nodes of W between a and b. These nodes a, b were straightforward to choose because there was only one pair of nodes in W that could possibly appear first and last on a path containing all nodes in W (for DAGs a and b were the first and last nodes in a topological ordering of W, and for undirected graphs a and b were the pair of nodes in W with largest distance).

For directed graphs however, it is entirely unclear how to pick these two initial extremal nodes. For instance, choosing the pair of nodes $a, b \in W$ with largest directed distance $\mathsf{dist}(a, b)$ does not work because there could be a shortest path Q containing a and b, where a and b are not the first and last nodes in W on Q (in particular, if b appears before a on the shortest path).

To circumvent this issue for directed graphs, we avoid selecting a pair of initial nodes at all. Without initial nodes as an anchor, we cannot build our path P in order from beginning to end as we did for DAGs and undirected graphs. Instead, our goal is to ensure the following weaker property: as we iteratively transform the overall collection of shortest paths $\mathcal{R}$, the set of nodes in W on the path in $\mathcal{R}$ containing the most nodes in W grows over time. That is, the path of $\mathcal{R}$ containing the most nodes in W might currently be P, but at the previous iteration, the path of $\mathcal{R}$ with the most nodes in W might have been a different path P'. In this case, our weaker property ensures that P currently contains a *superset* of the nodes in W that P' contained at the previous iteration.

To perform a single iteration with this guarantee, we may need to significantly reconfigure *many* different paths of $\mathcal{R}$ via many subpath swaps. As a result, our path building procedure is much more intricate than the procedures employed for DAGs and undirected graphs.

Challenge 2: No consistent ordering of nodes on shortest paths

In the proof for DAGs and undirected graphs, we were able to perform subpath swaps due to the following crucial property: consider an arbitrary set of nodes $v_1, \ldots, v_\ell$ (for any ℓ) in a DAG or an undirected graph. If there is a shortest path containing the nodes $v_1, \ldots, v_\ell$ in that order, then *every* shortest path containing these nodes has them in that same order.

For directed graphs, however, this property is not even close to being true. In fact, given that the nodes $v_1, \ldots, v_\ell$ appear in that order on some shortest path, there are *exponentially many* other possible orderings of these nodes on other shortest paths. For instance, when $\ell = 4$, given that the nodes v_1, v_2, v_3, v_4 appear in that order on some shortest path, there are eight possible orderings of these nodes on shortest paths, as depicted in Figure 4 (note that despite the large number of possible orderings, not all orderings are possible; for instance the ordering v_1, v_2, v_4, v_3 is not possible, as this would imply that $\mathsf{dist}(v_1, v_4) < \mathsf{dist}(v_1, v_3)$, which contradicts our assumption that some shortest path contains v_1, v_2, v_3, v_4 in that order).

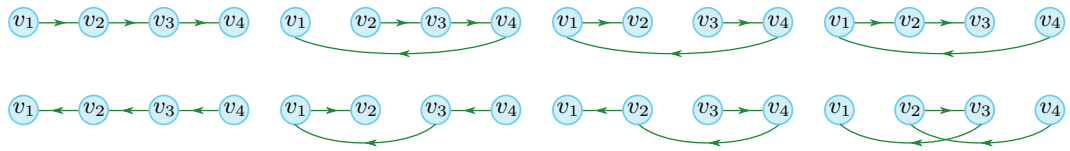


Figure 4 If nodes v_1, v_2, v_3, v_4 appear in that order on some shortest path, they can still appear on different shortest paths in up to seven other distinct possible orders.

Because directed graphs have no consistent ordering of nodes on shortest paths, it becomes much more difficult to perform subpath swaps like those in the algorithms for DAGs and undirected graphs.

To address this challenge, we provide a structural analysis of the ways in which shortest paths in directed graphs can interact with each other. First, as introduced in Section 1.3, we categorize shortest paths into two main types, *reversing* paths and *non-reversing* paths, and we prove the Reversing/Non-Reversing Path Lemma (Lemma 9). Roughly speaking, this lemma is useful because it helps us construct sets of nodes that exhibit some sort of consistent ordering property. This, in turn, enables us to perform sequences of subpath swaps. Defining these consistently ordered sets of nodes and the corresponding subpath swaps is the most involved part of the proof, and works differently for each of the two path types.

Proof structure

Our proof is structured as follows. Initially, we define P to be the path in $\mathcal{R}$ that contains the most nodes in W (breaking ties arbitrarily). Then we proceed with the following two cases:

- (Case 1) We first check whether, roughly speaking, P is contained in a *cycle* that contains *all* nodes in W. In this case, we can use a sequence of subpath swaps to build a second path P' so that the union of P and P' contains all nodes in W and have the "roundtrip" structure specified in the theorem statement, in which case we are done.
- (Case 2) If we are not in Case 1, our goal is to augment some path in $\mathcal{R}$ so that the set of nodes of W on the path in $\mathcal{R}$ with the most nodes in W grows (the goal introduced in the discussion of Challenge 1).

After going through these cases, if we are not done we redefine P as the path in $\mathcal{R}$ containing the most nodes in W and repeatedly apply the appropriate case, until we are done. Most technical details of our proof are in the path augmentation procedure of Case 2. We elaborate on the main ideas for this procedure next.

Handling Case 2: Path Augmentation

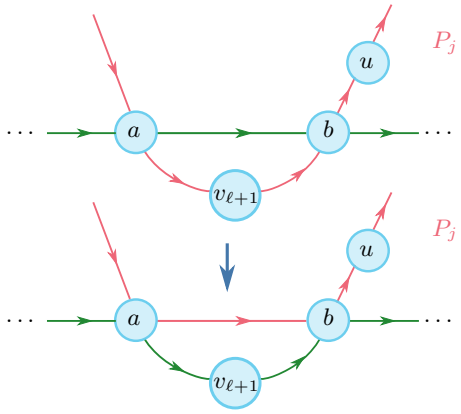
Recall that our goal is the following: Let P be the path in $\mathcal{R}$ containing the most nodes in W, and let $W' = W \cap P$. Fix a node $u \in W \setminus W'$. We wish to perform subpath swaps to yield a path P' that contains every node in $W' \cup \{u\}$.

We begin with a few warm-up cases to motivate our general approach.

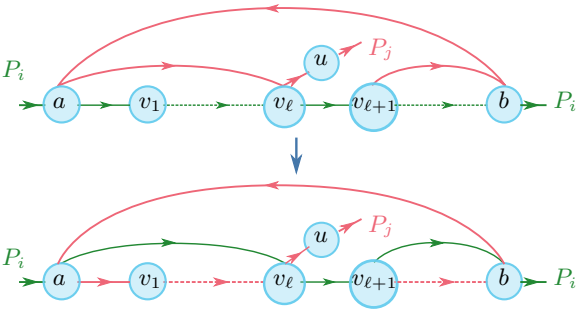
Warm-up cases

Let a and b be the first and last nodes on P that are in W, respectively. By property (†) there exists a path in $\mathcal{R}$ containing a, b, and u. We will suppose in all of the subsequent examples, that for *all* paths in $\mathcal{R}$ containing the nodes a, b, and u, the node u is always the last node in W on the path; this is not a conceptually important assumption and it makes the description simpler.

We claim that if there is a path $P' \in \mathcal{R}$ containing a, b, and u, such that a appears before b, then we are done. Indeed, as depicted in Figure 5a, in this case we can simply swap the subpath $P[a, b]$ with $P'[a, b]$, to form a new solution where P' contains $W' \cup \{u\}$.



(a) If a appears before b on P', a subpath swap lets P' pass through u together with all of the nodes in W'.



(b) More generally, when P' passes through $v_{\ell+1}, b, a, v_\ell$ in that order, we can perform two subpath swaps to get P' to pass through the rest of W'. Here, the dotted segments indicate portions of the paths which pass through the nodes of W' that are not labeled in the figure.

Figure 5 The two warm-up cases.

Now we will slightly generalize this warm-up case. Let $a, v_1, v_2, \ldots, b$ be the nodes in W' in the order they appear on P. Consider v_ℓ and $v_{\ell+1}$ for any ℓ. By property (†) there exists a path in $\mathcal{R}$ containing a, b, u, v_ℓ, and $v_{\ell+1}$. We know from the previous warm-up case that if there is a pathin $\mathcal{R}$ containing these nodes such that a appears before b, then we are done. We also claim that if there exists a path $P' \in \mathcal{R}$ containing these nodes such that $v_{\ell+1}, b, a, v_\ell$ appear in that order, then we are done. This is because as depicted in Figure 5b, we can swap the subpath $P[a, v_\ell]$ with $P'[a, v_\ell]$, and swap the subpath $P[v_{\ell+1}, b]$ with $P'[v_{\ell+1}, b]$. Now, P' contains $W' \cup \{u\}$.

General Approach: "Critical nodes"

We will motivate our general approach in the context of the above warm-up cases. In the second warm-up case, we considered 5 nodes (a, b, u, v_ℓ, and $v_{\ell+1}$) in W', and argued that if there is a path $P' \in \mathcal{R}$ containing these 5 nodes in one of several "good" orders, then we are done because we can perform subpath swaps to reroute P' through all of $W' \cup \{u\}$. Thus, our goal is to choose these 5 (or in general, at most 11) nodes carefully, to guarantee that they indeed fall into a "good" order on some path in $\mathcal{R}$. We refer to these at most 11 nodes as *critical nodes*:

▶ **Definition 20** (Critical Nodes (Informal)). *Given a path $P \in \mathcal{R}$, a set of nodes $T \subseteq W$ of size $|T| \leq 11$ are* critical nodes of P *if there exists a path $P' \in \mathcal{R}$ containing the nodes of T in an order that allows us to perform subpath swaps to reroute some path in $\mathcal{R}$ through all of $W' \cup \{u\}$ (where $W' = W \cap P$).*

Our general approach is to show that any path $P \in \mathcal{R}$ contains a set of at most 11 critical nodes. We remind the reader that this section only concerns Case 2, so our goal is to show that P contains a set of critical nodes only if P does not already fall into Case 1. After showing that P contains a set of critical nodes, we are done, because performing subpath swaps to yield a path containing $W' \cup \{u\}$ was our stated goal.

It is not at all clear a priori that any P should contain a set of critical nodes. Indeed, the critical nodes need to be chosen very carefully. Specifically, they need to be chosen based on the structure of the path P.

This is where the definitions from Section 1.3 come into play. We categorize P based on whether it is *reversing* or *non-reversing*, (or trivial). Then we construct the critical nodes of P using a different procedure specialized for which type of path P is.

If P is a reversing path, then we argue that a valid choice of critical nodes are a, b, and u, along with the two nodes at the two boundaries between the segments defined in Lemma 9 (and a few other nodes for technical reasons). To make this argument, which we will not detail here, we construct an involved series of subpath swaps to reroute some path through all of $W' \cup \{u\}$.

On the other hand, if P is a non-reversing path, the critical nodes are less straightforward to define than if P is a reversing path. Indeed, defining the critical nodes for non-reversing paths is the most conceptually difficult part of our proof. The high-level reason for this difficulty is the fact that the nodes of Segment 2 of a non-reversing path are quite unconstrained because they admit 3 possible shortest-path orderings instead of only 2.

To be more concrete, suppose every node on P falls into Segment 2. For simplicity, suppose we were to choose critical nodes a, b, w, where a and b are the endpoints of P and also happen to be in W', and w is any other node in W'. Consider the path $P' \in \mathcal{R}$ containing a, b, and w, which exists by property (†). By the definition of Segment 2, there are 3 possible orderings of a, b, w on P' ($a \to w \to b$, or $b \to a \to w$, or $w \to b \to a$). Note that these 3 orderings are cyclic shifts of one another. Suppose, as an illustrative example, that P' has the ordering $b \to a \to w$. We would like to reroute P' through all of $W' \cup \{u\}$, but we have a problem. For any node $s \in W'$ that appears between a and w on P, we can reroute P' through s by swapping the subpath $P[a, w]$ with $P'[a, w]$; however, for a node $s \in W'$ that appears between w and b on P, we cannot do this because there is no path segment from w to b on P', since b appears before w on P'. Thus, we cannot reroute P' through s. That is, no matter how we choose the critical nodes, if we do not impose extra structural constraints, we can always identify a segment of the path that we cannot reroute P' through. Overcoming this issue is our main technical challenge, and we defer it to the full proof.

References

1. Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 377–391. SIAM, 2016. doi:10.1137/1.9781611974331.ch28.
2. Saeed Akhoondian Amiri, Stephan Kreutzer, Dániel Marx, and Roman Rabinovich. Routing with congestion in acyclic digraphs. *Inf. Process. Lett.*, 151, 2019. doi:10.1016/j.ipl.2019.105836.
3. Saeed Akhoondian Amiri and Julian Wargalla. Disjoint shortest paths with congestion on dags. *CoRR*, abs/2008.08368, 2020. arXiv:2008.08368.

4 Matthew Andrews. Approximation algorithms for the edge-disjoint paths problem via raecke decompositions. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 277–286. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.33.

5 Matthew Andrews and Lisa Zhang. Hardness of the undirected congestion minimization problem. *SIAM J. Comput.*, 37(1):112–131, 2007. doi:10.1137/050636899.

6 Yossi Azar and Oded Regev. Combinatorial algorithms for the unsplittable flow problem. *Algorithmica*, 44(1):49–66, 2006. doi:10.1007/s00453-005-1172-z.

7 Alok Baveja and Aravind Srinivasan. Approximation algorithms for disjoint paths and related routing and packing problems. *Math. Oper. Res.*, 25(2):255–280, 2000. doi:10.1287/moor.25.2.255.12228.

8 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)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:14, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2021.26.

9 Kristóf Bérczi and Yusuke Kobayashi. The directed disjoint shortest paths problem. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 13:1–13:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.13.

10 Aaron Bernstein and Nicole Wein. Closing the gap between directed hopsets and shortcut sets, 2022. doi:10.48550/arXiv.2207.04507.

11 Greg Bodwin. Linear size distance preservers. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 600–615. SIAM, 2017. doi:10.1137/1.9781611974782.39.

12 Greg Bodwin. On the structure of unique shortest paths in graphs. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2071–2089. SIAM, 2019. doi:10.1137/1.9781611975482.125.

13 Greg Bodwin and Gary Hoppenworth. Folklore sampling is optimal for exact hopsets: Confirming the $\sqrt{n}$ barrier, 2023. doi:10.48550/arXiv.2304.02193.

14 Massimo Cairo, Roberto Grossi, and Romeo Rizzi. New bounds for approximating extremal distances in undirected graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 363–376. SIAM, 2016. doi:10.1137/1.9781611974331.ch27.

15 Shiri Chechik and Tianyi Zhang. Nearly 2-approximate distance oracles in subquadratic time. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 551–580. SIAM, 2022. doi:10.1137/1.9781611977073.26.

16 Chandra Chekuri and Alina Ene. Poly-logarithmic approximation for maximum node disjoint paths with constant congestion. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 326–341. SIAM, 2013. doi:10.1137/1.9781611973105.24.

17 Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Edge-disjoint paths in planar graphs with constant congestion. *SIAM J. Comput.*, 39(1):281–301, 2009. doi:10.1137/060674442.

18 Julia Chuzhoy. Routing in undirected graphs with constant congestion. *SIAM J. Comput.*, 45(4):1490–1532, 2016. doi:10.1137/130910464.

19 Daniel Cizma and Nati Linial. Geodesic geometry on graphs. *Discrete & Computational Geometry*, 68(1):298–347, January 2022. doi:10.1007/s00454-021-00345-w.

20 Daniel Cizma and Nati Linial. Irreducible nonmetrizable path systems in graphs. *Journal of Graph Theory*, 102(1):5–14, June 2022. doi:10.1002/jgt.22854.

21 Lenore Cowen and Christopher G. Wagner. Compact roundtrip routing for digraphs. In Robert Endre Tarjan and Tandy J. Warnow, editors, *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland, USA*, pages 885–886. ACM/SIAM, 1999. URL: http://dl.acm.org/citation.cfm?id=314500.315068.

22 Tali Eilam-Tzoreff. The disjoint shortest paths problem. *Discrete applied mathematics*, 85(2):113–138, 1998.

23 Michael Elkin and Ofer Neiman. Linear-size hopsets with small hopbound, and constant-hopbound hopsets in RNC. In Christian Scheideler and Petra Berenbrink, editors, *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, Phoenix, AZ, USA, June 22-24, 2019*, pages 333–341. ACM, 2019. doi:10.1145/3323165.3323177.

24 Shang-En Huang and Seth Pettie. Thorup-zwick emulators are universally optimal hopsets. *Inf. Process. Lett.*, 142:9–13, 2019. doi:10.1016/j.ipl.2018.10.001.

25 Shang-En Huang and Seth Pettie. Lower bounds on sparse spanners, emulators, and diameter-reducing shortcuts. *SIAM J. Discret. Math.*, 35(3):2129–2144, 2021. doi:10.1137/19M1306154.

26 Ken-ichi Kawarabayashi and Yusuke Kobayashi. Breaking $o(n^{1/2})$-approximation algorithms for the edge-disjoint paths problem with congestion two. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 81–88. ACM, 2011. doi:10.1145/1993636.1993648.

27 Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Stephan Kreutzer. An excluded half-integral grid theorem for digraphs and the directed disjoint paths problem. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 70–78, 2014.

28 Stavros G. Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using greedy algorithms and packing integer programs. In Robert E. Bixby, E. Andrew Boyd, and Roger Z. Ríos-Mercado, editors, *Integer Programming and Combinatorial Optimization, 6th International IPCO Conference, Houston, Texas, USA, June 22-24, 1998, Proceedings*, volume 1412 of *Lecture Notes in Computer Science*, pages 153–168. Springer, 1998. doi:10.1007/3-540-69346-7_12.

29 Ilia Krasikov and Steven D. Noble. Finding next-to-shortest paths in a graph. *Inf. Process. Lett.*, 92(3):117–119, 2004. doi:10.1016/j.ipl.2004.06.020.

30 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.

31 Raul Lopes and Ignasi Sau. A relaxation of the directed disjoint paths problem: A global congestion metric helps. *Theor. Comput. Sci.*, 898:75–91, 2022. doi:10.1016/j.tcs.2021.10.023.

32 Prabhakar Raghavan and Clark D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Comb.*, 7(4):365–374, 1987. doi:10.1007/BF02579324.

33 Liam Roditty, Mikkel Thorup, and Uri Zwick. Roundtrip spanners and roundtrip routing in directed graphs. *ACM Trans. Algorithms*, 4(3):29:1–29:17, 2008. doi:10.1145/1367064.1367069.

34 Aviad Rubinstein and Virginia Vassilevska Williams. SETH vs approximation. *SIGACT News*, 50(4):57–76, 2019. doi:10.1145/3374857.3374870.

35 Bang Ye Wu. A simpler and more efficient algorithm for the next-to-shortest path problem. In Weili Wu and Ovidiu Daescu, editors, *Combinatorial Optimization and Applications - 4th International Conference, COCOA 2010, Kailua-Kona, HI, USA, December 18-20, 2010, Proceedings, Part II*, volume 6509 of *Lecture Notes in Computer Science*, pages 219–227. Springer, 2010. doi:10.1007/978-3-642-17461-2_18.