The Next-to-Shortest Path in Undirected Graphs with Nonnegative Weights

Cong Zhang[1] **Hiroshi Nagamochi**[1]

[1] Department of Applied Mathematics and Physics,
Graduate School of Informatics, Kyoto University,
Yoshida Honmachi, Sakyo, Kyoto 606-8501, Japan,
Email: {tyou,naga}@amp.i.kyoto-u.ac.jp

Abstract

Given an edge-weighted undirected graph and two vertices s and t, the next-to-shortest path problem is to find an st-path whose length is minimum among all st-paths of lengths strictly larger than the shortest path length. The problem is shown to be polynomially solvable if all edge weights are positive, while the complexity status for the nonnegative weight case was open. In this paper we show that the problem in undirected graphs admits a polynomial-time algorithm even if all edge weights are nonnegative, solving the open problem. To solve the problem, we introduce a common generalization of the undirected graph version and the acyclic digraph version of the k vertex-disjoint paths problem.

Keywords: algorithm, shortest path, disjoint paths, time complexity, next-to-shortest

1 Introduction

Let $G = (V, E, w)$ be an undirected/directed graph, in which w is an edge weight. Let n and m denote the number of vertices and edges in a graph G given as an input, respectively. For two vertices $u, v \in V$, a uv-path is a path from u to v (a path has no repeated vertices, otherwise it is called a walk). The length $w(P)$ of a path P is defined to be the total weight of the edges in P. For a given pair (s, t) of vertices, an st-path is a shortest st-path if its length is minimum among all st-paths in G. The *shortest path problem* asks to find a shortest st-path. The problem is one of the most fundamental and important network optimization problems, and has been well-studied, bringing numerous variations of it. For example, the k *shortest path problem* asks to generate the k shortest st-paths, which is a well-studied graph optimization problem that is encountered in numerous applications in operations research, telecommunications and VLSI design (Eppstein, 1998). For the k shortest path problem, Yen (1971) and Katoh et al. (1982) gave $O(kn(m + n \log n))$ time and $O(k(m + n \log n))$ time algorithms in digraphs and undirected graphs, respectively. Faster algorithms are known for finding k shortest walks (Eppstein, 1998). Finding the kth smallest st-path in a strict sense that requires to have k st-paths $P_1, P_2, \ldots, P_k$ with distinct lengths $w(P_1) < w(P_2) < \cdots < w(P_k)$ seems much more challenging. A next-to-shortest st-path is the second smallest st-path in this sense, i.e., an st-path whose length is minimum among st-paths whose lengths are strictly larger than that of a shortest st-path. The *next-to-shortest path problem* is to find a next-to-shortest st-path for given G, s and t, which has applications in VLSI design and in optimizing compilers for embedded systems (Lalgudi et al., 2000). The problem was first studied by Lalgudi and Papaefthymiou (1997) in digraphs. They proved that the problem with nonnegative edge weights is NP-complete, and showed that when repeated vertices are allowed there is an efficient algorithm. Polynomial-time algorithms for the problem on special undirected graphs were obtained (Barman et al., 2007; Mondal and Pal, 2006). The first polynomial algorithm for undirected graphs with positive edge weights was found by Krasikov and Noble (2004). Their algorithm runs in $O(n^3m)$ time. Afterwards, algorithms with improved time bounds were obtained (Kao et al., 2010; Li et al., 2006; Wu, 2010)

However, the complexity status of the next-to-shortest path problem in undirected graphs with nonnegative edge weights remains open. In this paper, we prove that the next-to-shortest path problem is polynomially solvable even for this case. Our approach is to derive a kind of decomposition of a given graph. However, to solve the resulting subproblem, we need to rely on an algorithm for finding 3 vertex-disjoint paths in a mixed graph (a graph with directed and undirected edges). In general digraphs, finding k vertex-disjoint paths problem is NP-hard even for $k = 2$ (Fortune et al., 1980). Since the mixed graphs in our reduction induces a DAG (directed acyclic graph) by its directed edges, we only need to find a common generalization of the result by Fortune et al. (1980) on the k vertex-disjoint paths problem in DAGs and that by Robertson and Seymour (1995) on the k vertex-disjoint paths problem in undirected graphs. We then prove that a next-to-shortest path in undirected graph with nonnegative edge weights can be found by solving a polynomial number of the 3 vertex-disjoint paths problem in a mixed graph.

The paper is organized as follows. Section 2 discusses our disjoint path problem in mixed graphs. Section 3 first reviews the known result on the positive weight case (Krasikov and Noble, 2004), and then derives the structural properties on *non-shortest st-paths* to design a polynomial-time algorithm for the nonnegative weight case. Section 4 makes some concluding remarks.

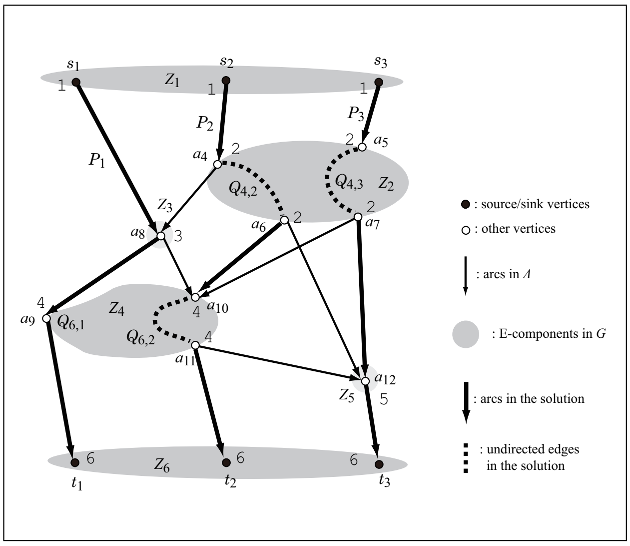

Figure 1: Illustration of a mixed graph G with six E-components Z_i, $i = 1, 2, \ldots, 6$, and a solution $\{P_1, P_2, P_3\}$ depicted by thick lines, where the number beside each vertex v denotes $\ell(v)$ and $\ell(Z_i) = i$ for each Z_i.

$\nu_1=(s_1, s_2, s_3, 1)$, $\ell(\nu_1)=1$
$\downarrow$ (i)
$\nu_2=(a_8, s_2, s_3, 1)$, $\ell(\nu_2)=1$
$\downarrow$ (i)
$\nu_3=(a_8, a_4, s_3, 1)$, $\ell(\nu_3)=1$
$\downarrow$ (i)
$\nu_4=(a_8, a_4, a_5, 1)$, $\ell(\nu_4)=2$
$\downarrow$ (ii)
$\nu_5=(a_8, a_6, a_7, 2)$, $\ell(\nu_5)=2$
$\downarrow$ (i)
$\nu_6=(a_8, a_{10}, a_7, 2)$, $\ell(\nu_6)=2$
$\downarrow$ (i)
$\nu_7=(a_8, a_{10}, a_{12}, 2)$, $\ell(\nu_7)=3$
$\downarrow$ (ii)
$\nu_8=(a_8, a_{10}, a_{12}, 3)$, $\ell(\nu_8)=3$
$\downarrow$ (i)
$\nu_9=(a_9, a_{10}, a_{12}, 3)$, $\ell(\nu_9)=4$
$\downarrow$ (ii)
$\nu_{10}=(a_9, a_{11}, a_{12}, 4)$, $\ell(\nu_{10})=4$
$\downarrow$ (i)
$\nu_{11}=(t_1, a_{11}, a_{12}, 4)$, $\ell(\nu_{11})=4$
$\downarrow$ (i)
$\nu_{12}=(t_1, t_2, a_{12}, 4)$, $\ell(\nu_{12})=5$
$\downarrow$ (ii)
$\nu_{13}=(t_1, t_2, a_{12}, 5)$, $\ell(\nu_{13})=5$
$\downarrow$ (i)
$\nu_{14}=(t_1, t_2, t_3, 5)$, $\ell(\nu_{14})=6$
$\downarrow$ (ii)
$\nu_{15}=(t_1, t_2, t_3, 6)$, $\ell(\nu_{15})=6$

Figure 2: Illustration of a sequence of states $\nu_1, \nu_2, \ldots, \nu_{14}$ which represents the solution $\{P_1, P_2, P_3\}$ in Fig. 1, where (i) (resp., (ii) beside each arrow indicated the movement (i) (resp., (ii)) is used to get the next state.

2 Disjoint Paths in Mixed Graphs

For two vertices u and v, an undirected edge joining them is denoted by $\{u, v\}$, and an arc (directed edge) that leaves u and enters v is denoted by (u, v). A graph with arcs and edges is called a *mixed graph*, denoted by $G = (V, A \cup E)$ with a set V of vertices, a set A of arcs and a set E of edges. We use $V(G)$ and $E(G)$ to denote the set of vertices and the set of arcs/edges in G, respectively. A *walk* P in G from u to v means a subgraph of G whose vertices are given by v_1 $(= u), v_2, \ldots, v_p$ $(= v)$ such that, for each $i = 1, \ldots, p-1$, P has either an arc $(v_i, v_{i+1}) \in A$ or an edge $\{v_i, v_{i+1}\} \in E$, and P has no other arc/edge, where v_1 and v_p are called the start and end vertices of P. Such walk P is denoted by $(v_1, v_2, \ldots, v_p)$. A walk in G is called a *path* if there are no repeated vertices, and is called a *cycle* if the start vertex is equal to the end vertex. A path from u to v is called a uv-path.

A connected component in the graph (V, E) with undirected edges in a mixed graph $G = (V, A \cup E)$ is called an *E-component* of G. We say that a mixed graph G is *acyclic* if there is no cycle C which can become a directed cycle by assigning orientations to all undirected edges in C, i.e., we have a DAG if we contract each E-component into a single vertex.

Given k pairs $(s_1, t_1), \ldots, (s_k, t_k)$ of vertices in a mixed graph, the k vertex-disjoint paths problem is to find k vertex-disjoint s_it_i-paths, $i = 1, \ldots, k$. We show that the problem is polynomially solvable for a fixed k in acyclic mixed graphs.

Theorem 1 *For each fixed k, there exists a polynomial-time algorithm for the k vertex-disjoint paths problem for acyclic mixed graphs.*

We prove Theorem 1 by a technical extension of the proofs for the vertex-disjoint paths problem in DAGs due to Fortune et al. (1980) and Schrijver (2003) so that it can include the result on the undirected graph version by Robertson and Seymour (1995). To prove Theorem 1, we may assume that each s_i has one leaving arc, but no other arc/edge, and each t_i has one entering arc, but no other arc/edge. We will show that the decision problem in Theorem 1 can be converted into a problem of finding a directed path between two specified vertices in an auxiliary digraph whose size is bounded by $O(n^k)$.

For a notational convenience, we treat $\{s_1, s_2, \ldots, s_k\}$ and $\{t_1, t_2, \ldots, t_k\}$ as E-components (see Z_1 and Z_6 in Fig. 1). Let $\mathcal{Z}$ denote the set of all E-components of G. Since G is acyclic, the digraph D^* obtained from the digraph (V, A) by contracting each E-component $Z \in \mathcal{Z}$ of G into a single vertex has neither directed cycle nor a self-loop. Let L be the number of vertices in D^*, and $\ell : V \to \{1, \ldots, L\}$ be a topological sort in G such that $\ell(u) < \ell(v)$ for each arc $(u, v) \in A$ and $\ell(u) = \ell(v)$ for each edge $\{u, v\} \in E$. Define $\ell(Z)$ of each E-component $Z \in \mathcal{Z}$ to be $\ell(v)$ of a vertex $v \in V(Z)$.

Firstly we show how to represent a solution $P_1, \ldots, P_k$ in a given instance G as a sequence of "movements" of k pebbles which trace the paths P_i from s_i to t_i. The current positions of k pebbles and a time-stamp defines a "state." Formally, a *state* is defined to be a $(k + 1)$-tuple $(v_1, \ldots, v_k, x)$ of distinct vertices of G and an integer $x \in \{1, \ldots, L\}$. Let $\mathcal{V}$ be the set of all states. For a state $\nu = (v_1, \ldots, v_k, x) \in \mathcal{V}$, let ℓ_ν denote the minimum of $\ell(v_i)$ over all $i = 1, \ldots, k$, and I_ν denote the set of all indices i with $\ell(v_i) = \ell_\nu$. Initially we place k pebbles, each on s_i, setting the time-stamp to be 1, which is represented by state $(s_1, \ldots, s_k, 1) \in \mathcal{V}$. We then move one or more pebbles with the minimum ℓ along the k paths $P_1, \ldots, P_k$ according the following rules (i)-(ii) until the ith pebble placed on each s_i arrives t_i along P_i.

Suppose that the ith pebble along P_i currently placed on a vertex v_i and x is the current time-stamp, which is represented by state $\nu = (v_1, \ldots, v_k, x) \in \mathcal{V}$.

(i) *Move one pebble along an arc*: If $x = \ell_\nu$, then choose an index $i \in I_\nu$, move the ith pebble on v_i to v'_i along the arc $(v_i, v'_i) \in A$ in P_i (keeping the time-stamp x unchanged).

(ii) *Move several pebbles within an E-component*: If $x < \ell_\nu$, then for all $i \in I_\nu$ move the pebble on v_i to v'_i along the maximal $v_iv'_i$-path Q_i of undirected edges in P_i (thus, these pebbles move in the E-component Z with $\ell(Z) = \ell_\nu$), and increase the time-stamp to be

ℓ_ν.

Note that in (ii) possibly $v_i = v'_i$, which simply implies that the corresponding pebble stays on the same vertex. Along the given solution, we can move all the pebbles to $t_1, \ldots, t_k$ with the time-stamp $x = L$. Hence the above rules (i)-(ii) produce a sequence of states from $(s_1, \ldots, s_k, 1)$ to $(t_1, \ldots, t_k, L)$. For example, Fig. 2 shows the resulting sequence of states from $(s_1, s_2, s_3, 1)$ to $(t_1, t_2, t_k, L = 6)$ for the solution $\{P_1, P_2, P_3\}$ in the acyclic mixed graph Fig. 1.

Next we introduce an auxiliary digraph $\mathcal{D} = (\mathcal{V}, \mathcal{A})$ on the state set $\mathcal{V}$ so that it has a directed path from $(s_1, \ldots, s_k, 1)$ to $(t_1, \ldots, t_k, L)$ if the given disjoint path problem has a solution. In $\mathcal{D}$ there is an arc from a state $\nu = (u_1, \ldots, u_k, x)$ to a state $\nu' = (v_1, \ldots, v_k, y)$ if and only if one of the following conditions (a) and (b) holds:

(a) There exists an $i \in I_\nu$ such that:
 (a-i) $\ell_\nu = x = y$;
 (a-ii) (u_i, v_i) is an arc in A; and
 (a-iii) $u_j = v_j$ for all $j \neq i$.

(b) The component Z with $\ell(Z) = \ell_\nu$ satisfies
 (b-i) $x < \ell_\nu = y$ and $I_\nu = I_{\nu'}$;
 (b-ii) graph Z has $|I_\nu|$ vertex-disjoint u_iv_i-paths Q_i, $i \in I_\nu$; and
 (b-iii) $u_j = v_j$ for all $j \notin I_\nu$.

Note that Q_i in (b-ii) may be a null path. Let $\mathcal{A}_a$ and $\mathcal{A}_b$ denote the sets of arcs (ν, ν') in $\mathcal{A}$ defined by (a) and (b), respectively.

Lemma 2 *A mixed graph G contains a solution $P_1, \ldots, P_k$, if and only if $\mathcal{D}$ contains a directed path P from $(s_1, \ldots, s_k, 1)$ to $(t_1, \ldots, t_k, L)$.*

Proof: By construction, we easily see that the "only-if" holds. We show the "if" part. Let P be a directed path from $(s_1, \ldots, s_k, 1)$ to $(t_1, \ldots, t_k, L)$ in $\mathcal{D}$. Assume that P follows the states

$$\nu_j = (v_{1,j}, \ldots, v_{k,j}, x_j) \text{ for } j = 0, \ldots, p,$$

where $x_0 = 1$, $x_p = L$, $v_{i,0} = s_i$ and $v_{i,p} = t_i$ for $i = 1, \ldots, k$. For each $i = 1, \ldots, k$, following $v_{i,j}$, $j = 0, \ldots, p$, we construct a walk P_i in G made up of the arcs/edges such that

(i) arcs $(v_{i,j}, v_{i,j+1}) \in A$ used in (a-ii) to define arcs $(\nu_j, \nu_{j+1}) \in \mathcal{A}_a$ in P;

(ii) edges in Q_i used in (b-ii) to define arcs $(\nu_j, \nu_{j+1}) \in \mathcal{A}_b$ in P.

Then P_i is a walk from s_i to t_i in G. We first show that P_i is a path, i.e., it has no repeated vertices). Since each arc $(\nu, \nu') \in \mathcal{A}_b$ increases the time-stamp of ν to ℓ_ν due to (b-i), each P_i can contain a path Q_i in the E-component Z with $\ell(Z) = \ell_\nu$ at most once. Note that any vertex v in P_i belongs to such a path Q_i, since conditions (a-i) and (b-i) imply that the time-stamp of a state needs to be $\ell(v)$ by passing through a path Q_i in the E-component Z with $\ell(Z) = \ell(v)$ before an arc $(v, v') \in A$ appears in P_i. Hence no vertex can appear repeatedly in P_i. We next claim that $P_1, \ldots, P_k$ are vertex-disjoint. To see this, suppose that two of them, say, P_1 and P_2 share a vertex $u \in V - \{s_1, \ldots, s_k, t_1, \ldots, t_k\}$, which is contained the subpath Q_i of P_i such that Q_i is the path in the E-component Z with $\ell(Z) = \ell(u)$. Again by (b-i), these Q_1 and Q_2 are generated in defining the same arc $(\nu, \nu') \in \mathcal{A}_b$ along P. This, however, contradicts that Q_1 and Q_2 are chosen to be vertex-disjoint by (b-ii). ∎

The lemma says that the k vertex-disjoint path problem in a mixed graph can be solved by testing whether the digraph $\mathcal{D}$ has a directed path P from $(s_1, \ldots, s_k, 1)$ to $(t_1, \ldots, t_k, L)$. For a fixed k, the size of $\mathcal{D}$ is polynomial since $|\mathcal{V}| \leq (n+1)^k$. For constructing each arc in $\mathcal{A}_b$, we need to find $|I_\nu|$ vertex-disjoint paths in the undirected graph Z, which can be solved in polynomial time for a fixed k (Robertson and Seymour, 1995). This proves Theorem 1.

3 Next-To-Shortest Paths

Let $G = (V, E, w)$ be an undirected graph with a vertex set V, an edge set E and a nonnegative edge weight function w. An edge of weight 0 is called a *zero-edge*, and an edge of a positive weight is called a *positive-edge*.

For a path P in G, let $w(P)$ denote the total weight of edges in P. Let $d(u, v; G)$ denote the length of a shortest uv-path in a graph G, where $d(u, v; G) = \infty$ if G has no uv-path. Let s and t be designated vertices in G. Since the edge weights are nonnegative, we have $d(s, u; G) + w(\{u, v\}) \geq d(s, v; G)$ and $d(u, t; G) + w(\{u, v\}) \geq d(v, t; G)$ for all $u, v \in V$. In particular, $d(s, u; G) = d(s, v; G)$ and $d(u, t; G) = d(v, t; G)$ for each zero-edge $\{u, v\} \in E$. For notational convenience in describing st-paths, we assume without loss of generality that s and t have only one incident edge (we add extra edges to s and t if necessary). A positive-edge is called *inner* if it is in a shortest st-path in G, and is called *outer* otherwise. Let E_0 be the set of zero-edges, E_1 be the set of inner edges $e \in E - E_0$, i.e., $E_1 = \{\{u, v\} \in E - E_0 \mid d(s, u; G) + w(\{u, v\}) = d(s, v; G),\ d(u, t; G) = w(\{u, v\}) + d(v, t; G)\}$, and E_2 denote the set $E - E_0 - E_1$ of outer edges. A path P with $E(P) \subseteq E_0 \cup E_1$ is called *pure*. Clearly every impure st-path is not a shortest st-path.

3.1 Finding Shortest Impure st-Paths

This subsection reviews the result by Krasikov and Noble (2004) to find a shortest impure st-path containing a specified outer edge. They use the next result.

Lemma 3 *Given an undirected graph $G = (V, E, w)$ with nonnegative edge weights, and specified vertices s, t and a, there is a polynomial time algorithm to find a shortest st-path passing through a.*

The problem in the lemma can be regarded as a minimum cost flow problem with flow value 2 in G with a vertex capacity 1, where a source a has demand 2 and sinks s and t have demand -1, respectively. The graph is then converted into a digraph D, where the vertex capacity is realized as an edge capacity 1. The problem can be solved by the standard method for the minimum cost flow algorithm, since any cycle in D has a nonnegative length and the cost of an optimal flow is equal to the shortest length of an st-path passing through a.

By Lemma 3, we can find a shortest st-path passing through a specified outer edge $\{u, v\} \in E_2$ by subdividing the edge with a new vertex a. Hence by solving $|E_2|$ such problems, we can find a shortest impure st-path (if any).

In what follows, we only consider how to find a shortest pure st-path of length larger than the shortest one. Hence we ignore all the outer edges unless stated otherwise.

3.2 Finding Shortest Pure st-Paths with Reversing Components

For an ordered pair (u, v) of the end vertices of an inner edge $\{u, v\} \in E_1$ is called an *forward edge* if $d(s, u; G) < d(s, v; G)$, and is called a *backward edge* otherwise. Note that a zero-edge is neither forward nor backward. Let A be the set of forward edges (u, v), $\{u, v\} \in E_1$. For a pure v_1v_k-path $P = (v_1, \ldots, v_k)$, an ordered pair (v_i, v_{i+1}) with $\{v_i, v_{i+1}\} \in E_1$ is called a forward edge of P if $(v_i, v_{i+1}) \in A$, and is called a backward edge of P otherwise. A connected component in the graph (V, E_0) with only zero-edges is called a *zero-component* of G if it contains at least one zero-edge. Let $\mathcal{Z}$ denote the set of all zero-components of G.

Lemma 4 *Let $P = (u_1, u_2, \ldots, u_k)$ be a pure path in which there is no backward edge. Then P is a shortest u_1u_k-path in G. In particular, if P contains a positive edge, then u_1 and u_k do not belong to the same zero-component Z.*

Proof: The second statement follows from the first one, since Z contains a u_1u_k-path Q with $w(Q) = 0$, implying that a u_1u_k-path P with $w(P) > 0$ cannot be a shortest one.

To show the first statement, we can assume that G has no zero-edges, since the distance of two vertices remains unchanged after contracting each zero-component into a single vertex and any path in the resulting graph corresponds a path with the same length in G.

We first observe that, for a shortest st-path $P^* = (v_1, v_2, \ldots, v_p)$, any subpath from v_i to v_j is a shortest v_iv_j-path in G, because if G has a shorter v_iv_j-path Q then we see that the union of Q and P^* contains an st-path with length shorter than P^* due to non-negativeness of edge weights.

Hence, it suffices to prove by induction that, for each u_i, $i = 2, \ldots, k$ in the path P, some shortest su_i-path P_i contains $(u_1, u_2, \ldots, u_i)$ as its subpath (for $i = k$, P is a subpath of P_k, which will be shown to be shortest). For $i = 2$, there exists such a shortest su_2-path P_2 since (u_1, u_2) is a forward edge in P. For $i = j$ $(2 \leq j < k)$, assume that there is a shortest su_j-path P_j which contains $(u_1, u_2, \ldots, u_j)$ as its subpath. Let $P'_{j+1} = [P_j, (u_j, u_{j+1})]$ be the walk from s to u_{j+1} obtained from P_j by adding edge $\{u_j, u_{j+1}\}$. There is a shortest st-path Q containing the edge $\{u_j, u_{j+1}\}$, and let Q_j (resp., Q_{j+1}) denote its subpath from s to u_j (resp., u_{j+1}). Since P_j is a shortest su_j-path by the induction hypothesis and it holds $w(P_j) \leq w(Q_j)$, we have $w(P'_{j+1}) = w(P_j) + w(\{u_j, u_{j+1}\}) \leq w(Q_j) + w(\{u_j, u_{j+1}\}) = w(Q_{j+1})$. Since there is no su_{j+1}-path shorter than Q_{j+1}, u_{j+1} cannot be a repeated vertex in P'_{j+1} (otherwise P'_{j+1} would contain such a shorter path). Hence P'_{j+1} is a desired shortest su_{j+1}-path. This completes the induction. ∎

By the lemma, a pure st-path is not a shortest st-path if and only if it has a backward edge. Hence we only need to investigate pure st-paths containing at least one backward edge.

Let $P = (v_1, \ldots, v_k)$ be a pure st-path in G. A vertex v_i with $2 \leq i \leq k-1$ is called a *reversing vertex* of P if $(v_{i-1}, v_i), (v_{i+1}, v_i) \in A$ or $(v_i, v_{i-1}), (v_i, v_{i+1}) \in A$ (i.e., (v_{i-1}, v_i) and (v_i, v_{i+1}) have different directions in the sense of forward/backward edges).

Krasikov and Noble (2004) also showed how to find a shortest pure st-path which contains a reversing vertex by using Lemma 3. We choose a pair of forward edges (u, a) and (v, a) for a vertex a, and then remove all other edges incident to a except for (u, a) and (v, a) to obtain a new graph in which vertex a has only two edges. By Lemma 3, we can find a shortest st-path P passing through a in which exactly of (u, a) and (v, a) appears as a backward edge, and vertex a is a reversing vertex. Similarly for a pair of forward edges (a, u) and (a, v), we can find a shortest st-path P passing through exactly of (a, u) and (a, v) as a backward edge. By applying the procedure for all the above pairs of forward edges, we can find a shortest pure st-path which contains a reversing vertex (if any). In fact, if a given graph G has no zero-edge, then no other case happens and this completes a proof for the fact that the next-to-shortest path problem in undirected graphs with only positive edge weights is polynomially solvable (Krasikov and Noble, 2004). On the other hand, if G has zero-edges, then the above method may find only a shortest st-path, since a zero-edge $\{u, a\}$ may appear as (u, a) and (a, u) in two shortest st-paths in G, respectively.

Let $P = (v_1, \ldots, v_k)$ be a pure st-path. A subpath Q of P is called a *zero-subpath* if Q consists of vertices and zero-edges in a zero-component Z and is maximal subject to this property (Q may contain only one vertex in Z). For each $Z \in \mathcal{Z}$, let $\rho(Z)$ denote the number of zero-subpaths Q of P such that Q is contained in Z. A zero-component Z with $\rho(Z) = 1$ is called *reversing* if its zero-subpath $Q = (v_i, v_{i+1}, \ldots, v_{j-1}, v_j)$ satisfies $(v_{i-1}, v_i), (v_{j+1}, v_j) \in A$ or $(v_i, v_{i-1}), (v_j, v_{j+1}) \in A$, and is called *trivial* otherwise. The zero-subpath of a trivial zero-component is also called *trivial*.

Finding a shortest pure st-path P which has a reversing zero-component $Z \in \mathcal{Z}$ can be computed in a similar manner with the case of reversing vertices after we contract Z into a single vertex a, which can be treated as a reversing vertex.

By definiton so far, we can classify non-shortest st-paths as follows.

Lemma 5 *Any st-path P which is not a shortest st-path in G satisfies one of the following conditions* (i)-(iv).

(i) *P is an impure path;*

(ii) *P is a pure path in which there is a reversing vertex;*

(iii) *P is a pure path in which there is a reversing zero-component; and*

(iv) *P is a pure path in which there is a backward edge, but no reversing vertex/zero-component.*

Therefore, the remaining task is to find an st-path with minimum length which satisfies condition (iv) in the lemma. We call such a path which satisfies condition (iv) *folding*. In the next subsection, we consider only folding st-paths.

3.3 Finding Shortest Folding st-Paths

In this subsection, we first examine structure of folding st-paths before we finally design an algorithm for computing a shortest folding st-path.

By definition, any folding st-path P has a zero-component Z with $\rho(Z) \geq 2$, which is called *arching*. We denote the zero-subpaths of an arching zero-component Z by $Q^1(Z), Q^2(Z), \ldots, Q^r(Z)$, $r = \rho(Z)$ in the order from s to t along P. We say that an arching zero-component Z *surrounds* a subpath P' of P if $Q^i(Z)P'Q^{i+1}(Z)$ is a subpath of P (where P' may contain a zero-edge which belongs to another zero-component Z').

Lemma 6 *Let P be a folding st-path that has the minimum length among all folding st-paths, and Z be an arching zero-component for P. Denote P by an alternating sequence of subpaths, $P = [P_1Q_1P_2 \dots Q_rP_{r+1}]$, where $Q_i = Q^i(Z)$, $i = 1, 2, \dots, r = \rho(Z)$ (each P_j may contain a zero-edge in another zero-component Z'). See Fig. 3. Then*

(i) *If Z contains a path Q connecting two zero-subpaths Q_a and Q_b ($1 \le a < b \le r$) such that Q is vertex-disjoint with any Q_i with $i < a$ or $b < i$, then all the backward edges of P appear between Q_a and Q_b along P.*

(ii) $\rho(Z) = 2$.

Proof: (i) By short-cutting with Q, we can obtain another folding st-path P'. Note that $w(P') < w(P)$ since the short-cutting skips at least one positive-edge in the subpath between Q_a and Q_b. Therefore, if P has a backward edge which does not appear between Q_a and Q_b, then P' still contains a backward edge, and hence it is a folding st-path which has shorter length than P, a contradiction. Therefore all the backward edges of P must appear between Q_a and Q_b along P.

(ii) To derive a contradiction, assume that $r \ge 3$. By applying (i) with $a = 1$ and $b = r$, we see that there is a subpath P_j with $2 \le j \le r$ which contains a backward edge. Assume without loss of generality that $j \le r - 1$ (the case of $j \ge 3$ can be treated symmetrically). To see that Q_{r-1} remains connected to some Q_h within $Z - V(Q_r)$, we consider the graph Z' obtained from Z by removing the vertices in zero-subpaths Q_i with $1 \le i \le r-2$, i.e., $Z' = Z - \cup_{1 \le i \le r-2} V(Q_i)$. In Z', let C_i, $i = r-1, r$ be the component containing Q_i (see Fig. 3). Note that $C_{r-1} \ne C_r$ since otherwise applying (i) to $a = r-1$ and $b = r$ would not allow P_j to contain a backward edge.

Now the graph $Z - V(C_r)$ contains a path Q' connecting Q_{r-1} and Q_h for some $h = 1, 2, \dots, r-2$. Hence by applying (i) with $a = h$ and $b = r - 1$, we see that P_r contains no backward edge. Since P_r contain only forward edges or zero-edges and connects two vertices in Z, this contradicts Lemma 4. Therefore $r = 2$ holds. ∎

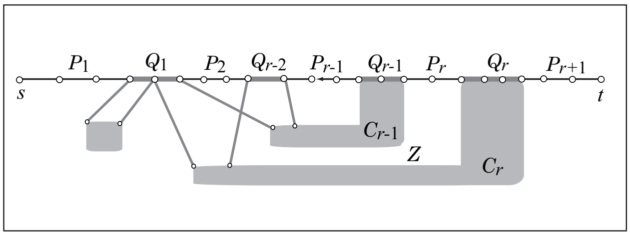

Figure 3: Illustration of a zero-component Z for a pure st-path $P = [P_1Q_1P_2 \cdots Q_rP_{r+1}]$, where each Q_i is a zero-subpath of Z.

For an st-path P, we say that two arching zero-components $Z_1, Z_2 \in \mathcal{Z}$ with $\rho(Z_1) = \rho(Z_2) = 2$ *cross* each other if for each $i = 1, 2$, the subpath between $Q^1(Z_i)$ and $Q^2(Z_i)$ contains a zero-subpath of Z_j, $j \in \{1, 2\} - \{i\}$ (see Fig. 4(a)).

Lemma 7 *Let P be a folding st-path that has the shortest length among all folding st-paths. Then*

(i) *If P has an arching zero-component, then it has another arching zero-component, and they cross each other.*

(ii) *Assume that P has $q \ge 3$ arching zero-components, then they can be indexed as Z_i, $i = 1, 2, \dots, q$ so that their zero-subpaths appear in the order $Q^1(Z_1)$, $Q^1(Z_3), Q^1(Z_4), \dots, Q^1(Z_q)$, $Q^1(Z_2), Q^2(Z_1)$, $Q^2(Z_3), Q^2(Z_4), \dots, Q^2(Z_q)$, $Q^2(Z_2)$ along P (see Fig. 5(a)).*

Proof: (i) Let Z be an arching zero-component, which has exactly two zero-subpaths by Lemma 6(ii). If the path P' surrounded by Z has no zero-subpath of another arching zero-component, then all the positive-edges in P' are backward edges (since P has no reversing vertex/zero-component), and P' connects two vertices in the same zero-component, contradicting Lemma 4. Hence P has another arching zero-component.

Next assume that there are two arching zero-components which do not cross each other. Hence P is denoted by $P = [P_1Q_1P_2Q_2P_3Q_3P_4Q_4P_5]$ such that Q_1 and Q_4 are the zero-subpaths of an arching zero-component Z_1 and Q_2 and Q_3 are those of another Z_2 (see Fig. 4(b)). By Lemma 6 applied to Z_2, there is no backward edge in the subpaths P_2 and P_4 along P from s to t. Hence the subgraph consisting of P_2, Z_2 and P_4 contains a pure path P' from the last vertex in Q_1 to the first vertex of Q_4 such that no backward edge appears along P'. Since P' connects two vertices in the same zero-component Z_1, this contradicts Lemma 4. Therefore any two arching zero-components cross each other.

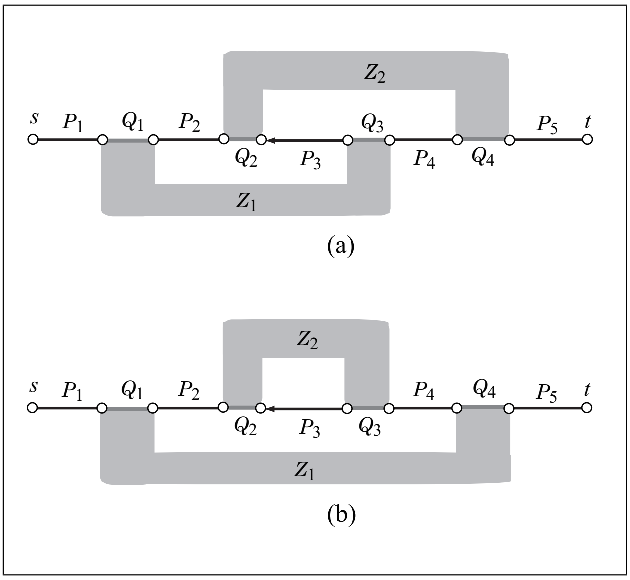

Figure 4: Illustration of two zero-components Z_1 and Z_2: (a) crossing Z_1 and Z_2: (b) non-crossing Z_1 and Z_2.

(ii) By definition, a folding st-path P is given as an alternating sequence $P_1Q_1 \cdots Q_{2q}P_{2q+1}$ of subpaths P_i and nontrivial zero-subpaths Q_i such that all positive edges in each P_i have the same direction, either forward or backward (P_i may contain trivial zero-subpaths). By definition there is at least one subpath P_j which consists of only backward edges and trivial zero-subpaths. Assume that P has three arching zero-components. Then zero-subpaths Q_{j-1} and Q_j must be contained in distinct arching zero-components, say Z_1 and Z_2, since otherwise the one containing both zero-subpaths cannot cross any other one, contradicting (i). Again by (i), Z_1 and Z_2 cross each other. The third zero-component Z_3 needs to surround P_j and cross both Z_1 and Z_2 by (i) of this lemma and Lemma 6(i). Hence the zero-subpaths of Z_1, Z_2 and Z_3 must appear in the order of $Q^1(Z_1), Q^1(Z_3), Q^1(Z_2), Q^2(Z_1), Q^2(Z_3), Q^2(Z_2)$ along P, as shown in Fig. 5(b). For other zero-components, we can assume without loss of generality that their first zero-subpaths appear in the order of $Q^1(Z_3), Q^1(Z_4), \ldots, Q^1(Z_q)$ along P. Since each Z_i crosses all $Z_1, Z_2, \ldots, Z_{j-1}$, we see that all the zero-subpaths of arching zero-components satisfy the ordering in (i). ∎

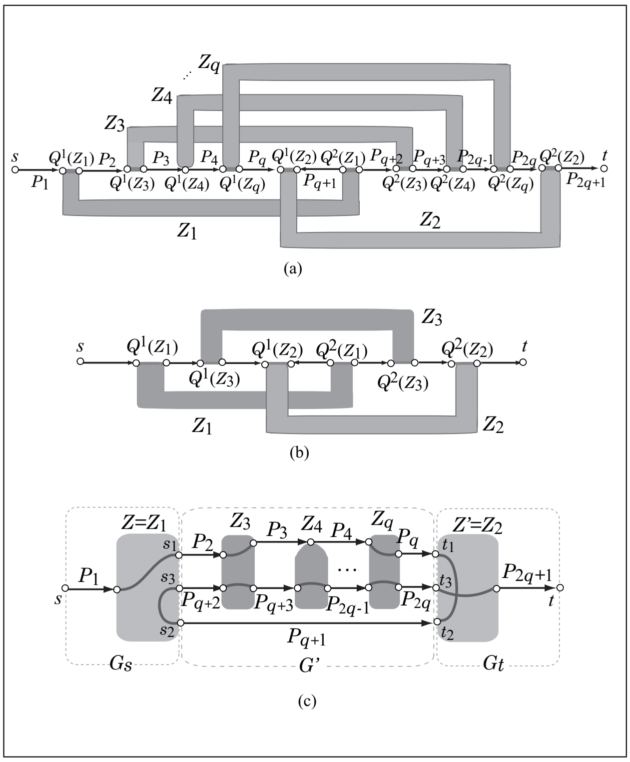

Figure 5: (a) Crossing r arching zero-components: (b) crossing three zero-components: (c) disjoint-path problem instances $(G_s, \{(s, s_1), (s_2, s_3)\})$, $(G_t, \{(t_1, t_2), (t_3, t)\})$, and $(G', \{(s_1, t_1), (s_2, t_2), (s_3, t_3)\})$.

Let us call the zero-components Z_1 and Z_2 in the lemma the *source* and *sink* components of the folding st-path P. See Fig. 5(c), which illustrates the same configuration of all arching zero-components $Z_1, \ldots, Z_r$ in Fig. 5(a).

We now show how to find a shortest folding st-path with specified source and sink components $Z, Z' \in \mathcal{Z}$. Given a shortest folding st-path P, which admits the structure in Lemma 7(ii), we let s_1, s_2 and s_3 be the initial end points of P_2, P_{q+2} and P_{q+1} and t_1, t_2 and t_3 be the last end points of P_q, P_{q+1} and P_{2q}, as shown in Fig. 5(b). Then these six vertices $s_1, \ldots, t_3$ satisfy the following conditions:

(i) In the subgraph G_s of $(V, E_0 \cup E_1)$ induced by the vertices v with $d(s, v; G) \le d(s, s_1; G)$, there are two vertex-disjoint path, ss_1-path P_{ss_1} and s_2s_3-path $P_{s_2s_3}$;

(ii) In the subgraph G_t of $(V, E_0 \cup E_1)$ induced by the vertices v with $d(s, t_1; G) \le d(s, v; G)$, there are two vertex-disjoint paths, t_1t_2-path $P_{t_1t_2}$ and t_3t-path P_{t_3t}; and

(iii) In the subgraph G' of $(V, E_0 \cup E_1)$ induced by the vertex set $\{s_1, s_2, s_3\} \cup \{v \in V \mid d(s, s_1; G) < d(s, v; G) < d(s, t_1; G)\} \cup \{t_1, t_2, t_3\}$, there are three vertex-disjoint paths, s_it_i-paths $P_{s_it_i}$, $i = 1, 2, 3$

(we treat G_s, G_t and G' as mixed graphs by regarding each positive edge $\{u, v\}$ with $d(s, u; G) < d(s, v; G)$ as an arc (u, v)). Note that the three graphs G_s, G_t and G' are vertex-disjoint except for the six vertices. We call any set of six vertices $s_1, s_2, s_3 \in V(Z)$ (possibly $s_2 = s_3$) and $t_1, t_2, t_3 \in V(Z')$ (possibly $t_1 = t_3$) satisfying the above conditions (i)-(iii) *feasible* to (Z, Z').

Lemma 8 *There is a folding st-path with source and sink components $Z, Z' \in \mathcal{Z}$ if and only if there is a feasible set of vertices $s_1, s_2, s_3 \in V(Z)$ and $t_1, t_2, t_3 \in V(Z')$.*

Proof: We have observed the "only if" part. We show the "if" part. Given a feasible set of six vertices and disjoint paths in (i)-(iii), a folding st-path P can be obtained as the concatenation

$$P_{ss_1} P_{s_1t_1} P_{t_1t_2} \overline{P_{s_2t_2}} P_{s_2s_3} P_{s_3t_3} P_{t_3t},$$

where $\overline{P_{s_2t_2}}$ denotes the t_2s_2-path obtained from $P_{s_2t_2}$ by reversing the direction. Note that the length of P (if any) is always given by $w(P) = d(s, t; G) + 2d(s, t_1; G) - 2d(s, s_1; G)$, indicating that P is a shortest one with the specified source and sink components Z and Z'. The resulting path P may pass though arching zero-components in a different way from the configuration in Lemma 6(ii) (for example, an arching zero-component may have one of its zero-subpaths in $P_{s_2t_2}$). However, it is always a folding st-path with the source and sink components Z and Z'. ∎

For each choice of such six vertices, we can determine whether such disjoint paths in (i)-(iii) exist or not in polynomial time by using Theorem 1 with $k \le 3$. Since the total number of all possible choices of source and sink components and six vertices in them is $O(n^6)$, we can find a shortest folding st-path (if any) in polynomial time. The algorithm based on the proof of Lemma 8 is described as follows.

Algorithm SHORTEST-FOLDING-PATHS
Input: The graph $(V, E_0 \cup E_1)$ for an undirected graph $G = (V, E)$ with a nonnegative edge weight w, and two vertices $s, t \in V$.
Output: A shortest folding st-path in G (if exists).
for each ordered pair of zero-components $Z, Z' \in \mathcal{Z}$ **do**
  if there is a feasible set of six vertices $s_1, s_2, s_3 \in V(Z)$ and $t_1, t_2, t_3 \in V(Z')$ **then**
    $\mu(Z, Z') := d(s, t; G) + 2d(s, t_1; G) - 2d(s, s_1; G)$;
    Let P_{ss_1} and $P_{s_2s_3}$ be vertex-disjoint ss_1-path and s_2s_3-path in G_s in (i);
    Let $P_{t_1t_2}$ and P_{t_3t} be vertex-disjoint t_1t_2-path and t_3t-path in G_t in (ii);
    Let $P_{s_it_i}$, $i = 1, 2, 3$ be vertex-disjoint s_it_i-paths in G' in (iii);
    Let $P_{(Z,Z')} := [P_{ss_1} P_{s_1t_1} P_{t_1t_2} \overline{P_{s_2t_2}} P_{s_2s_3} P_{s_3t_3} P_{t_3t}]$;
  else
    $\mu(Z, Z') := \infty$
  endif

endfor;
$(Z^*, Z^{**}) := \text{argmin}\{\mu(Z, Z') \mid Z, Z' \in \mathcal{Z}\}$;
Output $P_{(Z^*, Z^{**})}$ if $\mu(Z^*, Z^{**}) < \infty$, or
report that G has no folding st-path otherwise.

From the arguments in this and previous subsections, we finally obtain the next result.

Theorem 9 *The next-to-shortest path problem in undirected graphs with nonnegative edge weights can be solved in polynomial time.*

4 Concluding Remarks

In this paper, we showed that the next-to-shortest path problem in undirected graphs with nonnegative edge weights can be solved by reducing the problem to the k vertex-disjoint paths problem in acyclic mixed graphs with a fixed $k \leq 3$. A natural question in this line would be whether finding an st-path with the strictly third shortest length can be again reduced to the k vertex-disjoint paths problem with a fixed k.

References

Barman, S. C., Mondal, S., and Pal, M. (2007), An efficient algorithm to find next-to-shortest path on trapezoid graphs, Adv. Appl. Math. Anal. 2, 97–107.

Eppstein, D. (1998), Finding the k shortest paths, SIAM J. Comput. 28, 652–673.

Fortune, S., Hopcroft J. E., and Wyllie, J. (1980), The directed subgraph homeomorphism problem, Theoret. Comput. Sci. 10, 111–121.

Katoh, N., Ibaraki, T., and Mine, H. (1982), An efficient algorithm for K shortest simple paths, Networks 12, 411–427.

Kao, K.-H., Chang, J.-M., Wang, Y.-L., and Juan, J. S.-T. (2010), A quadratic algorithm for finding next-to-shortest paths in graphs, Algorithmica 61, 402–418.

Krasikov, I., and Noble, S. D. (2004), Finding next-to-shortest paths in a graph, Inf. Process. Lett. 92, 117–119.

Lalgudi, K. N., and Papaefthymiou, M. C. (1997), Computing strictly-second shortest paths, Inf. Process. Lett. 63, 177–181.

Lalgudi, K. N., Papaefthymiou, M. C., and Potkonjak, M. (2000), Optimizing systems for effective block-processing, ACM Trans. on Design Automation of Electronic Systems 5, 604–630.

Li, S., Sun, G., and Chen, G. (2006), Improved algorithm for finding next-to-shortest paths, Inf. Process. Lett. 99, 192–194.

Mondal, S., and Pal, M. (2006), A sequential algorithm to solve next-to-shortest path problem on circular-arc graphs, J. Phys. Sci. 10, 201–217.

Robertson, N., and Seymour, P. D. (1995), Graph minors. XIII. the disjoint paths problem, J. Combinatorial Theory, Series B 63, 65–110.

Schrijver, A. (2003), Combinatorial Optimization, Polyhedra and Efficiency, Springer-Verlag, Berlin.

Wu, B.-Y. (2010), A simpler and more efficient algorithm for the next-to-shortest path problem, W. Wu and O. Daescu (Eds.): COCOA 2010, Part II, LNCS 6509, 219–227.

Yen, J.-Y. (1971), Finding the K shortest loopless paths in a network, Manag. Sci. 17, 712–716.