Counting Vertex-Disjoint Shortest Paths in Graphs

Adi Botea,[1]* Massimiliano Mattetti,[2] Akihiro Kishimoto,[3] Radu Marinescu,[4] Elizabeth Daly[4]

[1]Eaton
[2]Microsoft
[3]IBM Research, Tokyo, Japan
[4]IBM Research Europe
adibotea@eaton.com, massimiliano.mattetti@microsoft.com, akihiro.kishimoto@ibm.com, radu.marinescu@ie.ibm.com, elizabeth.daly@ie.ibm.com

Abstract

Finding a shortest path in a graph is at the core of many combinatorial search problems. A closely related problem refers to counting the number of shortest paths between two nodes. Such problems are solvable in polynomial time in the size of the graph. However, more realistic problem formulations could additionally specify constraints to satisfy. We study the problem of counting the shortest paths that are vertex disjoint and can satisfy additional constraints. Specifically, we look at the problems of counting vertex-disjoint shortest paths in edge-colored graphs, counting vertex-disjoint shortest paths with directional constraints, and counting vertex-disjoint shortest paths between multiple source-target pairs. We give a detailed theoretical analysis, and show formally that all of these three counting problems are NP-complete in general.

Introduction

The shortest path problem is perhaps one of the most studied combinatorial optimization problems. It has a wide range of applications in areas such as artificial intelligence, operations research, bioinformatics, navigation or vehicle routing. A closely related problem is counting the number of shortest paths between two nodes in a graph. The latter, which is also known to be solvable in polynomial time, is relevant to problems such as sensitivity analysis and strategic planning (Byers and Waterman 1984; Cormen et al. 2009).

However, in many real-world scenarios it is important to consider shortest paths that satisfy additional properties. For example, in a logistics scenario, in order to minimize the risk of failure, it may be relevant to compute a set of shortest routes between a source and a destination in a road network with the additional property that they do not have common points of failure (i.e., do not share common nodes[1] or road segments) (Ahuja, Magnati, and Orlin 1993; Bast et al. 2016). Constraints on the paths can arise, for instance, from the fact that, at certain intersections, the possible directions to continue depend on the incoming route. This can be due to the physical configuration of the road, or the presence of traffic signs (e.g., right turn only). These are examples of directional constraints.

Similar requirements can be present in evacuations scenarios in case of natural disasters. To minimize road congestion during the actual evacuation, it could be relevant to compute a set of diverse shortest paths from a particular source to multiple destinations, such that the paths share no common points or road segments (Even, Pillac, and Hentenryck 2015; Kumar, Romanski, and Van Hentenryck 2016).

In network flows problems, counting the paths between a source and a destination, under the condition that paths satisfy additional constraints, can be used to compute the strength of a relation between the two vertices in the graph (Edmonds and Karp 1972; Ahuja, Magnati, and Orlin 1993).

Social networks and, more generally, knowledge graphs allow to consider heterogeneous types of edges, to describe multiple types of relations between entities. For example, in a social network, "friend" and "colleague" can represent two types of edges. Graphs with edges labelled with a type are also called edge-colored graphs. Vertex-disjoint paths with coloring constraints are relevant in problems such as social network analysis (Wu 2012).

Thus, vertex-disjoint paths with and without constraints are relevant in multiple domains. In this paper, we focus on counting shortest vertex-disjoint paths with and without constraints. More specifically, we consider the following three related problems: (1) counting vertex disjoint shortest paths in edge-colored graphs; (2) counting vertex-disjoint shortest paths with ordering constraints; and (3) counting vertex-disjoint shortest paths for multiple source–target pairs. We show formally that, unlike the classical counting shortest paths problem, these counting problems are all NP-hard.

The paper is organized as follows. Next we present related work, followed by the technical background with definitions and notations used in our analysis. Then we give the main hardness results regarding the counting problems considered. The last section provides concluding remarks and outlines some directions of future research.

*This work has been performed when the first, the second and the third authors were affiliated with IBM Research, Ireland.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]We say that two or more paths are vertex disjoint if they have no common vertices, other than the start and the end points.

Related Work

Finding disjoint paths in a graph has been studied extensively in the literature. As summarized in this section, some variants of disjoint path-finding are proven to be solvable in polynomial time, while others are proven to be NP-hard. It is therefore non-trivial to analyze the computational complexity of our new variants.

Given a graph and k pairs of vertices $(s_0, t_0), (s_1, t_1), \ldots, (s_{k-1}, t_{k-1})$, the *$k$ vertex-disjoint paths problem* is to find k pairwise vertex-disjoint paths connecting the pairs (s_i, t_i), with $i = 0, 1, 2, \ldots, k-1$. Karp (1975) proved that this problem is NP-complete when k is a variable part of the input. The problem remains NP-hard even if the graph is constrained to be planar (Lynch 1975; Middendorf and Pfeiffer 1993). If k is a fixed number, k pairwise vertex-disjoint paths can be found in polynomial time in directed planar graphs (Schrijver 1994) as well as in directed acyclic graphs (Fortune, Hopcroft, and Wyllie 1980), whereas the problem in general directed graphs is NP-hard even if $k = 2$ (Fortune, Hopcroft, and Wyllie 1980). Polynomial time algorithms have been found for undirected graphs when k is fixed (Robertson and Seymour 1995; Kawarabayashi, Kobayashi, and Reed 2012).

Optimization versions of the problem focus on *short* vertex-disjoint paths. In the *min-max k vertex-disjoint paths problem*, given a graph where each edge has a non-negative integer length, and k pairs of vertices $(s_0, t_0), (s_1, t_1), \ldots, (s_{k-1}, t_{k-1})$, the task is to find k pairwise vertex-disjoint paths $P_0, P_1, \ldots, P_{k-1}$ such that for each i with $0 \leq i \leq k-1$, P_i is a path from s_i to t_i and the maximum length of the paths is minimized. Li, McCormick, and Simchi-Levi (1990) proved that this problem is strongly NP-complete for both directed and undirected graphs even when $k = 2$. Moreover, the problem is strongly NP-hard for general directed graphs when s_0, s_1, t_0 and t_1 are distinct (Itai, Perl, and Shiloach 1982). In the *min-sum k disjoint paths problem* the task is to minimize the total length (minimum sum) of the paths. The problem is NP-hard in general (Karp 1975; Fortune, Hopcroft, and Wyllie 1980), with a few known cases that are solvable in polynomial time (Scheffler 1994; Kobayashi and Sommer 2010; Verdière and Schrijver 2011).

The *length-bounded k vertex-disjoint paths problem* is defined as follows. Given a graph in which each edge has a non-negative integer length, k pairs of vertices $(s_0, t_0), (s_1, t_1), \ldots, (s_{k-1}, t_{k-1})$, and k length upper bounds $b_0, b_1, \ldots, b_{k-1}$, find k pairwise vertex-disjoint paths $P_0, P_1, \ldots, P_{k-1}$ such that for each i, $0 \leq i \leq k-1$, P_i is a path from s_i to t_i and the length of P_i is at most b_i. For directed graphs the problem is NP-complete even when $k = 2$ and the graph is a directed acyclic graph (Itai, Perl, and Shiloach 1982). For undirected graphs the problem is NP-complete even when $k = 2$ (Li, McCormick, and Simchi-Levi 1990). Moreover, on an undirected planar graph, the problem is strongly NP-hard for non-fixed k and is NP-hard for $k = 2$ (van der Holst and de Pina 2002). Cai and Ye (2016) presented fixed-parameter tractable algorithms for this problem when parameterized by the length constraints b_i on (s_i, t_i)-paths P_i for $i = 0, 1$.

In the *max-disjoint-l-bounded-paths problem* the task is to find the maximum number of disjoint paths between two distinct vertices s and t of length at most l. Itai, Perl, and Shiloach (1982) and Bley (2003) showed that the problem is solvable in polynomial time for both edge-disjoint paths and vertex-disjoint paths, when $l \leq 3$ and $l \leq 4$, respectively, and it is APX-complete for both types of paths when $l \geq 5$. When restricted to disjoint paths of a *shortest possible length*, the problem admits a polynomial time solution (Tragoudas and Varol 1997). Moreover, fixing the number k of paths instead of the length l the problem is still NP-complete even when $k = 2$ (Itai, Perl, and Shiloach 1982). Golovach and Thilikos (2011) studied the parameterized complexity of the problem and presented fixed-parameter tractable algorithms parameterized in both k and l.

Eilam-Tzoreff (1998) introduced the *k disjoint shortest paths (kDSP) problem*. Given a graph and k pairs of distinct vertices $(s_0, t_0), (s_1, t_1), \ldots, (s_{k-1}, t_{k-1})$, find whether there exist k pairwise disjoint *shortest* paths P_i, between s_i and t_i for each i, with $0 \leq i \leq k-1$. It can be defined on directed or undirected graphs and the paths can be vertex disjoint or edge disjoint. These four problems are NP-complete when k is part of the input, even for planar graphs (Eilam-Tzoreff 1998). In the same paper, polynomial time algorithms are also presented for the undirected 2DSP problem for both vertex disjoint and edge disjoint paths.

Wu (2012) presented a variant of the problem called the *maximum disjoint paths problem on edge-colored graphs* (MaxCDP). It arose from social network analysis (SNA) where different kinds of relations are considered. Given a graph, and two terminals s and t, the goal is to find the maximum number of vertex-disjoint paths consisting of edges of the same color (also called uni-color paths) connecting s to t. MaxCDP is polynomial time solvable when the input graph contains exactly one color, while it is NP-hard when the edges of the input graph are associated with at least two colors (Wu 2012). The length-bounded version of the problem (l-LCDP) is proven to be NP-hard when $l \geq 4$, while it is polynomial time solvable for $l < 4$ (Wu 2012).

Notation and Definitions

Let $G = (V, E)$ be an undirected graph with V being the set of vertices and E being the set of edges. Two vertices $v_i, v_j \in V$ are *adjacent* if they are incident to an edge $e_{i,j}$ in E, i.e., $e_{i,j} = (v_i, v_j) \in E$. A path in G is a sequence of vertices $\pi = (v_{i_1}, \ldots, v_{i_n})$ such that v_{i_j} is adjacent with $v_{i_{j+1}}$ for $1 \leq j < n$. Given a non-negative real-valued weight function $f : E \rightarrow \mathbb{R}_+$, the cost of a path $\pi = (v_{i_1}, \ldots, v_{i_n})$ is defined as $\sum_{j=1}^{n-1} f(e_{i_j, i_{j+1}})$. A shortest path from s to t is a path with a mimimum cost among all paths from s to t. When each edge in the graph has weight 1, i.e., f is a constant function $f : E \rightarrow \{1\}$, a shortest path minimizes the number of edges. The problem is also called the single-pair shortest path problem. Other variations of this problem include the all-pairs shortest path problem and single-source or single-destination shortest path problems.

Given a graph G together with the source and destination vertices s and t, respectively, there could be more than one shortest paths between s and t. Clearly, these paths can have common vertices and/or edges. Computing the number of shortest paths between s and t can be done in polynomial time (Ahuja, Magnati, and Orlin 1993; Yen 1971; Eppstein 1998).

However, here we consider vertex-disjoint paths, which in addition can feature constraints such as the following.

Definition 1 (Directional constraints)**.** *Given a graph $G = (V, E)$ and a vertex $v_b \in V$, a* directional constraint *C on v_b is a function E_i that assigns to each edge $e_{i,b} = (v_i, v_b) \in E$ incident to v_b a set of edges $E_i(e_{i,b})$ different from $e_{i,b}$. A path π in G satisfies the directional constraint C at v_b if and only if π includes an edge from $E_i(e_{i,b})$ as an outgoing edge from v_b whenever $e_{i,b}$ is the incoming edge at v_b in the path.*

Definition 2 (Forced-sequence directional constraints)**.** *A* forced-sequence directional constraint *C on a vertex v_b is a* directional constraint *where $|E_i(e_{i,b})| = 1$ for each edge $e_{i,b} = (v_i, v_b) \in E$ adjacent to v_b.*

For example, at an intersection on a road map, it may be forbidden to turn to the left or to the right, in which case going straight is the *only way to continue*.

Definition 3 (Uni-color path constraints)**.** *A c-edge-colored graph is defined as $G = (V, \xi)$, where V denotes the set of vertices of G and $\xi = \{E_1, E_2, ..., E_c\}$ is a collection of c edge sets. For $1 \leq i \leq c, E_i \subseteq V \times V$ is the edge set of color i. A path π in G is called a* uni-color path *if all the edges of π have the same color, that is they belong to the same set E_i, $1 \leq i \leq c$.*

In a heterogeneous graph representing various types of relations between entities, some types of pathways representing indirect relations between entities could be considered relevant, whereas other types of pathways could be considered irrelevant. For example, consider two types of edges in a social network, having the types FRIEND and COLLEAGUE. A two-step pathway of the type FRIEND–FRIEND could be considered relevant, and so would be COLLEAGUE–COLLEAGUE. However, FRIEND–COLLEAGUE might be considered irrelevant in our example.

Observe that the types of constraints presented in this section are not necessarily mutually exclusive. For example, as the name suggests, forced-sequence directional constraints are a particular type of a directional constraint.

Definition 4 (Constrained paths)**.** *Given a graph G and a set of constraints $\mathcal{C}$, a* constrained path *relative to $\mathcal{C}$ is a path in G that satisfies those constraints. When $\mathcal{C}$ is clear from the context, we simply say a constrained path, with no explicit reference to $\mathcal{C}$.*

Definition 5 (Constrained shortest paths)**.** *Given a graph G and set of constraints $\mathcal{C}$, a* constrained shortest path *is a path in G that satisfies the constraints, and that has a smallest cost among all paths in G that satisfy the constraints.*

Definition 6 (Vertex disjoint paths)**.** *Given a graph $G = (V, E)$ and two vertices $s, t \in G$, we say that two or more paths from s to t in G are* vertex disjoint *if they have no other vertices in common except for s and t, respectively.*

We will show next that, unlike counting shortest paths, which can be done in polynomial time (Ahuja, Magnati, and Orlin 1993; Yen 1971; Eppstein 1998), counting constrained shortest paths that are also vertex disjoint is actually hard.

Counting Disjoint Uni-Colored Shortest Paths in Edge-Colored Graphs

We consider counting shortest paths in graphs with colored edges, which are also called *edge-colored graphs* or *c-edge-colored graphs* when colors are taken from a set $[c] = \{1, 2, \ldots, c\}$. We are interested in counting *uni-colored* shortest paths.

Definition 7. *The NVD-USP[2] problem is defined as follows. Input: a edge-colored graph $G = (V, \xi)$, a pair of vertices (s, t), and an integer N. Let c^* be the apriori unknown (i.e., not given as input) cost of a shortest uni-color path from s to t. Question: Are there at least N vertex-disjoint shortest uni-color paths (i.e., uni-color paths of cost c^*) from s to t?*

Lemma 1. *Given a colored graph G with z distinct edge colors, the cost of a shortest uni-color path from a node s to a node t can be computed in polynomial time.*

Proof. It is sufficient to observe that we can consider z copies of the graph, one limited to edges of the corresponding color, compute the cost of a shortest path in each graph in polynomial time, and take their minimum value. $\square$

Note that the proof idea to the previous lemma does not imply that NVD-USP can be solved in polynomial time. In other words, separately counting shortest uni-color paths, in separate copies of the graph, does not give a solution to NVD-USP. The reason is that, in NVD-USP, all paths considered need to be vertex disjoint, a property that is not observed if uni-color paths are considered separately from each other, with one copy of the graph for each color. In fact, rather than being solvable in polynomial time, we claim and prove below that NVD-USP is hard. Specifically:

Theorem 1. *NVD-USP is NP-complete.*

Proof. Using Lemma 1, we can see that the problem is within NP, as a solution can be verified in polynomial time.

We prove the hardness with a reduction from the set cover problem (SCP), an NP-hard problem (Karp 1972). The SCP problem definition includes a set of elements (universe) $U = \{1, 2, \ldots, n\}$, and a collection $E = \{A_1, A_2, \ldots, A_m\}$ of m subsets of U, with the property that the union of all subsets is U. A sub-collection C of E is a cover if the union of all subsets in C is equal to U. Given an integer k, the task is to answer whether a cover of size no larger than k exists.

Given an arbitrary SCP instance, we build a NVD-USP instance. We use a toy SCP problem as a running example. In the example, $U = \{1, 2, 3, 4, 5\}$, $E = \{A_1, A_2, A_3\}$, $A_1 = \{1, 2, 3\}$, $A_2 = \{2, 3\}$ and $A_3 = \{4, 5\}$. Figure 1 illustrates how to build the graph of the NVD-USP instance.

For each element in $i \in U$, we define a vertex O_i. In the running example, there are five such vertices. For each subset A_j, we define $|A_j|$ vertices. Each such a vertex represents

[2]The acronym stands for Number of Vertex-Disjoint Uni-Color Shortest Paths.

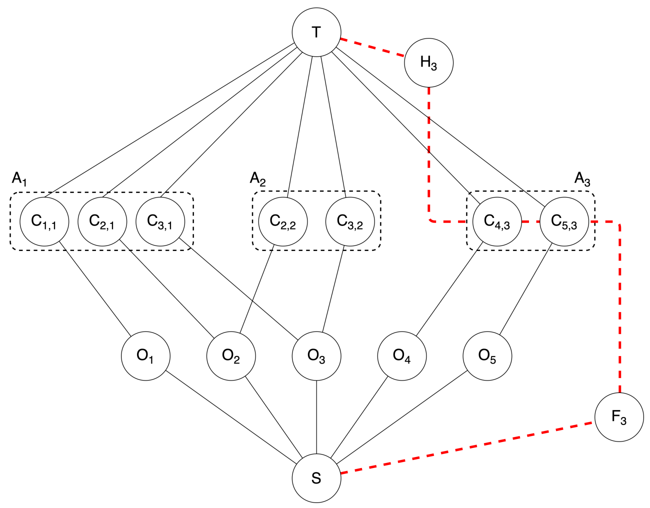


Figure 1: Building the graph of the NVD-USP instance for the running example.

the corresponding element in A_j. We denote these vertices as $C_{i,j}$, where i is the index of the corresponding element in U.

For each subset A_j, we further define two vertices F_j and H_j. Figure 1 shows this only for subset A_3, to avoid clutter.

We define two more vertices, denoted as S and T.

There are two types of edges (colors), solid and dashed. There is a solid edge from S to each vertex O_i. Furthermore, we have solid edges from O_i to vertices $C_{i,j}$, for all j for which $C_{i,j}$ is defined. There is a solid edge from each vertex $C_{i,j}$ to T as well.

Dashed edges are defined as illustrated in Figure 1 for A_3. For each subset A_j, define a chain of dashed edges going from S to F_j, then going through each vertex $C_{i,j}$ defined for A_j (in some arbitrary but fixed order), then going to H_j and finally going to T.

In the graph built for the NVD-USP instance, we further impose that the paths are constructed in such a way that all solid paths and all dashed paths from S to T have the same length. On a weighted graph, set the weights of edges accordingly. On a unit-cost graph, insert additional vertices along some edges, to obtain the desired property.

We claim that our instance has at least $n + m - k$ disjoint constrained paths from S to T iff the SCP instance has a cover of at most k subsets.

Assume that our instance admits a set Z of at least $n + m - k$ disjoint constrained paths. With no loss of generality, assume that Z contains n solid paths. If Z contains fewer than n solid paths, it follows that there exists a vertex O_p not contained in a path in Z. If none of the solid paths $S, O_p, C_{p,j}, T$ (for all j where $C_{p,j}$ is defined) is disjoint from the current set Z, it follows that all vertices $C_{p,j}$ are already included in paths of the type $S, F_j, \ldots, H_j, T$ (i.e., a dashed path). Remove one of these dashed paths from Z, and add the solid path containing O_p instead. Continue recursively until n solid paths belong to Z. It follows that at least $m - k$ paths are dashed paths. This further implies that all n solid paths go through at most k subsets A_j. In other words, we have a cover set with at most k subsets.

Assume now that a cover set with at most k subsets exist. Route n solid paths through those subsets, and define one dashed path for each of the remaining subsets. These add up to at least $n + m - k$ disjoint constrained shortest paths in total. □

Counting Disjoint Shortest Paths with Directional Constraints

In this section we consider the problem of counting the number of vertex-disjoint shortest paths under the assumption that additional constraints such as directional constraints must be satisfied. Before giving the hardness result, we start by defining formally the counting problem.

Definition 8. *The NVD-CSP[3] problem is defined as follows. Input: a graph $G = (V, E)$, a pair of vertices (s, t), a set of directional constraints, and an integer N. Question: Are there at least N vertex-disjoint constrained shortest paths from s to t?*

Lemma 2. *Given a graph G and a set of directional constraints, the cost of a constrained shortest path between two nodes s and t can be computed in polynomial time.*

Proof. The cost of a constrained shortest path can be computed, for instance, with a modified version of Dijkstra's algorithm (Dijkstra 1959). In this version, we allow defining in the search several copies of a node $v \in V$, one copy for each incoming edge into v. When expanding a given copy of v, consider as successors only the outgoing edges that satisfy the constraint at hand. (When no constraint is defined for the pair containg v and the incoming edge at hand, all outgoing edges are considered as successors). This increases the search space by a polynomial factor, as the number of the copies of a given node v is bounded by the number of its incoming edges. The search considers only paths that satisfy the directional constraints, and thus it returns a correct result (i.e., the cost of a constrained shortest path). □

Theorem 2. *NVD-CSP is NP-complete.*

Proof. Lemma 2 allows us to state that a solution can be verified in polynomial time, and therefore the problem is within NP.

We prove hardness using a reduction from the 3-dimensional matching problem (3DM). The 3DM problem is defined as follows. Let X, Y, Z be three finite and disjoint sets, and let $R \subseteq X \times Y \times Z$ be a relation between these sets. Then $M \subseteq R$ is a 3-dimensional matching if for any two distinct triples $(x_1, y_1, z_1) \in M$ and $(x_2, y_2, z_2) \in M$ it follows that $x_1 \neq x_2$, $y_1 \neq y_2$ and $z_1 \neq z_2$, respectively.

Given a 3DM instance and an integer k, answering whether there exists a matching M with $|M| \geq k$ is known to be an NP-hard problem (Karp 1972).

Consider now an arbitrary 3DM instance. We build a graph as shown in Figure 2, where all elements from X, Y and Z generate one vertex each. For each triple $(x, y, z) \in R$,

[3]The name stands for Number of Vertex-Disjoint Constrained Shortest Paths.

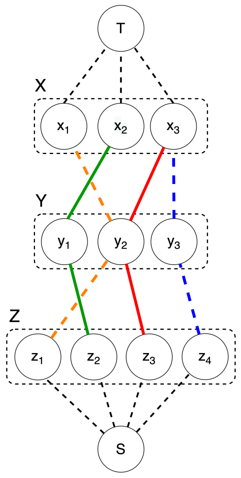


Figure 2: A graph built from a toy 3DM problem.

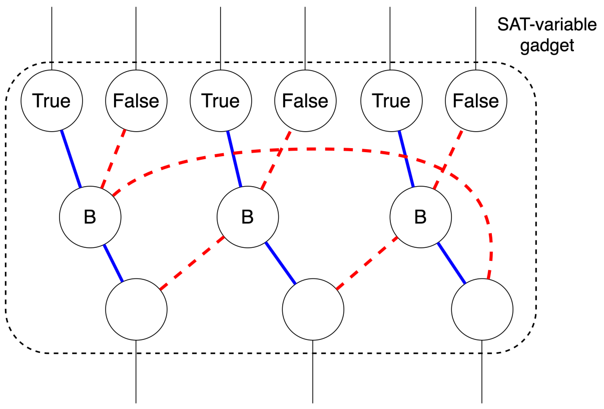


Figure 3: A SAT-variable gadget. In this example, $p = 3$.

define two edges, (x, y) and (y, z). In a constrained path S, x, y, z, T, we allow two consecutive edges (x, y) and (y, z) iff (x, y, z) is a triple. Add two more vertices, S and T. Connect S to all vertices corresponding to Z, and T to all vertices corresponding to X. Observe that all paths between S and T have the same length. The graph has at least k disjoint constrained shortest paths from S to T iff the 3DM instance admits a matching of size at least k. □

Corollary 1. *Theorem 2 holds even if the length of the shortest disjoint paths is as small as 4.*

In addition, we also have that:

Theorem 3. *NVD-CSP is NP-complete even if the constraints are limited to* forced-sequence directional constraints *only.*

We will prove the hardness using a reduction from SAT. In the literature, reductions from SAT have been used to prove the NP-hardness of the decision problem corresponding to multi-agent pathfinding with makespan optimization (Surynek 2010), and a decision problem corresponding to quantum circuit compilation (Botea, Kishimoto, and Marinescu 2018). Our proofs in this work use related building blocks. However, the problems we study are significantly different, and their hardness proofs end up being significantly different as well.

We present some auxiliary constructs which will be used in the proof. Consider a SAT formula where, for each variable v, the number of its positive literals is equal to the number of negative literals. For a variable v in the formula, with p positive literals and p negative literals, a *SAT-variable gadget* corresponding to v is a graph as illustrated in Figure 3. Specifically, the vertices are structured on four layers. The top layer has $2p$ vertices, and each of the remaining layers has p vertices. Paths through such a graph have to satisfy the following forced-sequence directional constraint. If a vertex labelled as B is reached through a red-color edge, the path must continue with a red[4] edge. Likewise, if such a vertex is achieved through a blue[5] edge, the path must continue with a blue edge.

Lemma 3. *Consider a variable v with $2p$ literals (p positive and p negative) in a propositional formula in the conjunctive normal form (CNF). The maximum number of disjoint constrained paths crossing v's SAT-variable gadget is p. A maximal set of disjoint constrained paths satisfies one of the following conditions: they contain all vertices labelled "True" and ignore all vertices labelled "False"; or they contain all vertices labelled "False" and ignore all vertices labelled "True".*

Proof. By construction, all the paths going through the *SAT-variable gadget* contain a vertex of type B. Since there are p vertices of type B, the maximum number of disjoint constrained paths cannot exceed p. Moreover, due to the *forced-sequence directional constraint* a path going through a vertex labelled as B can either proceed towards a vertex labelled as $True$ or one labelled as $False$. Since the disjoint property prevents two paths to share the same vertex B, it follows that the maximum number of disjoint paths, i.e. p, is achieved only when all the paths proceed towards the same type of labelled vertex (either all the $True$ vertices or all the $False$ ones). □

Lemma 3 states that the corresponding number of paths is maximized iff the *Boolean consistency* of the corresponding variable v is ensured. In other words, either all positive literals of v are set to true, and the negative literals are set to false; or all positive literals are set to false, and the negative ones are set to true. We are now ready to prove Theorem 3.

Proof of Theorem 3. Consider an arbitrary SAT instance in CNF. We convert it into an equivalent CNF formula where, for each variable, the number of positive literals is equal to the number of negative literals. This can be performed as follows: assume that initially variable v has o_p occurrences as a positive literal and o_n occurrences as a negative literal, with $o_p \neq o_n$. If $o_n < o_p$, add a clause $(v \vee \neg v \vee \cdots \vee \neg v)$, with $o_p - o_n + 1$ copies of $\neg v$ in that clause. If $o_p < o_n$, add a clause $(\neg v \vee v \vee \cdots \vee v)$, with $o_n - o_p + 1$ copies of v in that clause.

[4]Thick dashed line in black and white print.
[5]Thick solid line in black and white print.

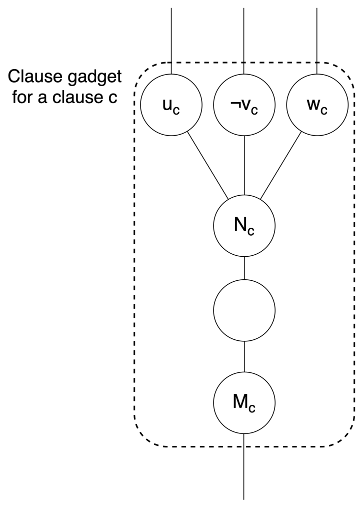


Figure 4: A clause gadget. In this example, the clause $(u \vee \neg v \vee w)$ has three literals: $u, \neg v, w$.

We build a graph as illustrated in Figure 5. As shown in the figure, the graph includes two vertices S and T, which will be the start–target pair of vertices for which we will count the disjoint constrained shortest paths.

For each variable in the SAT formula, we build a SAT-variable gadget. In Figure 5, the gadget corresponds to a variable v with 3 positive literals and 3 negative literals. For each clause c, we build a sub-graph called a *clause gadget*. Figure 4 shows an example for a clause with three literals. Each literal l in the clause generates a vertex l_c, shown at the top level. There are four more vertices, chaining up from M_c to N_c, after which N_c is connected to each of the top-level vertices. Clause gadgets are disjoint from each other.

Next we describe the so-called inter-gadget edges. Given a variable v, there are p clauses that contain v and p clauses that contain $\neg v$. We also have p $True$ vertices and p $False$ vertices in the SAT-variable gadget of v. We (randomly) pick a one-to-one mapping between the clauses that contain v and the vertices labelled $False$. Assume that a given $False$ vertex is mapped to a given clause d containing v. We add an edge between that $False$ vertex and the corresponding v_d vertex in the clause d. Likewise, we map $True$ vertices onto clauses c containing $\neg v$. Then draw an edge between the $\neg v_c$ vertex of a clause c and the vertex $True$ mapped to c. In Figure 5, inter-gadget edges are shown with green[6] dashed lines.

Finally, we connect S to all bottom-layer vertices of all gadgets of both kinds. We also connect all top-layer vertices of clause gadgets to T.

Observe that there are two types of paths between S and T, called *two-gadget paths* and *clause-specific paths*, respectively. We call a two-gadget path a path going from S into a SAT-variable gadget, then exiting from the SAT-variable gadget into a clause gadget via an inter-gadget edge, and finally exiting the clause gadget and reaching T. We call a clause-specific path a path going from S to the bottom-level vertex of a clause gadget, then crossing the clause gadget and finally reaching T.

[6]Thin dashed line in black and white print.

We claim that the SAT formula has a valid assignment (making the formula true) iff our graph has at least $l + m$ disjoint constrained shortest paths, where l is the total number of literals in the formula, and m is the number of clauses.

Assume that the SAT formula has a valid assignment. We build a set of $l + m$ paths. If, in the SAT assignment, a variable v is set to true, add to our set of paths the constrained paths that go through the SAT-variable gadget of v and contain the $True$ vertices. Otherwise, add the constrained paths going through $False$ vertices. According to Lemma 3, each of these two alternatives maximizes the number of disjoint constrained paths traversing that gadget.

By construction, a path containing a $True$ vertex continues through an inter-gadget edge to a vertex $\neg v_c$ that belongs to a clause gadget. As such, when a variable v is set to true, no vertex v_d contained in a clause gadget (for some clause d) belongs to any two-gadget path added to our set. Thus, such a vertex can be used to construct a clause-specific path (for clause d) all the way from S to T (likewise, a path through a $False$ vertex leaves $\neg v_c$ vertices available to be included in a clause-specific path). Since each clause has at least one literal set to true, it follows that we can construct m clause-specific paths.

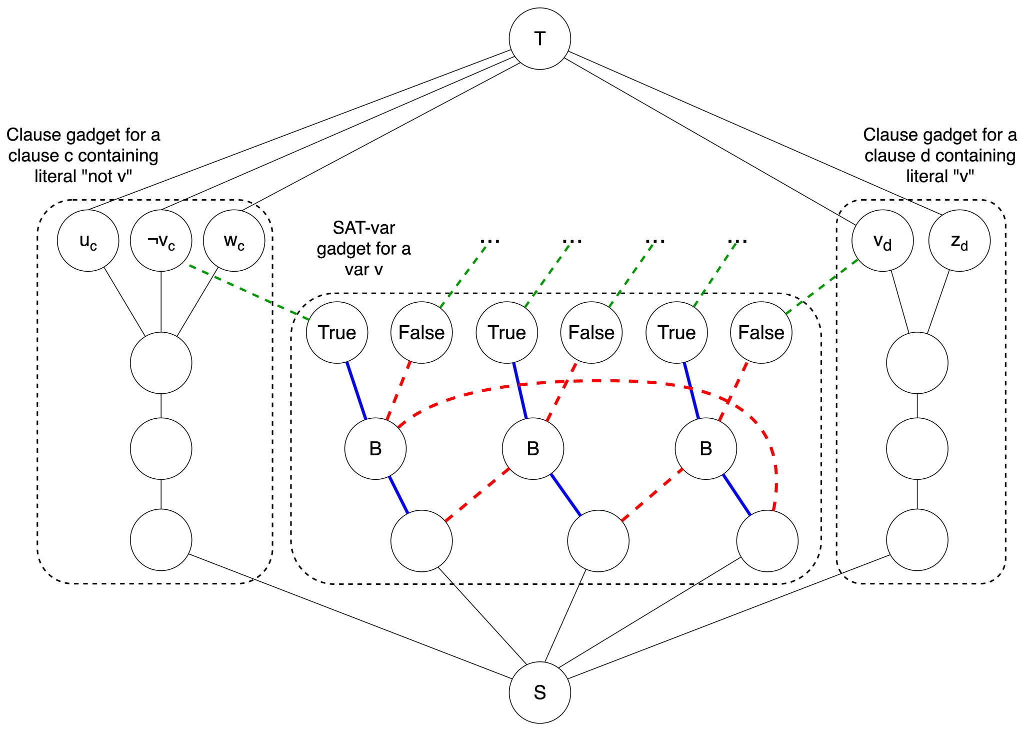
Conversely, consider a set of $l + m$ disjoint constrained shortest paths. The only way to achieve this number of paths is to maximize the number of two-gadget paths, and in addition have m *clause-specific* paths. It follows that each SAT-var gadget provides a maximum number of disjoint paths crossing it (i.e., p paths for a variable v with p positive and p negative literals). This further implies the Boolean consistency, according to Lemma 3. Since each clause contributes with one clause-specific path, it follows that each clause has at least one vertex on the top layer not taken by a two-gadget path. Set to true the literal corresponding to that vertex. This results in a valid assignment to the SAT formula. $\square$

Notice that, the proofs of Theorem 2 and Theorem 3 provide two results with different strengths. The former shows that counting vertex-disjoint shortest path is NP-hard when a *directional constraint* is imposed on a subset of vertices in the graph (and it holds even if the length of the paths is ≥ 4 as stated in Corollary 1). On the other hand, the latter proof shows that the problem is still NP-hard when a *directional constraint forces* the paths to traverse a specific *sequence* of edges.

Corollary 2. *Theorems 2 and 3 hold even if the constraints are limited to only one vertex per disjoint path.*

Counting Disjoint Shortest Paths for Multiple Source-Target Pairs

In the previous sections we focused on counting constrained disjoint shortest paths for one source-target nodes pair in a graph. In this section we tackle the problem of counting disjoint shortest paths for multiple pairs of source-target nodes, without imposing any constraint on the paths.



T
Clause gadget for a clause c containing literal "not v"
Clause gadget for a clause d containing literal "v"
SAT-var gadget for a var v
u_c
¬v_c
w_c
v_d
z_d
True
False
True
False
True
False
B
B
B
...
S

Figure 5: Building a graph for an NVD-CSP instance as a reduction from SAT.

Definition 9. *The NVD-MPP*[7] *problem is defined as follows. Input: a graph $G = (V, E)$, a set of pairs of vertices $(s_0, t_0), \dots, (s_R, t_R)$, and a set of integers $N_0, N_1, \dots, N_R$. Question: Are there at least N_i shortest paths from s_i to t_i, for each i, with the property that all paths considered are disjoint?*

At a quick glance, it might seem that our previous results, such as Theorems 2 and 3, imply the NP-hardness of NVD-MPP. However, this is not the case. The reason is that, in these previous results, the problem considered had constraints on the paths, whereas NVD-MPP imposes no constraints. Thus, we formulate the following claim.

Theorem 4. *NVD-MPP is NP-complete.*

Proof. The problem is in NP, as solutions can be verified in polynomial time. The hardness is shown with a reduction from the set cover problem (SCP). Given an arbitrary SCP instance, we build an NVD-MPP instance as follows. We use a toy SCP problem as a running example. In the example, $U = \{1, 2, 3, 4, 5\}$, $S = \{A_1, A_2, A_3\}$, $A_1 = \{1, 2, 3\}$, $A_2 = \{2, 3\}$ and $A_3 = \{4, 5\}$.[8]

Figure 6 illustrates how to construct the NVD-MPP instance. For each element in $i \in U$, we define a vertex O_i. In the running example, there are five such vertices. For each subset A_j, we define $|A_j|$ vertices. Each such vertex represents the corresponding element in A_j. We denote these vertices as $C_{i,j}$, where i is the index of the corresponding element in U.

For each subset A_j, we further define two vertices F_j and H_j. Figure 6 shows this for subsets A_1 and A_3 only (skipping A_2), to avoid clutter. We define four more vertices, denoted as S, T, S', T'.

Define an edge from S to each vertex O_i. Define an edge from each $C_{i,j}$ to T. Define an edge from S' to each vertex F_j, and from each vertex H_j to T'. Define $s_0 = S, t_0 = T, s_1 = S', t_1 = T'$, $N_0 = n$ and $N_1 = m - k$.

If needed, we modify the graph to ensure that all paths from S' to T' and all paths from S to T have the same cost. This is achieved, for example, by inserting additional nodes and edges within some of those paths.

We claim that the SCP instance has a cover of size at most k iff we have at least n shortest paths from S to T and at least $m - k$ shortest paths from S' to T', with all these paths disjoint from each other.

Assume that the SCP instance has a cover of size at most k. It follows that there are n paths from S to T such that each of these paths crosses some subset A_j contained in the cover. For the remaining $m - k$ subsets, define a path from S' to T'. All paths mentioned here are disjoint from each other. See Figure 6 for an illustration.

For the reverse implication, assume that we have at least n shortest paths from S to T and at least $m - k$ shortest paths from S' to T', with all these paths disjoint from each other. Each of the $m - k$ shortest paths from S' to T' correspond to one subset A_j. It follows that each of the n shortest paths from S to T must cross one of the remaining k subsets. This further implies that the remaining k subsets form a cover. □

[7]This stands for Number of Vertex-Disjoint Multiple Source-Target Pairs Paths.

[8]The reduction is related to Theorem 3's proof, but with significant technical differences.

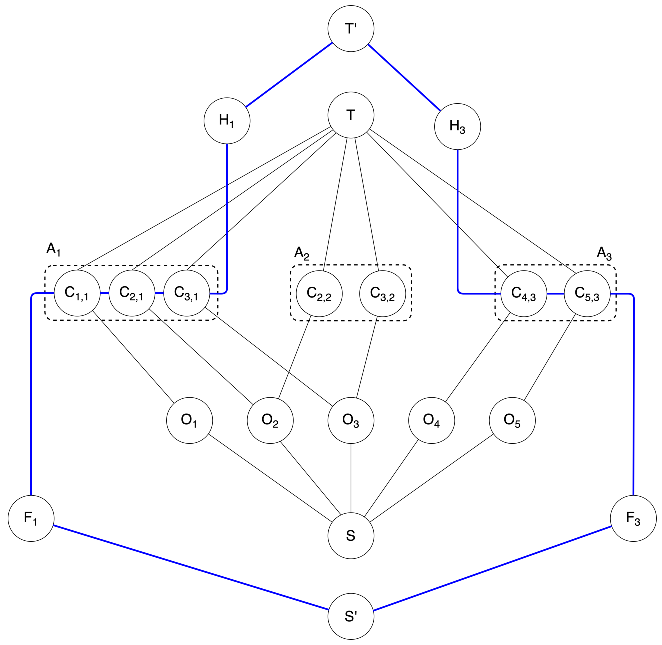


Figure 6: Building the graph of the NVD-MPP instance for the running example. The path $S', F_2, C_{2,2}, C_{3,2}, H_2, T'$ is not shown, to avoid clutter.

Conclusion

Shortest path problems are perhaps amongst the most studied combinatorial problems in computer science. Shortest path problems come in many flavours, and they have many applications. For example, vertex-disjoint shortest paths that might additionally satisfy directional constraints or uni-color constraints have applications in areas such as logistics, evacuation scenarios and social network analysis.

In this paper we have focused on counting vertex-disjoint shortest paths. Specifically, we have presented several variants of the problem, namely counting vertex-disjoint unicolor shortest paths in edge-colored graphs, counting vertex-disjoint shortest paths under directional and forced-sequence directional constraints, and counting vertex-disjoint shortest paths between multiple source-target node pairs. We have formally shown that these counting problems are NP-complete. This result is in contrast with the classical counting shortest paths problem which is known to be polynomial time solvable.

In the future, we plan to investigate the existence of polynomial-time approximation algorithms for all the different variants of the counting problem considered. We further plan to explore efficient heuristics and algorithms for the problems presented.

References

Ahuja, R. K.; Magnati, T. L.; and Orlin, J. B. 1993. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall.

Bast, H.; Delling, D.; Goldberg, A.; Müller-Hannemann, M.; Pajor, T.; Sanders, P.; Wagner, D.; and Werneck, R. F. 2016. Route planning in transportation networks. In Kliemann, L.; and Sanders, P., eds., *Algorithm engineering*, 19–80. Springer.

Bley, A. 2003. On the complexity of vertex-disjoint length-restricted path problems. *Computational Complexity* 12: 131–149.

Botea, A.; Kishimoto, A.; and Marinescu, R. 2018. On the Complexity of Quantum Circuit Compilation. In Bulitko, V.; and Storandt, S., eds., *Proceedings of the Eleventh International Symposium on Combinatorial Search, SOCS 2018*, 138–142. AAAI Press. URL https://aaai.org/ocs/index.php/SOCS/SOCS18/paper/view/17959.

Byers, T. H.; and Waterman, M. S. 1984. Determining all optimal and near-optimal solutions when solving shortest path problems by dynamic programming. *Operations Research* 1(32): 1381–1384.

Cai, L.; and Ye, J. 2016. Finding Two Edge-Disjoint Paths with Length Constraints. In Heggernes, P., ed., *Proceedings of Graph-Theoretic Concepts in Computer Science: 42nd International Workshop*, 62–73. ISSN 0302-9743. doi:https://doi.org/10.1007/978-3-662-53536-3_6.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2009. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition. ISBN 0262033844, 9780262033848.

Dijkstra, E. W. 1959. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* 1(1): 269–271.

Edmonds, J.; and Karp, R. M. 1972. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *Journal of the ACM* 19(2): 248–264. ISSN 0004-5411. doi:10.1145/321694.321699. URL http://doi.acm.org/10.1145/321694.321699.

Eilam-Tzoreff, T. 1998. The disjoint shortest paths problem. *Discrete Applied Mathematics* 85(2): 113–138. ISSN 0166-218X. doi:https://doi.org/10.1016/S0166-218X(97)00121-2. URL http://www.sciencedirect.com/science/article/pii/S0166218X97001212.

Eppstein, D. 1998. Finding the k Shortest Paths. *SIAM Journal on Computing* 2(28): 652–673.

Even, C.; Pillac, V.; and Hentenryck, P. V. 2015. Convergent Plans for Large-Scale Evacuations. In Bonet, B.; and Koenig, S., eds., *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 1121–1127. AAAI Press. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9782.

Fortune, S.; Hopcroft, J.; and Wyllie, J. 1980. The Directed Subgraph Homeomorphism Problem. *Theoretical Computer Science* 10(2): 111–121. ISSN 0304-3975. doi:https://doi.org/10.1016/0304-3975(80)90009-2. URL http://www.sciencedirect.com/science/article/pii/0304397580900092.

Golovach, P. A.; and Thilikos, D. M. 2011. Paths of Bounded Length and Their Cuts: Parameterized Complexity and Algorithms. *Discrete Optimization* 8(1): 72–86. ISSN 1572-5286. doi:https://doi.org/10.1016/j.disopt.2010.09.009. URL http://www.sciencedirect.com/science/article/pii/S1572528610000678. Parameterized Complexity of Discrete Optimization.

Itai, A.; Perl, Y.; and Shiloach, Y. 1982. The Complexity of Finding Maximum Disjoint Paths with Length Constraints. *Networks* 12(3): 277–286. doi:10.1002/net.3230120306. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230120306.

Karp, R. M. 1972. Reducibility among Combinatorial Problems. In Miller R.E., Thatcher J.W., B. J., ed., *Complexity of Computer Computations*, 85–103. ISSN 978-1-4684-2003-6. doi:https://doi.org/10.1007/978-1-4684-2001-2_9.

Karp, R. M. 1975. On the Computational Complexity of Combinatorial Problems. *Networks* 5(1): 45–68. ISSN 0028-3045. doi:10.1002/net.1975.5.1.45. URL https://doi.org/10.1002/net.1975.5.1.45.

Kawarabayashi, K.; Kobayashi, Y.; and Reed, B. 2012. The Disjoint Paths Problem in Quadratic Time. *Journal of Combinatorial Theory, Series B* 102(2): 424–435. ISSN 0095-8956. doi:https://doi.org/10.1016/j.jctb.2011.07.004. URL http://www.sciencedirect.com/science/article/pii/S0095895611000712.

Kobayashi, Y.; and Sommer, C. 2010. On Shortest Disjoint Paths in Planar Graphs. *Discrete Optimization* 7(4): 234–245. ISSN 1572-5286. doi:https://doi.org/10.1016/j.disopt.2010.05.002. URL http://www.sciencedirect.com/science/article/pii/S157252861000037X.

Kumar, K.; Romanski, J.; and Van Hentenryck, P. 2016. Optimizing infrastructure enhancements for evacuation planning. In Schuurmans, D.; and Wellman, M., eds., *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press.

Li, C.; McCormick, S. T.; and Simchi-Levi, D. 1990. The Complexity of Finding Two Disjoint Paths with Min-Max Objective Function. *Discrete Applied Mathematics* 26(1): 105–115. ISSN 0166-218X. doi:https://doi.org/10.1016/0166-218X(90)90024-7. URL http://www.sciencedirect.com/science/article/pii/0166218X90900247.

Lynch, J. F. 1975. The Equivalence of Theorem Proving and the Interconnection Problem. *ACM SIGDA Newsletter* 5(3): 31–36. ISSN 0163-5743. doi:10.1145/1061425.1061430. URL http://doi.acm.org/10.1145/1061425.1061430.

Middendorf, M.; and Pfeiffer, F. 1993. On the complexity of the disjoint paths problem. *Combinatorica* 13(1): 97–107.

Robertson, N.; and Seymour, P. 1995. Graph Minors. XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B* 63(1): 65–110. ISSN 0095-8956. doi:https://doi.org/10.1006/jctb.1995.1006. URL http://www.sciencedirect.com/science/article/pii/S0095895685710064.

Scheffler, P. 1994. A Practical Linear Time Algorithm for Disjoint Paths in Graphs with Bounded Tree-width. Technical report, Technischen Universität Berlin.

Schrijver, A. 1994. Finding k Disjoint Paths in a Directed Planar Graph. *SIAM Journal on Computing* 23(4): 780–788. ISSN 0097-5397. doi:10.1137/S0097539792224061. URL http://dx.doi.org/10.1137/S0097539792224061.

Surynek, P. 2010. An Optimization Variant of Multi-Robot Path Planning Is Intractable. In Fox, M.; and Poole, D., eds., *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence, AAAI-10*. AAAI Press. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1768.

Tragoudas, S.; and Varol, Y. L. 1997. Computing Disjoint Paths with Length Constraints. In *Proceedings of 23rd International Workshop on Graph-Theoretic Concepts in Computer Science*, 375–389. Springer. ISSN 978-3-540-62559-9. doi:https://doi.org/10.1007/3-540-62559-3_30.

van der Holst, H.; and de Pina, J. 2002. Length-Bounded Disjoint Paths in Planar Graphs. *Discrete Applied Mathematics* 120(1): 251–261. ISSN 0166-218X. doi:https://doi.org/10.1016/S0166-218X(01)00294-3. URL http://www.sciencedirect.com/science/article/pii/S0166218X01002943. Special Issue devoted to the 6th Twente Workshop on Graphs and Combinatorial Optimization.

Verdière, E. C. D.; and Schrijver, A. 2011. Shortest Vertex-disjoint Two-face Paths in Planar Graphs. *ACM Transactions on Algorithms* 7(2): 19:1–19:12. ISSN 1549-6325. doi:10.1145/1921659.1921665. URL http://doi.acm.org/10.1145/1921659.1921665.

Wu, B. Y. 2012. On the Maximum Disjoint Paths Problem on Edge-Colored Graphs. *Discrete Optimization* 9(1): 50–57. ISSN 1572-5286. doi:https://doi.org/10.1016/j.disopt.2012.01.002. URL http://www.sciencedirect.com/science/article/pii/S1572528612000035.

Yen, J. Y. 1971. Finding the K Shortest Loopless Paths in a Network. *Management Science* 17(11): 712–716. doi:10.1287/mnsc.17.11.712. URL https://ideas.repec.org/a/inm/ormnsc/v17y1971i11p712-716.html.