Journal of the Operations Research Society of Japan
Vol. 56, No. 2, June 2013, pp. 137–154

DIJKSTRA-BASED ALGORITHMS FOR THE SHORTEST PATH PROBLEM WITH EDGES OF NEGATIVE LENGTH

Akira Nakayama
Fukushima University

Tsutomu Anazawa
Kurume University

(Received March 5, 2012; Revised February 22, 2013)

Abstract In this paper, we propose three $O(n_0 S(m,n))$ algorithms for finding the shortest paths from a vertex in a conservative network with edges of negative length, where $S(m,n)$ is the complexity of Dijkstra's method for a digraph with m edges and n vertices, and n_0 is the number of vertices incident to negative edges. Our study is motivated by S. Fujishige's method, which has a running time that is similar to that of our proposed method. He dealt with the problem of, when the given length function is decreased, updating the shortest paths from all the vertices in a vertex set to all of the given vertices, which is a problem slightly different from ours. His contribution was to solve the problem by applying a simple Dijkstra's method with less initial data than those of a previously proposed algorithm by S. Goto and A. Sangiovanni-Vincentelli. Fujishige introduced the subproblem of computing the shortest paths from a vertex v, subject to the constraint that negative edges are only contained in edges incident to v, and pointed out that the problem can be solved by consecutive applications of Dijkstra's method to a series of such subproblems, given the length of the shortest path between each pair of vertices as *prior information*. Our approaches are more general purpose than Fujishige's. In fact, ours are not limited to only updating the shortest paths, but also require no such prior information. Each of our algorithms also makes use of the concept of reduced length functions that he employed. We further show that when our algorithms incorporate an additional routine, they only apply Dijkstra's method at most $n_0/2$ times for a specific class of instances whose subgraphs induced by negative edges are undirected forests.

Keywords: Combinatorial optimization, algorithm, shortest path, Dijkstra's method

1. Introduction

We propose in this paper some Dijkstra-based algorithms for the shortest path problem with edges of negative length. Before presenting our results, we present a brief survey on the problem, state the motivation for our study, and describe some terminology and notation of graph theory.

Fundamentally, the shortest path problem is the problem of finding the path with the minimum total length between two vertices . There are a lot of variations of the problem, which are, roughly speaking, divided into five categories: (1) finding a shortest path from a particular vertex, called *source*, to every vertex ([4]); (2) finding the shortest paths between all pairs of vertices ([1],[14]); (3) finding the second, third, $\cdots$, and k-th shortest paths between two vertices ([10]); (4) finding a shortest path with specific conditions ([8], [13], [20], [19]); and (5) finding a shortest path in a specific network class ([17],[21]). Many experiments to evaluate various algorithms for the shortest path problem have been reported, such as in [11]. One excellent survey is [3], and see [1] for some examples of applications. Among the five categories, we focus on category (1). Major algorithms in (1) are divided into two groups: so-called *label-setting* and *label-correcting* approaches. A well-known algorithm in the label-setting case is Dijkstra's ([4]), which was devised for a network with

a nonnegative length function. The fastest strongly polynomial-time algorithm for such a network is Dijkstra's with Fibonacci heap implementation ([1], [18]), which has complexity $O(m + n \log n)$ where n is the number of vertices and m is the number of edges. Recently, S. Peyer, D. Rautenbach, and J. Vygen [16] discussed a generalization of Dijkstra's shortest path algorithm and presented applications to very-large-scale integration (VLSI). In 2010, Y. Dinitz and R. Itzhak ([5]) considered the single-source cheapest-path problem in a digraph with negative edge costs allowed, and they presented a hybrid of the Bellman–Ford and Dijkstra algorithms that had an improved running time compared with that of the Bellman–Ford algorithm for graphs with a sparse distribution of negative cost edges. On the other hand, the fastest strongly polynomial-time label-correcting algorithms have complexity $O(mn)$ when instance networks are allowed to have edges of negative length. See Moore [15], Bellman [2], and Ford [6] in addition to [18] for details. P.N. Klein et al. [12] considered the shortest paths in directed planar graphs with negative lengths.

In this paper, we propose three $O(n_0 S(m, n))$ algorithms for the shortest path problem in a conservative network with edges of negative length, where $S(m, n)$ is the complexity of Dijkstra's method for a digraph with m edges and n vertices, and n_0 is the number of vertices incident to negative edges. Our study is motivated by S. Fujishige's method [7], which has a running time that is similar to that of our proposed method. His method updates the shortest paths from all vertices in $V(G)$ to all vertices in a given subset U of $V(G)$ when a given length function l is decreased to $\hat{l}$, where G is a digraph and the functions $l, \hat{l}$ are defined on the edge set $E(G)$. The problem he considered was previously investigated by S. Goto and A. Sangiovanni-Vincentelli [9]. His contribution was to solve the problem by applying a simple Dijkstra's method with less initial data than those required by the algorithm by Goto et al. [9]. Fujishige introduced the subproblem of computing the shortest paths from a vertex v subject to a constraint that the decreased edges are only contained in a *star* $\delta_G(v) := \{(v, w) \in E(G) : w \in V(G)\} \cup \{(w, v) \in E(G) : w \in V(G)\}$, and he observed that we may have $\hat{l}_p(e) < 0$ only for e in the star $\delta_G(v)$ if the reduced lengths $l_p(e)$ $(e \in E(G))$ are nonnegative, where p is a potential function. He pointed out that the problem can be solved by consecutive applications of Dijkstra's method to a series of such subproblems, given the length of a shortest path between each pair of vertices as *prior information*. But he did not present any description of an algorithm that solved the problem. His algorithm runs in $O(kS(m, n))$ time, where k is the number of such stars. While each of our proposed algorithms also uses a similar technique of a reduced length function, our approaches are more general purpose than Fujishige's. In fact, ours are not limited to only updating the shortest paths, and such prior information is not necessary. Further, we show that our algorithms that incorporate an additional routine only apply Dijkstra's method at most $n_0/2$ times for a specific class of instances whose subgraphs induced by negative edges are undirected forests.

For a rigorous discussion, we introduce here the terminology and notation used in this paper. A *digraph* is a pair (V, E) of a vertex set V and an edge set E. A digraph is *simple* if each edge is defined uniquely as an ordered pair of two distinct vertices. Throughout this paper, we deal only with simple digraphs, so we will call them graphs. For a graph G, the vertex (resp. edge) set of G is denoted by $V(G)$ (resp. $E(G)$). For an edge $e := (v, w) \in E(G)$, let $\partial_G^+(e) := v$ and $\partial_G^-(e) := w$. We define $\delta_G(v) := \delta_G^+(v) \cup \delta_G^-(v)$ for a vertex $v \in V(G)$, where $\delta_G^\pm(v) := \{e \in E(G) : \partial_G^\pm(e) = v\}$, respectively. For a vertex $v \in V(G)$, let $\Gamma_G^\pm(v)$ be respectively defined by $\{\partial_G^\mp(e) \in V(G) : e \in \delta_G^\pm(v)\}$ and $\Gamma_G(v) := \Gamma_G^+(v) \cup \Gamma_G^-(v)$. For a graph G and $E' \subseteq E(G)$, $G[E'] := (V', E')$ is a subgraph of G *induced* by E' where $V' := \bigcup_{e \in E'} \{\partial_G^+(e), \partial_G^-(e)\}$. For a subgraph H' of a graph H and a function $\rho : E(H) \to \mathbb{R}$,

let $\rho(H') := \sum_{e \in E(H')} \rho(e)$ and let $\rho|E(H')$ be a *restriction* of ρ to $E(H')$. Given a graph G, consider a sequence $W := (v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1})$ composed of $v_1, \ldots, v_{k+1} \in V(G)$ and $e_1, \ldots, e_k \in E(G)$ such that $e_i = (v_i, v_{i+1})$ $(i \in \{1, \ldots, k\})$ and $v_i \neq v_j$ for all i, j with $1 \leq i < j \leq k$. Also, consider a subgraph $G' := (\{v_1, \ldots, v_{k+1}\}, \{e_1, \ldots, e_k\})$ of G. If $v_1 \neq v_{k+1}$ and $k \geq 1$ (resp. $v_1 = v_{k+1}$ and $k \geq 2$), then G' is a *path* (resp. *cycle*) in G, and W is the *trail* of G'. When $G' = G$, the phrase "in G" is omitted. If G' is a path with the trail W, then we say that G' is a path from v_1 to v_{k+1}, G' is a v_1-v_{k+1}-path, and v_{k+1} is reachable from v_1 in G'. For a path P, let $P_{[x,y]}$ be an x-y-path with $E(P_{[x,y]}) \subseteq E(P)$ (i.e., a subpath of P from x to y). For a graph G and an edge $(v, w) \in E(G)$ such that $(w, v) \notin E(G)$, *reversing* an edge (v, w) is to construct a graph $(V(G), E(G) \cup \{(w, v)\} \setminus \{(v, w)\})$. Consider a graph P with $\delta_P(x) = \delta_P(y) = 1$ and $\delta_P(v) = 2$ $(v \in V(P) \setminus \{x, y\})$ for some $x, y \in V(P)$ $(x \neq y)$. If we can obtain an x-y-path by reversing some edges in $E(P)$, then P is called an *undirected path* between x and y. An *undirected cycle* is similarly defined. A graph G is *connected* if there exists an undirected path between x and y for any distinct vertices $x, y \in V(G)$. The maximal connected subgraphs of a graph G are *connected components* of G. A graph without undirected cycles is an *undirected forest*. A connected undirected forest is an *undirected tree*. A *network* $\mathcal{N} := (G, l)$ is composed of a graph G and a *length* function $l : E(G) \to \mathbb{R}$ where $\mathbb{R}$ is the set of real numbers. The *length* of a path P in $\mathcal{N}$ is $l(P) := \sum_{e \in E(P)} l(e)$. The length of a cycle is defined similarly. If $l(C) < 0$ for a cycle C in G, then C is called *negative*. $\mathcal{N} := (G, l)$ is *conservative* if $\mathcal{N}$ has no negative cycles.

In section 2, we take an approach to our problem that uses an enlarged network. Section 3 improves the results of the previous section by introducing an auxiliary network in place of the enlarged network. In section 4, we point out that after incorporating an additional routine into the algorithms shown in the previous sections, they iterate Dijkstra's method at most $n_0/2$ times for a specific class of instances whose subgraphs induced by negative edges are undirected forests.

2. An Approach That Uses an Enlarged Network

We now begin our study. Before introducing an enlarged network, we first define a reduced network, which is a fundamental tool for various problems in network theory.

2.1. A reduced network

Given a network $\mathcal{N} := (G, l)$, a function $\pi : V(G) \to \mathbb{R}$, and an edge $e := (v, w) \in E(G)$, we call $l_\pi(e) := l(e) + \pi(v) - \pi(w)$ a *reduced length* of e, $l_\pi : E(G) \to \mathbb{R}$ a *reduced length function*, and $\mathcal{N}_\pi := (G, l_\pi)$ a *reduced network* with respect to π. An example of a network $\mathcal{N}$ with π and its reduced network $\mathcal{N}_\pi$ are shown in Figure 1, where $\pi(v)$ $(v \in V(G))$ and $l(e)$ $(e \in E(G))$ are attached in the left-hand-side network, while $l_\pi(e)$ $(e \in E(G))$ are given in the right-hand-side network. $\mathcal{N}_\pi$ is a *conservative reduced network* if $\mathcal{N}_\pi$ has no negative cycles with respect to l_π. The following propositions and a corollary are easy to see.

Proposition 2.1. For a network $\mathcal{N} := (G, l)$ and a function $\pi : V(G) \to \mathbb{R}$, we have (i) and (ii) for each pair $(x, y) \in V(G)^2$ with $x \neq y$.

(i) If there is an x-y-path P in G, then we have $l_\pi(P) = l(P) + \pi(x) - \pi(y)$.

(ii) P is a shortest x-y-path in $\mathcal{N}$ if and only if P is a shortest x-y-path in a reduced network $\mathcal{N}_\pi$.

□

Proposition 2.2. Let $\mathcal{N}_\pi := (G, l_\pi)$ be a reduced network. For any cycle C in G, we have $l_\pi(C) = l(C)$. □

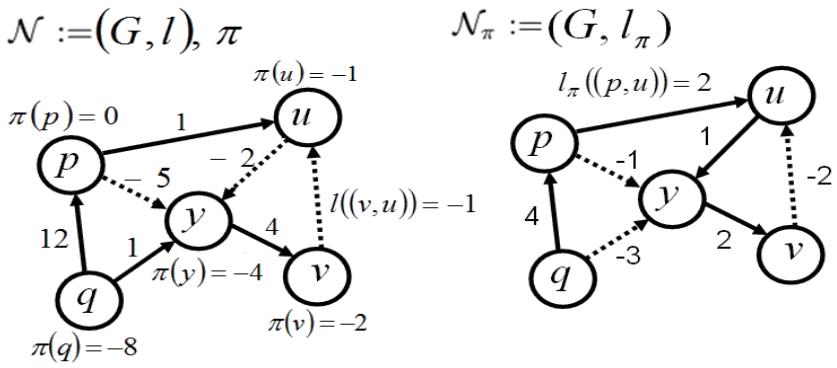

Figure 1: A network $\mathcal{N} := (G, l)$ with π and its reduced network $\mathcal{N}_\pi := (G, l_\pi)$ (dotted lines denote edges with negative length)

Corollary 2.1. Let $\pi : V(G) \to \mathbb{R}$ be any function. The reduced network $\mathcal{N}_\pi := (G, l_\pi)$ is conservative if and only if $\mathcal{N} := (G, l)$ is conservative. □

2.2. An enlarged network

Next, we define an *enlarged network* $(\hat{G}, \hat{l})$ of a reduced network $\mathcal{N}_\pi := (G, l_\pi)$.

Definition 2.1. Let $\mathcal{N}_\pi := (G, l_\pi)$ be a reduced network, and $\Upsilon_\pi := \{e \in E(G) : l_\pi(e) < 0\}$. For a vertex $y \in V(G)$, let $\Upsilon^-_{\pi,y} := \Upsilon_\pi \cap \delta^-_G(y)$, $\Upsilon^+_{\pi,y} := \Upsilon_\pi \cap \delta^+_G(y)$, $\Upsilon_{\pi,y} := \Upsilon^-_{\pi,y} \cup \Upsilon^+_{\pi,y}$, $\overline{\Upsilon_{\pi,y}} := \Upsilon_\pi \setminus \Upsilon_{\pi,y}$, and $\partial^-_G(\Upsilon^+_{\pi,y}) := \{\partial^-_G(e) \in V(G) : e \in \Upsilon^+_{\pi,y}\}$. Suppose $\Upsilon^-_{\pi,y} \neq \emptyset$ and $\Upsilon^+_{\pi,y} \neq \emptyset$. We define a graph $\hat{G}$ and a length function $\hat{l} : E(\hat{G}) \to \mathbb{R}$ as follows:

$$
\begin{aligned}
V(\hat{G}) &:= V(G) \cup \{\hat{s}\},\\
E(\hat{G}) &:= E(G) \cup \{(\hat{s}, v) : v \in V(G) \setminus \partial^-_G(\Upsilon^+_{\pi,y})\},\\
\hat{l}(e) &:= \begin{cases} l_\pi(e) & (e \in E(G) \setminus \overline{\Upsilon_{\pi,y}})\\ c_e \geq 0 & (e \in \overline{\Upsilon_{\pi,y}})\\ |l_\pi(e^+)| & (e = (\hat{s}, y))\\ |l_\pi(e^-)| + |l_\pi(e^+)| & (e = (\hat{s}, v), v \in V(G) \setminus (\{y\} \cup \partial^-_G(\Upsilon^+_{\pi,y}))) \end{cases},
\end{aligned}
$$

where $\hat{s} \notin V(G)$ is a new vertex called a *super source*, $e^\pm$ are respectively edges in $\Upsilon^\pm_{\pi,y}$ such that $|l_\pi(e^\pm)| = \max_{e \in \Upsilon^\pm_{\pi,y}} |l_\pi(e)|$, $(\hat{s}, v)$ $(v \in V(G) \setminus \partial^-_G(\Upsilon^+_{\pi,y}))$ are additional edges, and $c_e \geq 0$ $(e \in \overline{\Upsilon_{\pi,y}})$ are arbitrarily given. We call $\hat{\mathcal{N}} := (\hat{G}, \hat{l})$ an *enlarged network* with respect to y. Moreover, we define a graph $\hat{G}'$ and a length function $\hat{l}' : E(\hat{G}') \to \mathbb{R}$ as follows:

$$
\begin{aligned}
V(\hat{G}') &:= V(\hat{G}),\\
E(\hat{G}') &:= E(\hat{G}) \cup E' \setminus \Upsilon_{\pi,y},\\
\hat{l}'(e) &:= \begin{cases} \hat{l}(e) & (e \in E(\hat{G}) \setminus \Upsilon_{\pi,y})\\ |l_\pi(e^+)| + l_\pi((y,v)) & (e := (\hat{s}, v) \in E') \end{cases},
\end{aligned}
$$

where $E' := \{(\hat{s}, v) : v \in \partial^-_G(\Upsilon^+_{\pi,y})\}$. We call $\hat{\mathcal{N}}' := (\hat{G}', \hat{l}')$ a *related network* with $\hat{\mathcal{N}}$. □

This definition is motivated by the following approach (Korte and Vygen [14]) of finding a feasible reduced length function of a conservative network: Add a new source $\hat{s}$ to G and edges of zero length from $\hat{s}$ to all vertices in $V(G)$. Denote the resulting network by (G', l'). Then we can obtain such a desired function by computing the distances of the shortest paths in G' from $\hat{s}$ with respect to l'.

Remark We assume $\Upsilon^-_{\pi,y} \neq \emptyset$ and $\Upsilon^+_{\pi,y} \neq \emptyset$ throughout this paper. Similar discussions will be omitted for the remaining cases, $(\Upsilon^+_{\pi,y} \neq \emptyset, \Upsilon^-_{\pi,y} = \emptyset)$ and $(\Upsilon^-_{\pi,y} \neq \emptyset, \Upsilon^+_{\pi,y} = \emptyset)$. Examples of three networks $\mathcal{N}_\pi, \hat{\mathcal{N}}$, and $\hat{\mathcal{N}}'$ are shown in Figure 2. The following observation comes directly from Definition 2.1.

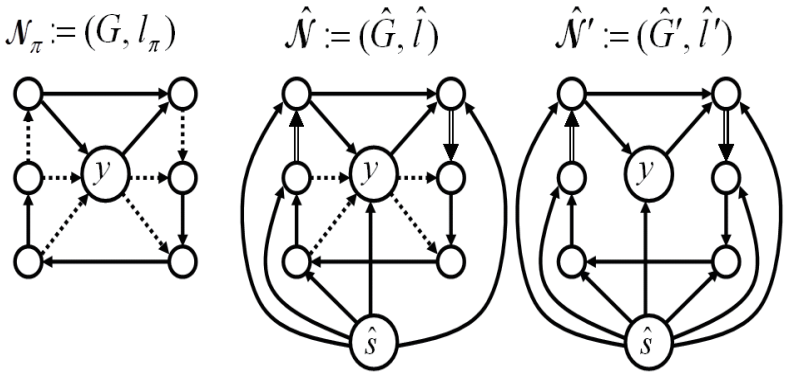

Figure 2: Networks $\mathcal{N}_\pi := (G, l_\pi)$, $\hat{\mathcal{N}} := (\hat{G}, \hat{l})$, and $\hat{\mathcal{N}}' := (\hat{G}', \hat{l}')$ (dotted lines denote edges with negative length, and double lines denote edges with any nonnegative lengths)

Observation There are $\hat{s}$-y-paths Q_1 and Q_2 in $\hat{\mathcal{N}}$ such that $E(Q_1) = \{(\hat{s}, \partial^+_G(e^-)), e^-\}$, $E(Q_2) = \{(\hat{s}, y)\}$, and $\hat{l}(Q_i) = |l_\pi(e^+)|$ $(i = 1, 2)$, where e^- and e^+ are defined in Definition 2.1. □

In addition to the observation, the following claim holds.

Claim 2.1. Any $\hat{s}$-y-path Q in $\hat{\mathcal{N}}$ satisfies $\hat{l}(Q) \geq |l_\pi(e^+)|$.

Proof. Let $(\hat{s}, w)$ be an edge in $E(Q)$ for some $w \in V(G)$. If $w = y$, then it is easy to see that the claim holds with equality. Otherwise, if $V(Q) \cap \partial^+_G(\Upsilon^-_{\pi,y}) = \emptyset$ then we have $\hat{l}(Q) \geq |l_\pi(e^-)| + |l_\pi(e^+)|$ else $\hat{l}(Q) \geq |l_\pi(e^-)| + |l_\pi(e^+)| + l_\pi((x', y)) \geq |l_\pi(e^+)|$, where $\{x'\} = V(Q) \cap \partial^+_G(\Upsilon^-_{\pi,y})$. Hence, we have $\hat{l}(Q) \geq |l_\pi(e^+)|$, i.e., the claim holds. □

Proposition 2.3. For a vertex $v \in V(G)$, let P be a shortest $\hat{s}$-v-path in $\hat{\mathcal{N}}$. Then we obtain

$$\hat{l}(P) \begin{cases} = |l_\pi(e^+)| + l_\pi((y,x)) + \hat{l}(P_{[x,v]}) & (\Upsilon^+_{\pi,y} \cap E(P) = \{(y,x)\}) \\ = |l_\pi(e^+)| + \hat{l}(P_{[y,v]}) & (\Upsilon^+_{\pi,y} \cap E(P) = \emptyset, y \in V(P)) \\ \geq |l_\pi(e^+)| + |l_\pi(e^-)| & (\Upsilon^+_{\pi,y} \cap E(P) = \emptyset, y \notin V(P)) \end{cases},$$

and subpaths $P_{[x,v]}$, $P_{[y,v]}$ of P are the shortest in $\hat{\mathcal{N}}$.

Proof. First, we consider the case of $\Upsilon^+_{\pi,y} \cap E(P) \neq \emptyset$. Obviously $\Upsilon^+_{\pi,y} \cap E(P) = \{(y,x)\}$ for some $x \in \partial^-_G(\Upsilon^+_{\pi,y})$. Since P is a shortest $\hat{s}$-v-path in $\hat{\mathcal{N}}$, $P_{[x,v]}$ is a shortest x-v-path in $\hat{\mathcal{N}}$, and $P_{[\hat{s},x]}$ is a shortest $\hat{s}$-x-path in $\hat{\mathcal{N}}$ with $\Upsilon^+_{\pi,y} \cap E(P_{[\hat{s},x]}) = \{(y,x)\}$. By Claim 2.1, $Q^* \in \{Q_1, Q_2\}$ in Observation is a shortest $\hat{s}$-y-path in $\hat{\mathcal{N}}$, and we obtain

$$\hat{l}(P) = \hat{l}(P_{[\hat{s},y]}) + \hat{l}((y,x)) + \hat{l}(P_{[x,v]}) = |l_\pi(e^+)| + l_\pi((y,x)) + \hat{l}(P_{[x,v]}).$$

We divide the case of $\Upsilon^+_{\pi,y} \cap E(P) = \emptyset$ into two subcases. If $y \in V(P)$, then from the observation and Claim 2.1 we have $\hat{l}(P) = |l_\pi(e^+)| + \hat{l}(P_{[y,v]})$. Otherwise $(y \notin V(P))$, P has an edge e_1 with $\partial^+_{\hat{G}}(e_1) = \hat{s}$ and $\hat{l}(e_1) = |l_\pi(e^-)| + |l_\pi(e^+)|$, and each length of the other edges is nonnegative. Hence, $\hat{l}(P) \geq |l_\pi(e^-)| + |l_\pi(e^+)|$ holds. □

The second step is to show that the shortest path problem in an enlarged network $\hat{\mathcal{N}} := (\hat{G}, \hat{l})$ is essentially equivalent to that in its related network $\hat{\mathcal{N}}' := (\hat{G}', \hat{l}')$. Note that $\hat{l}'$ is nonnegative. In order to realize our aim, observe the following lemma.

Lemma 2.1. For a vertex $v \in V(G)$, let P be a shortest $\hat{s}$-v-path in $\hat{\mathcal{N}}$, and P' a shortest $\hat{s}$-v-path in $\hat{\mathcal{N}}'$. Then $\hat{l}(P) = \hat{l}'(P')$ holds.

Proof. If $E(P') \cap E' = \emptyset$, then P' also exists in $\hat{\mathcal{N}}$, which implies $\hat{l}'(P') = \hat{l}(P') \geq \hat{l}(P)$ by the optimality of P. Here, E' is an edge set in Definition 2.1. Otherwise, let $(\hat{s}, w)$ be an edge of $E(P') \cap E'$. Then $\hat{\mathcal{N}}$ has a path Q which consists of $(\hat{s}, y)$, (y, w) and $P'_{[w,v]}$. Hence $\hat{l}'(P') = \hat{l}(Q) \geq \hat{l}(P)$ holds. Thus we obtain $\hat{l}'(P') \geq \hat{l}(P)$ in both cases. In order to prove $\hat{l}'(P') \leq \hat{l}(P)$, we consider the following three cases (i)–(iii).

(i) If $\Upsilon^+_{\pi,y} \cap E(P) = \{(y, x)\}$, we have

$$\hat{l}(P) = |l_\pi(e^+)| + |l_\pi((y, x))| + \hat{l}(P_{[x,v]}) = \hat{l}'((\hat{s}, x)) + \hat{l}'(P_{[x,v]}) \geq \hat{l}'(P')$$

by Proposition 2.3, the definition of $\hat{l}'$, and the optimality of P'.

(ii) If $\Upsilon^+_{\pi,y} \cap E(P) = \emptyset$ and $y \in V(P)$, we have

$$\hat{l}(P) = |l_\pi(e^+)| + \hat{l}(P_{[y,v]}) = \hat{l}'((\hat{s}, y)) + \hat{l}'(P_{[y,v]}) \geq \hat{l}'(P')$$

by Proposition 2.3, the definition of $\hat{l}'$, and the optimality of P'.

(iii) If $\Upsilon^+_{\pi,y} \cap E(P) = \emptyset$ and $y \notin V(P)$, P exists in $\hat{\mathcal{N}}'$. Hence we have $\hat{l}(P) = \hat{l}'(P) \geq \hat{l}'(P')$ by the optimality of P'.

Thus we obtain $\hat{l}(P) \geq \hat{l}'(P')$, which completes the proof. □

Lemma 2.1 implies the equivalence of the following two problems:

(A) Calculate the length of a shortest $\hat{s}$-v-path in $\hat{\mathcal{N}} := (\hat{G}, \hat{l})$ for each $v \in V(G)$.
(B) Calculate the length of a shortest $\hat{s}$-v-path in $\hat{\mathcal{N}}' := (\hat{G}', \hat{l}')$ for each $v \in V(G)$.

2.3. A Dijkstra-based algorithm

From the above discussion, we can propose the **ModifiedDijkstra**$(\mathcal{N}, s)$ algorithm, which iterates the following four steps: (i) choose $y \in V(G)$ incident to the edges of negative value of l_π, (ii) construct an enlarged network $\hat{\mathcal{N}}$ with respect to y, (iii) construct a network $\hat{\mathcal{N}}'$ related to $\hat{\mathcal{N}}$, and (iv) apply **Dijkstra**$(\hat{\mathcal{N}}', \hat{s})$ and update π. Here, **Dijkstra**$(\hat{\mathcal{N}}', \hat{s})$ is an implementation of Dijkstra's method with the inputs $\hat{\mathcal{N}}'$ and $\hat{s}$.

ModifiedDijkstra$(\mathcal{N}, s)$

Input: conservative network $\mathcal{N} := (G, l)$ and source $s \in V(G)$.

Output: set $\mathcal{P}$ of vertices reachable from s in G, length $d(v)$ of a shortest s-v-path in $\mathcal{N}$ and its second-to-last vertex $p(v)$ for each $v \in \mathcal{P} \setminus \{s\}$ in addition to $d(s) = 0$.

(S1) Let $\pi(v) \leftarrow 0$ for each $v \in V(G)$ and $V' \leftarrow V(G)$. If $\Upsilon_\pi = \emptyset$ then go to (S5).

(S2) Call **FindPivot**(G, Υ_π, V', y). If $V' = \emptyset$ and $\Upsilon_{\pi,y} = \emptyset$ then go to (S5).

(S3) Construct $\hat{\mathcal{N}}$ and $\hat{\mathcal{N}}'$ in Definition 2.1, where $c_e := \pi(v) - \pi(w)$ for $e := (v, w) \in \overline{\Upsilon_{\pi,y}}$.

(S4) By **Dijkstra**$(\hat{\mathcal{N}}', \hat{s})$, find length $\hat{d}'(v)$ of a shortest $\hat{s}$-v-path in $\hat{\mathcal{N}}'$ for each $v \in V(\hat{G}')$. Update π as $\pi(v) \leftarrow \pi(v) + \hat{d}'(v)$ $(v \in V(G))$, and return to (S2) after renewing l_π and Υ_π.

(S5) By **Dijkstra**$(\mathcal{N}_\pi, s)$, obtain a set $\mathcal{P}$ of vertices reachable from s in G, length $d''(v)$ $(v \in V(G))$ of a shortest s-v-path in $\mathcal{N}_\pi$ and second-to-last vertex $p(v)$ $(v \in \mathcal{P} \setminus \{s\})$.

Put $d(v) \leftarrow d''(v) + \pi(v) - \pi(s)$ $(v \in \mathcal{P})$ and $d(v) \leftarrow \infty$ $(v \in V(G) \setminus \mathcal{P})$. Stop with output $d(v)$ $(v \in V(G))$ and $p(v)$ $(v \in \mathcal{P} \setminus \{s\})$.

This algorithm calls subroutines **FindPivot** and **Dijkstra** where **Dijkstra**$(\mathcal{N}_\pi, s)$ is similar to **Dijkstra**$(\hat{\mathcal{N}}', \hat{s})$. The former subroutine is defined as follows.

FindPivot(G, Υ_π, V', y)
Input: edge set Υ_π and vertex set V'.
Output: V' and vertex y (if determined).
(S1) Let $\Upsilon_{\pi,v} \leftarrow \Upsilon_\pi \cap \delta_G(v)$ for each $v \in V'$. Find $y \in V'$ satisfying $|\Upsilon_{\pi,y}| = \max_{v \in V'} |\Upsilon_{\pi,v}|$. If $|\Upsilon_{\pi,y}| = 0$ then let $V' \leftarrow \emptyset$. Otherwise, update $V' \leftarrow V' \setminus \{y\}$ and return y.

Lemma 2.2. For each iteration of the loop (S2)–(S4) in **ModifiedDijkstra**, the following four claims hold.

(i) Immediately before (S3), we have $\pi(v) - \pi(w) \geq 0$ for each $(v, w) \in \Upsilon_\pi$.
(ii) Immediately after (S3), $\hat{\mathcal{N}} := (\hat{G}, \hat{l})$ is an enlarged network with respect to y chosen in **FindPivot** of (S2), and $\hat{\mathcal{N}}' := (\hat{G}', \hat{l}')$ is a related network with $\hat{\mathcal{N}}$.
(iii) The length $\hat{d}'(v)$ $(v \in V(G))$ of a shortest $\hat{s}$-v-path in $\hat{\mathcal{N}}'$ found in (S4) is equal to that of a shortest $\hat{s}$-v-path in $\hat{\mathcal{N}}$.
(iv) For updated values $\pi'(v) := \pi(v) + \hat{d}'(v)$ $(v \in V(G))$ in (S4), we have

$$\begin{aligned} l_{\pi'}(e) &\geq 0 \quad (e \in E(G) \setminus \overline{\Upsilon_{\pi,y}}), \\ \pi'(v) - \pi'(w) &\geq 0 \quad (e := (v, w) \in \overline{\Upsilon_{\pi,y}}), \\ \Upsilon_{\pi'} &\subseteq \overline{\Upsilon_{\pi,y}} \ (:= \Upsilon_\pi \setminus \Upsilon_{\pi,y}) \end{aligned}$$

at the end of (S4), where $l_{\pi'}(e)$ is the reduced length of e with respect to π', and $\Upsilon_{\pi'} := \{e \in E(G) : l_{\pi'}(e) < 0\}$.

Proof. In this proof, (Sx) $(x = 1, \ldots, 5)$ denote steps in **ModifiedDijkstra**. Our proof is by induction on the number k of iterations of the loop.

Consider the case of $k = 1$. Since the first iteration starts immediately after setting $\pi(v) \leftarrow 0$ $(v \in V(G))$ in (S1), and π does not change during (S2), we find that (i) is true. To show (ii), we need only check $c_e := \pi(v) - \pi(w) \geq 0$ $(e := (v, w) \in \overline{\Upsilon_{\pi,y}})$, which is obviously true from $\pi(v) = 0$ $(v \in V(G))$. We can prove (iii) from Lemma 2.1 and the equivalence of problems (A) and (B). Let us prove (iv). Note that

$$\hat{l}_{\hat{d}'}(e) := \hat{l}(e) + \hat{d}'(v) - \hat{d}'(w) \geq 0 \quad (e := (v, w) \in E(G)) \tag{1}$$

holds just after **Dijkstra** in (S4). In fact, we have $\hat{l}_{\hat{d}'}(e) = \hat{l}(e) + \hat{d}'(v) - \hat{d}'(w) = \hat{l}'(e) + \hat{d}'(v) - \hat{d}'(w) \geq 0$ for each $e := (v, w) \in E(G) \setminus \Upsilon_{\pi,y}$ from the optimality of $\hat{d}'(v)$ $(v \in V(G))$ in $\hat{\mathcal{N}}'$. Also, we have $\hat{l}_{\hat{d}'}(e) \geq 0$ for each $e \in \Upsilon_{\pi,y}$ from the optimality of $\hat{d}'(v)$ $(v \in V(G))$ in $\hat{\mathcal{N}}$. Note that the latter optimality is guaranteed by (iii). Hence, inequality (1) is true. The following discussion holds no matter what values π may take. From $\pi'(v) = \pi(v) + \hat{d}'(v)$ $(v \in V(G))$, we have $\hat{l}_{\hat{d}'}(e) = \hat{l}_{\pi'}(e) - (\pi(\partial_G^+(e)) - \pi(\partial_G^-(e)))$ $(e \in E(G))$. Moreover, it is easy to see that

$$\begin{aligned} \hat{l}_{\pi'}(e) &= \hat{l}(e) + \pi'(v) - \pi'(w) \\ &= \begin{cases} l_{\pi'}(e) + \pi(v) - \pi(w) & (e := (v, w) \in E(G) \setminus \overline{\Upsilon_{\pi,y}}) \\ \pi(v) - \pi(w) + \pi'(v) - \pi'(w) & (e := (v, w) \in \overline{\Upsilon_{\pi,y}}) \end{cases}. \end{aligned}$$

Hence, for each $e := (v, w) \in E(G) \setminus \overline{\Upsilon_{\pi,y}}$, we have

$$\hat{l}_{\hat{d}'}(e) = l_{\pi'}(e) + \pi(v) - \pi(w) - (\pi(v) - \pi(w)) = l_{\pi'}(e) \geq 0, \tag{2}$$

where the last inequality comes from (1). For each $e := (v, w) \in \overline{\Upsilon_{\pi,y}}$, we have

$$\hat{l}_{\hat{d}'}(e) = \pi(v) - \pi(w) + \pi'(v) - \pi'(w) - (\pi(v) - \pi(w)) = \pi'(v) - \pi'(w) \geq 0,$$

due to (1). $\Upsilon_{\pi'} \subseteq \overline{\Upsilon_{\pi,y}}$ $(:= \Upsilon_\pi \setminus \Upsilon_{\pi,y})$ holds from (2). These summaries show that (iv) is true.

Supposing that (i)–(iv) hold in the case of $k = i \geq 1$, we consider the $(i+1)$-th iteration. We obtain (i) directly from induction hypothesis (iv). Then, we have $\hat{l}(e) := c_e := \pi(v) - \pi(w) \geq 0$ $(e := (v, w) \in \overline{\Upsilon_{\pi,y}})$, and hence (ii) holds. It is easy to see (iii) from Lemma 2.1 and the equivalence of problems (A) and (B). We can show (iv) by the same argument stated in the case of $k = 1$. Hence, we find that (i)–(iv) hold in the case of $k = i + 1$. □

Lemma 2.3. **ModifiedDijkstra** runs in $O(n_0 S(m, n))$ time where $n_0 := |\{v \in V(G) : \exists e \in \delta_G(v), l(e) < 0\}|$ and $S(m, n)$ is the complexity of Dijkstra's method for an underlying graph with m edges and n vertices.

Proof. We assume $n_0 \geq 1$. The number of iterations of the loop is at most n_0. In fact, from Lemma 2.2, the loop continues while we have $v \in V'$ such that there is an edge e in $\delta_G(v)$ with $l_\pi(e) < 0$. In each iteration, operations except for (S4) run in $O(m)$, which is not greater than $S(m, n)$. Hence, we have this lemma. □

3. Another Approach by Using an Auxiliary Network

In the previous section, we proposed a Dijkstra-based algorithm called **ModifiedDijkstra**, which iterates the following four steps.

(i) Choose $y \in V(G)$ incident to edges $e \in E(G)$ with $l_\pi(e) < 0$.
(ii) Construct $\hat{\mathcal{N}}$ with respect to y by adding a super source $\hat{s}$ and some new edges leaving $\hat{s}$.
(iii) Construct $\hat{\mathcal{N}}'$ by deleting some edges incident to y and adding new edges leaving $\hat{s}$.
(iv) Apply **Dijkstra** to $\hat{\mathcal{N}}'$ and update π.

These steps seem quite tedious, and this complicated transformation motivates us to revise the algorithm.

3.1. Improved results

We directly used the underlying graph to improve **ModifiedDijkstra** to another Dijkstra-based algorithm, as follows.

ModifiedDijkstra1$(\mathcal{N}, s)$
Input and **Output** are the same as for **ModifiedDijkstra**.
(S1) The same as (S1) in **ModifiedDijkstra**.
(S2) The same as (S2) in **ModifiedDijkstra**.
(S3) Construct $\tilde{\mathcal{N}} := (G, \tilde{l})$ by

$$\tilde{l}(e) \leftarrow \begin{cases} l_\pi(e) + |l_\pi(e^+)| & (e \in \delta_G^+(y)) \\ l_\pi(e) & (e \in \overline{\delta_G^+(y)} \setminus (\Upsilon_{\pi,y}^- \cup \overline{\Upsilon_{\pi,y}})) \\ \pi(v) - \pi(w) & (e := (v, w) \in \Upsilon_{\pi,y}^- \cup \overline{\Upsilon_{\pi,y}}) \end{cases},$$

where e^+, $\Upsilon^-_{\pi,y}$, and $\overline{\Upsilon_{\pi,y}}$ are defined in Definition 2.1, and $\overline{\delta^+_G(y)} := E(G) \setminus \delta^+_G(y)$.

(S4) By **Dijkstra**$(\tilde{\mathcal{N}}, y)$, find the length $\tilde{d}(v)$ of a shortest y-v-path in $\tilde{\mathcal{N}}$ for each $v \in V(G) \setminus \{y\}$.

Update π as

$$\pi(v) \leftarrow \pi(v) + \begin{cases} |l_\pi(e^+)| & (v = y) \\ \tilde{d}(v) & (v \in \partial^-_G(\Upsilon^+_{\pi,y})) \\ \min\{\tilde{d}(v), |l_\pi(e^-)| + |l_\pi(e^+)|\} & (v \in V(G) \setminus (\{y\} \cup \partial^-_G(\Upsilon^+_{\pi,y}))) \end{cases}$$

where e^- and $\Upsilon^+_{\pi,y}$ are defined in Definition 2.1. Renew l_π and Υ_π and return to (S2).

(S5) The same as (S5) in **ModifiedDijkstra**.

We call the network $\tilde{\mathcal{N}} := (G, \tilde{l})$ constructed in each (S3) an *auxiliary network* with respect to y. Note that both **ModifiedDijkstra** and **ModifiedDijkstra1** call a common subroutine **FindPivot** in (S2), and it returns some chosen vertex $y \in V'$. We represent the k-th execution of (Sx) ($x = 1, 2, \cdots, 5$) in these two algorithms by the k-th (Sx).

Theorem 3.1. Let $\hat{\pi}_k$ be the function $\pi : V(G) \to \mathbb{R}$ immediately before the k-th (S2) in **ModifiedDijkstra**, and $\hat{y}_k$ the vertex $y \in V'$ selected at **FindPivot** in the k-th (S2). Also, define $\tilde{\pi}_k$ and $\tilde{y}_k$ for the k-th (S2) in **ModifiedDijkstra1** in the same way as in **ModifiedDijkstra**. For common inputs $\mathcal{N}$ and s, we can execute both algorithms simultaneously so as to satisfy $\hat{\pi}_k = \tilde{\pi}_k$ and $\hat{y}_k = \tilde{y}_k$ for each $k \geq 1$. □

A proof of this theorem is shown below. Suppose that Theorem 3.1 is true. For given inputs $\mathcal{N}$ and s, if **ModifiedDijkstra** stops after N (≥ 0) iterations of a loop (S2)–(S4), then **ModifiedDijkstra1** stops after at most N iterations. From Lemma 2.3, we have $N \leq n_0 := |\{v \in V(G) : \exists e \in \delta_G(v), l(e) < 0\}|$. By a similar discussion as in Lemma 2.3, we obtain the following:

Corollary 3.1. ModifiedDijkstra1 runs in $O(n_0 S(m, n))$ time. □

Although both algorithms have the same theoretical complexity, the implementation of **ModifiedDijkstra1** is expected to be simpler than that of **ModifiedDijkstra**.

3.2. Preliminaries

In order to prove Theorem 3.1, we introduce two intermediate networks and will summarize their properties as two lemmas.

Definition 3.1. For an enlarged network $\hat{\mathcal{N}} := (\hat{G}, \hat{l})$ with respect to y defined in Definition 2.1, let $\breve{\mathcal{N}} := (\breve{G}, \breve{l})$, where $\breve{G} := \hat{G}[E(G) \cup \{(\hat{s}, y)\}]$ and $\breve{l} := \hat{l}|E(\breve{G})$. Also, given $c_e \geq 0$ ($e \in \Upsilon^-_{\pi,y}$) arbitrarily, let $\bar{\mathcal{N}} := (\bar{G}, \bar{l})$, where $\bar{G} := (V(\breve{G}) \setminus \{\hat{s}\}, E(\breve{G}) \setminus \{(\hat{s}, y)\})$ ($= G$),

$$\bar{l}(e) := \begin{cases} \breve{l}(e) + |l_\pi(e^+)| & (e \in \delta^+_G(y)) \\ \breve{l}(e) & (e \in \overline{\delta^+_G(y)} \setminus \Upsilon^-_{\pi,y}) \\ c_e & (e \in \Upsilon^-_{\pi,y}) \end{cases},$$

$\overline{\delta^+_G(y)} := E(G) \setminus \delta^+_G(y)$, and e^+ is defined in Definition 2.1. □

Examples of $\hat{\mathcal{N}}$, $\breve{\mathcal{N}}$, and $\bar{\mathcal{N}}$ are illustrated in Figure 3. It is easy to see that $\bar{l}$ is a nonnegative function. In fact, for each $e \in \Upsilon^+_{\pi,y}$ ($\subseteq \delta^+_G(y)$), we have $\bar{l}(e) = \breve{l}(e) + |l_\pi(e^+)| = l_\pi(e) + |l_\pi(e^+)| \geq 0$ due to the definition of e^+; for each $e \in \overline{\Upsilon_{\pi,y}}$ ($\subseteq \overline{\delta^+_G(y)} \setminus \Upsilon^-_{\pi,y}$), we have $\bar{l}(e) = \breve{l}(e) = \hat{l}(e) = c_e \geq 0$ due to Definition 2.1; for each $e \in \Upsilon^-_{\pi,y}$, we have $\bar{l}(e) = c_e \geq 0$ due to Definition 3.1. Nonnegativity of $\bar{l}$ means that we can apply **Dijkstra** to $\bar{\mathcal{N}}$. We will

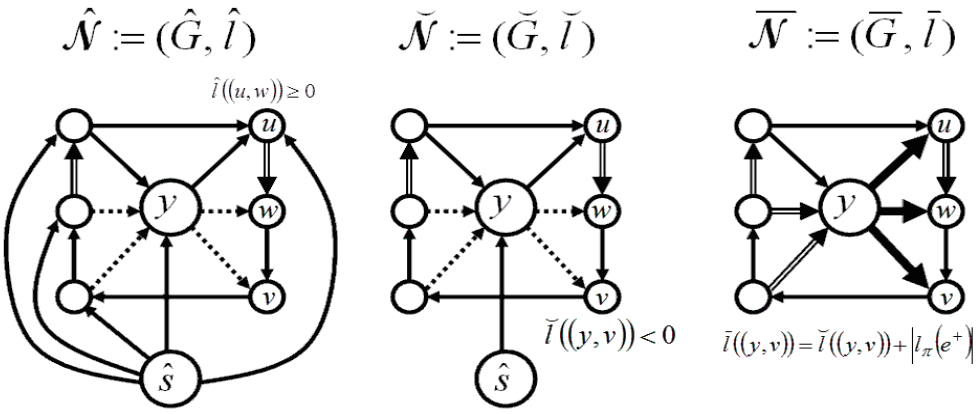

Figure 3: Three networks $\hat{\mathcal{N}}$, $\breve{\mathcal{N}}$, and $\bar{\mathcal{N}}$ (dotted, double, and thick lines respectively denote edges with negative length, edges with nonnegative length, and edges with an additional length)

also find later that $\bar{\mathcal{N}}$ is essentially equivalent to an auxiliary network $\tilde{\mathcal{N}}$. First, we show a property of $\breve{\mathcal{N}}$.

Lemma 3.1. Let $\hat{d}(v)$ (resp. $\breve{d}(v)$) be the length of a shortest $\hat{s}$-v-path for each $v \in V(G)$ in $\hat{\mathcal{N}}$ (resp. $\breve{\mathcal{N}}$). Then we have

$$\hat{d}(v) = \begin{cases} |l_\pi(e^+)| & (v = y) \\ \breve{d}(v) & (v \in \partial_G^-(\Upsilon^+_{\pi,y})) \\ \min\{\breve{d}(v), |l_\pi(e^-)| + |l_\pi(e^+)|\} & (v \in V(G) \setminus (\{y\} \cup \partial_G^-(\Upsilon^+_{\pi,y}))) \end{cases},$$

where e^- and $\Upsilon^+_{\pi,y}$ are defined in Definition 2.1.

Proof. By using Claim 2.1, let us prove the lemma. We consider the following two cases.
Case 1 ($v = y$): Let P be an arbitrary $\hat{s}$-y-path in $\hat{\mathcal{N}}$. Then $\hat{l}(P) \geq |l_\pi(e^+)|$ comes directly from Claim 2.1. Also, noting that the length of a path consisting of only an edge $(\hat{s}, y)$ is $|l_\pi(e^+)|$, we see $\hat{d}(y) = |l_\pi(e^+)|$.
Case 2 ($v \in V(G) \setminus \{y\}$): Let $\mathcal{P}_v$ be the set of all $\hat{s}$-v-paths in $\hat{\mathcal{N}}$, and $\mathcal{P}_v^y := \{P \in \mathcal{P}_v : (\hat{s}, y) \in E(P)\}$. Then

$$\hat{d}(v) = \min\left\{ \min_{P \in \mathcal{P}_v^y} \hat{l}(P), \min_{P \in \mathcal{P}_v \setminus \mathcal{P}_v^y} \hat{l}(P) \right\}$$

is obvious. Also, it is easy to see that $\breve{d}(v) = \min_{P \in \mathcal{P}_v^y} \hat{l}(P)$ holds. Hence, we have

$$\hat{d}(v) = \min\left\{ \breve{d}(v), \min_{P \in \mathcal{P}_v \setminus \mathcal{P}_v^y} \hat{l}(P) \right\}. \tag{3}$$

For any $P \in \mathcal{P}_v \setminus \mathcal{P}_v^y$, if $y \in V(P)$ then we find from $\hat{l}((\hat{s}, y)) = |l_\pi(e^+)|$ and Claim 2.1 that

$$\begin{aligned} \hat{l}(P) &= \hat{l}(P_{[\hat{s},y]}) + \hat{l}(P_{[y,v]}) \\ &\geq \hat{l}((\hat{s}, y)) + \hat{l}(P_{[y,v]}) = \breve{l}((\hat{s}, y)) + \breve{l}(P_{[y,v]}) \geq \breve{d}(v) \end{aligned} \tag{4}$$

holds. Otherwise ($y \notin V(P)$), from $\hat{l}(e) \geq 0$ ($e \in E(P_{[x,v]})$), P satisfies

$$\hat{l}(P) = \hat{l}((\hat{s}, x)) + \hat{l}(P_{[x,v]}) \geq |l_\pi(e^-)| + |l_\pi(e^+)| \tag{5}$$

for some $x \in V(G) \setminus \{y\}$. By (4) and (5), we have

$$\min_{P \in \mathcal{P}_v \setminus \mathcal{P}_v^y} \hat{l}(P) \geq \min\left\{ \breve{d}(v), |l_\pi(e^-)| + |l_\pi(e^+)| \right\}. \tag{6}$$

Noting inequality (6), we divide Case 2 into the following two subcases.
Case 2.1 ($v \in \partial_G^-(\Upsilon_{\pi,y}^+)$): From $l_\pi((y,v)) < 0$, we have

$$\breve{d}(v) \leq \hat{l}((\hat{s}, y)) + \hat{l}((y,v)) = |l_\pi(e^+)| + l_\pi((y,v)) < |l_\pi(e^+)|. \tag{7}$$

Then we find from (6) and (7) that

$$\min_{P \in \mathcal{P}_v \setminus \mathcal{P}_v^y} \hat{l}(P) \geq \min\left\{ \breve{d}(v), |l_\pi(e^-)| + |l_\pi(e^+)| \right\} = \breve{d}(v)$$

holds. By equation (3), we have $\hat{d}(v) = \breve{d}(v)$.
Case 2.2 ($v \in V(G) \setminus (\{y\} \cup \partial_G^-(\Upsilon_{\pi,y}^+))$): If $\breve{d}(v) < |l_\pi(e^-)| + |l_\pi(e^+)|$, then we obtain $\min_{P \in \mathcal{P}_v \setminus \mathcal{P}_v^y} \hat{l}(P) \geq \breve{d}(v)$ by inequality (6). By equation (3), $\hat{d}(v) = \breve{d}(v) = \min\{\breve{d}(v), |l_\pi(e^-)| + |l_\pi(e^+)|\}$ holds. Otherwise ($\breve{d}(v) \geq |l_\pi(e^-)| + |l_\pi(e^+)|$), we obtain $\min_{P \in \mathcal{P}_v \setminus \mathcal{P}_v^y} \hat{l}(P) \geq |l_\pi(e^-)| + |l_\pi(e^+)|$ by inequality (6). The fact that the length of a path consisting of only an edge $(\hat{s}, v)$ equals $|l_\pi(e^-)| + |l_\pi(e^+)|$ implies $\min_{P \in \mathcal{P}_v \setminus \mathcal{P}_v^y} \hat{l}(P) = |l_\pi(e^-)| + |l_\pi(e^+)|$. By equation (3), we have $\hat{d}(v) = \min\{\breve{d}(v), |l_\pi(e^-)| + |l_\pi(e^+)|\}$. □

Next, we show a property of $\bar{\mathcal{N}}$ derived from $\breve{\mathcal{N}}$.

Lemma 3.2. Let $\breve{d}(v)$ ($v \in V(G)$) be the length of a shortest $\hat{s}$-v-path in $\breve{\mathcal{N}}$. Also, let $\bar{d}(v)$ ($v \in V(G)$) be the length of a shortest y-v-path in $\bar{\mathcal{N}}$ derived from $\breve{\mathcal{N}}$. Then $\breve{d}(v) = \bar{d}(v)$ holds for each $v \in V(G) \setminus \{y\}$.

Proof. For an arbitrary vertex $v \in V(G) \setminus \{y\}$, let P be a shortest $\hat{s}$-v-path in $\breve{\mathcal{N}}$, and let $(v_1 = \hat{s}, e_1, v_2 = y, \ldots, v_k, e_k, v_{k+1} = v)$ be the trail of P. Noting that P does not contain any edges in $\Upsilon_{\pi,y}^-$, we find from Definition 3.1 and $\breve{l}(e_1) = \breve{l}((\hat{s}, y)) = |l_\pi(e^+)|$ that

$$\breve{d}(v) = \breve{l}(P) = |l_\pi(e^+)| + \breve{l}(e_2) + \sum_{i=3}^{k} \breve{l}(e_i) = \bar{l}(e_2) + \sum_{i=3}^{k} \bar{l}(e_i) = \bar{l}(P_{[y,v]}).$$

It is easy to see that $P_{[y,v]}$ is a shortest y-v-path in $\bar{\mathcal{N}}$ (otherwise, $\breve{\mathcal{N}}$ must have an $\hat{s}$-v-path whose length is less than $\breve{l}(P)$, which is a contradiction). Hence, we have $\breve{d}(v) = \bar{l}(P_{[y,v]}) = \bar{d}(v)$. □

3.3. Proof of Theorem 3.1

For simplicity, we abbreviate **ModifiedDijkstra** as **MD** and **ModifiedDijkstra1** as **MD1**. Note that both **MD** and **MD1** call the common subroutine **FindPivot**, from which they get the common return values. Let $\hat{V}'_k$ (resp. $\tilde{V}'_k$) be the set V' of vertices immediately after the k-th (S2) in **MD** (resp. **MD1**) where k is the number of iterations of the loop (S2)–(S4). $\hat{V}'_0$ and $\tilde{V}'_0$ denote the initial V' immediately after (S1) in these two algorithms.

Suppose that both **MD** and **MD1** are executed with common inputs $\mathcal{N}$ and s. We prove the theorem by induction on k.

Consider the case of $k = 1$. Since $\hat{\pi}_1(v) = \tilde{\pi}_1(v) = 0$ $(v \in V(G))$, we have $\hat{\pi}_1 = \tilde{\pi}_1$. Note that $\hat{V}'_0 = \tilde{V}'_0 = V(G)$ holds. **FindPivot** chooses $\hat{y}_1 \in \hat{V}'_0$ and $\tilde{y}_1 \in \tilde{V}'_0$ with $\hat{y}_1 = \tilde{y}_1$ in the respective (S2) of both algorithms. Also note that $\hat{V}'_1 = \tilde{V}'_1$ can be satisfied immediately after the first (S2).

Suppose that the following equations $\hat{\pi}_k = \tilde{\pi}_k$, $\hat{y}_k = \tilde{y}_k$, and $\hat{V}'_k = \tilde{V}'_k$, are satisfied immediately after the k-th (S2) of **MD** and **MD1** for $k \geq 1$. Consider the $(k+1)$-th iteration.

We find from Lemma 2.2(ii) that $\hat{\mathcal{N}} := (\hat{G}, \hat{l})$ given immediately after the k-th (S3) in **MD** is an enlarged network with respect to $\hat{y}_k$, where

$$\hat{l}(e) = \begin{cases} |l_{\hat{\pi}_k}(e^+)| & (e = (\hat{s}, \hat{y}_k)) \\ |l_{\hat{\pi}_k}(e^-)| + |l_{\hat{\pi}_k}(e^+)| & (e = (\hat{s}, v), v \in V(G) \setminus (\{\hat{y}_k\} \cup \partial^-_G(\Upsilon^+_{\hat{\pi}_k, \hat{y}_k}))) \\ l_{\hat{\pi}_k}(e) & (e \in E(G) \setminus \overline{\Upsilon_{\hat{\pi}_k, \hat{y}_k}}) \\ \hat{\pi}_k(v) - \hat{\pi}_k(w) & (e := (v, w) \in \overline{\Upsilon_{\hat{\pi}_k, \hat{y}_k}}) \end{cases}.$$

Note that $\hat{\pi}_k(v) - \hat{\pi}_k(w) \geq 0$ for each $(v, w) \in \overline{\Upsilon_{\hat{\pi}_k, \hat{y}_k}}$ due to Lemma 2.2 (i). Let $\hat{d}(v)$ $(v \in V(G))$ be the length of a shortest $\hat{s}$-v-path in this network $\hat{\mathcal{N}}$. Then we find from Lemma 2.2 (iii) that, immediately after the k-th (S4),

$$\hat{\pi}_{k+1}(v) = \hat{\pi}_k(v) + \hat{d}(v) \ (v \in V(G)). \tag{8}$$

For $\hat{\mathcal{N}}$ with respect to y_k, let $\breve{\mathcal{N}} := (\breve{G}, \breve{l})$ be the network defined in Definition 3.1. Also, let $\breve{d}(v)$ $(v \in V(G))$ be the length of a shortest $\hat{s}$-v-path in $\breve{\mathcal{N}}$. Then we find from Lemma 3.1 and (8) that

$$\hat{\pi}_{k+1}(v) = \hat{\pi}_k(v) + \begin{cases} |l_{\hat{\pi}_k}(e^+)| & (v = \hat{y}_k) \\ \breve{d}(v) & (v \in \partial^-_G(\Upsilon^+_{\hat{\pi}_k, \hat{y}_k})) \\ \min\{\breve{d}(v), |l_{\hat{\pi}_k}(e^-)| + |l_{\hat{\pi}_k}(e^+)|\} & (v \in V(G) \setminus (\{\hat{y}_k\} \cup \partial^-_G(\Upsilon^+_{\hat{\pi}_k, \hat{y}_k}))) \end{cases}. \tag{9}$$

Let $\bar{\mathcal{N}} := (\bar{G}, \bar{l})$ be the network defined in Definition 3.1, where $c_e := \hat{\pi}_k(v) - \hat{\pi}_k(w)$ for each $e := (v, w) \in \Upsilon^-_{\hat{\pi}_k, \hat{y}_k}$. Note that $c_e \geq 0$ for each $e \in \Upsilon^-_{\hat{\pi}_k, \hat{y}_k}$ due to Lemma 2.2 (i). Also, let $\bar{d}(v)$ $(v \in V(G))$ be the length of a shortest y-v-path in $\bar{\mathcal{N}}$. Then we find from Lemma 3.2 and (9) that

$$\hat{\pi}_{k+1}(v) = \hat{\pi}_k(v) + \begin{cases} |l_{\hat{\pi}_k}(e^+)| & (v = \hat{y}_k) \\ \bar{d}(v) & (v \in \partial^-_G(\Upsilon^+_{\hat{\pi}_k, \hat{y}_k})) \\ \min\{\bar{d}(v), |l_{\hat{\pi}_k}(e^-)| + |l_{\hat{\pi}_k}(e^+)|\} & (v \in V(G) \setminus (\{\hat{y}_k\} \cup \partial^-_G(\Upsilon^+_{\hat{\pi}_k, \hat{y}_k}))) \end{cases}. \tag{10}$$

We must also note that $\bar{l}$ can be written as

$$\bar{l}(e) = \begin{cases} l_{\hat{\pi}_k}(e) + |l_{\hat{\pi}_k}(e^+)| & (e \in \delta^+_G(\hat{y}_k)) \\ l_{\hat{\pi}_k}(e) & (e \in \overline{\delta^+_G(\hat{y}_k)} \setminus (\Upsilon^-_{\hat{\pi}_k, \hat{y}_k} \cup \overline{\Upsilon_{\hat{\pi}_k, \hat{y}_k}})) \\ \hat{\pi}_k(v) - \hat{\pi}_k(w) & (e := (v, w) \in \Upsilon^-_{\hat{\pi}_k, \hat{y}_k} \cup \overline{\Upsilon_{\hat{\pi}_k, \hat{y}_k}}) \end{cases}. \tag{11}$$

On the other hand, the length function $\tilde{l}$ obtained immediately after the k-th (S3) in **MD1** satisfies

$$\tilde{l}(e) = \begin{cases} l_{\tilde{\pi}_k}(e) + |l_{\tilde{\pi}_k}(e^+)| & (e \in \delta^+_G(\tilde{y}_k)) \\ l_{\tilde{\pi}_k}(e) & (e \in \overline{\delta^+_G(\tilde{y}_k)} \setminus (\Upsilon^-_{\tilde{\pi}_k,\tilde{y}_k} \cup \overline{\Upsilon_{\tilde{\pi}_k,\tilde{y}_k}})) \\ \tilde{\pi}_k(v) - \tilde{\pi}_k(w) & (e := (v,w) \in \Upsilon^-_{\tilde{\pi}_k,\tilde{y}_k} \cup \overline{\Upsilon_{\tilde{\pi}_k,\tilde{y}_k}}) \end{cases}.$$

From $\bar{G} = G$, (11), and the induction hypothesis ($\hat{\pi}_k = \tilde{\pi}_k$, $\hat{y}_k = \tilde{y}_k$), we find that $\tilde{\mathcal{N}} = \bar{\mathcal{N}}$ and

$$\tilde{d}(v) = \bar{d}(v) \ (v \in V(G) \setminus \{y\}), \tag{12}$$

where $\tilde{d}$ is obtained in the k-th (S4) of **MD1**. Immediately after the k-th (S4), we have

$$\tilde{\pi}_{k+1}(v) = \tilde{\pi}_k(v) + \begin{cases} |l_{\tilde{\pi}_k}(e^+)| & (v = \tilde{y}_k) \\ \tilde{d}(v) & (v \in \partial^-_G(\Upsilon^+_{\tilde{\pi}_k,\tilde{y}_k})) \\ \min\{\tilde{d}(v), |l_{\tilde{\pi}_k}(e^-)| + |l_{\tilde{\pi}_k}(e^+)|\} & (v \in V(G) \setminus (\{\tilde{y}_k\} \cup \partial^-_G(\Upsilon^+_{\tilde{\pi}_k,\tilde{y}_k}))) \end{cases}.$$

Equations (10), (12), and the induction hypothesis imply that $\hat{\pi}_{k+1} = \tilde{\pi}_{k+1}$. Similarly, we can show $\hat{V}'_{k+1} = \tilde{V}'_{k+1}$ in the $(k+1)$-th (S2). Hence, we can execute the $(k+1)$-th (S2) of both **MD** and **MD1** such that $\hat{y}_{k+1} = \tilde{y}_{k+1}$. □

4. Additional Device for a Specific Class of Instances

For a conservative network $\mathcal{N} := (G, l)$ with $\Upsilon := \{e \in E(G) : l(e) < 0\} \neq \emptyset$, let $H := G[\Upsilon]$ be a subgraph of G induced by Υ. In (S2) of **ModifiedDijkstra1**, a vertex y is chosen from $V(H)$. We will show that with a more careful choice of y, we can expect that the running time will be improved. In fact, if H is an undirected forest, then the number of executions of a revised **FindPivot**, shown later, is reduced by at least $n_0/2$, where $n_0 := |\{v \in V(G) : \exists e \in \delta_G(v), l(e) < 0\}|$.

Note that H has no isolated vertices and satisfies $|V(H)| = n_0$. We determine the order in which y is chosen by the algorithm **Ordering**, which is based on the breadth-first search (BFS) . In **Ordering**, we use two sets S and Q of vertices implemented by stack and queue, respectively. The order of choosing y is that of *popping* vertices one by one from S. This order will be incorporated in the revised version of **FindPivot** defined later. The vertices visited for the first time during BFS are stored in the set Q. **Ordering** chooses some vertices in $V(H)$ as 'roots', and puts $\rho(v) = 1$ (resp. 0) if v is a root (resp. not a root).

Ordering(H, S, ρ)
Input: induced subgraph H of G.
Output: stack S of vertices and a function $\rho : V(H) \to \{0, 1\}$.
(S1) Set $U \leftarrow V(H)$ and $S \leftarrow \emptyset$.
(S2) While $U \neq \emptyset$ do (S2.1) and (S2.2).
  (S2.1) Choose $r \in U$ and call it a *root*. Let $U \leftarrow U \setminus \{r\}$, $Q \leftarrow \{r\}$ and $\rho(r) \leftarrow 1$.
  (S2.2) While $Q \neq \emptyset$ do (S2.2.1) and (S2.2.2).
    (S2.2.1) Dequeue v from Q, and push v to S.
    (S2.2.2) For each $w \in \Gamma_H(v) \cap U$, let $U \leftarrow U \setminus \{w\}$, enqueue w to Q, and set $\rho(w) \leftarrow 0$. Then we say v is a *parent* of w.

We can observe some properties of the output S of **Ordering**. Obviously, $S = V(H)$ holds. Let C be an arbitrary connected component of H. Then each vertex in $V(C)$ is successively pushed onto S. Also, C has a unique root, say r, and r is pushed onto S first among the vertices in $V(C)$. More generally, we find from a property of the BFS that v is pushed onto S prior to w being pushed onto S for any $v, w \in V(C)$ satisfying $d_C(r, v) < d_C(r, w)$, where $d_C(p, q)$ denotes the minimum number of edges of an undirected path between p and q in C. Further, if $v \in V(C)$ is not a root, then v has a unique parent, say p, and p is pushed onto S prior to v being pushed onto S.

We define **ModifiedDijkstra2** where (S1) and (S2) in **ModifiedDijkstra1** are replaced by the following (S1)' and (S2)', respectively.

(S1)' Let $\pi(v) \leftarrow 0$ for each $v \in V(G)$. If $\Upsilon = \emptyset$, then go to (S5), otherwise let $H \leftarrow G[\Upsilon]$ and call **Ordering**(H, S, ρ).

(S2)' Call **FindPivot2**$(G, \Upsilon_\pi, H, S, \rho, y)$. If $y =$ **null** then go to (S5).

Here, **null** means that a desired vertex y cannot be determined, and **FindPivot2** (a revised version of **FindPivot**) is defined as follows:

FindPivot2$(G, \Upsilon_\pi, H, S, \rho, y)$

Input: graph G, edge set Υ_π, induced subgraph H of G, stack S of vertices, and function ρ.

Output: H, S, and vertex y.

(S1) While $S \neq \emptyset$ do (S1.1) and (S1.2).

(S1.1) Pop y from S.

(S1.2) If y satisfies **(a)** $\rho(y) = 1$ and $|\delta_H(y)| \geq 1$, or **(b)** $\rho(y) = 0$ and $|\delta_H(y)| \geq 2$, then do (S1.2.1) and (S1.2.2).

(S1.2.1) Set $T \leftarrow \delta_H(y)$ and $H \leftarrow (V(H), E(H) \setminus T)$.

(S1.2.2) If $\Upsilon_\pi \cap T \neq \emptyset$ then return y.

(S2) Set $y \leftarrow$ **null** and return y.

Note that $\Upsilon_\pi \cap T = \emptyset$ is possible at (S1.2.2) in the k-th execution of **FindPivot2** for $k \geq 2$. To consider the reason, let π_0 (resp. π_1) be the π immediately before (resp. after) the first execution of (S4) in **ModifiedDijkstra2**, and let y_1 be the y returned by the first execution of **FindPivot2**. By Lemma 2.2 (iv), we have $\Upsilon = \Upsilon_{\pi_0} \supseteq \overline{\Upsilon_{\pi_0, y_1}} \supseteq \Upsilon_{\pi_1} \supseteq \cdots$. Let T_k $(k \geq 1)$ be the T found immediately after the k-th (S1.2.1) in **FindPivot2**. It is easy to see that $T_1 = \Upsilon_{\pi_0, y_1} \subseteq \Upsilon_{\pi_0}$ (hence, $\Upsilon_{\pi_0} \cap T_1 \neq \emptyset$) and $T_2 \subseteq \overline{\Upsilon_{\pi_0, y_1}}$. However, if $\overline{\Upsilon_{\pi_0, y_1}} \setminus \Upsilon_{\pi_1} \neq \emptyset$, then it may happen that $T_2 \subseteq \overline{\Upsilon_{\pi_0, y_1}} \setminus \Upsilon_{\pi_1}$, i.e., $\Upsilon_{\pi_1} \cap T_2 = \emptyset$. In fact, such a case occurs in a numerical example shown below.

Lemma 4.1. All vertices in S are popped during **ModifiedDijkstra2**, that is, **ModifiedDijkstra2** must terminate.

Proof. Since $\Upsilon \neq \emptyset$, **FindPivot2** in (S2)' must be called at least once. If $S \neq \emptyset$ immediately before calling **FindPivot2**, then at least one vertex in S must be popped in executing **FindPivot2**. As the loop (S2)–(S4) is iterated, the number of elements in S decreases monotonically and must attain 0. □

Lemma 4.2. Let y^i and H^i $(i = 1, \ldots, n_0)$ be the y and H found immediately after the i-th execution of (S1.1) in **FindPivot2** during **ModifiedDijkstra2**. If y^i is not a root, then $|\delta_{H^i}(y^i)| \geq 1$ holds.

Proof. Let us note the following: If y^j $(j = 1, \ldots, n_0 - 1)$ satisfies condition (a) or (b) in (S1.2), then edges in $\delta_{H^j}(y^j)$ are deleted from $E(H^j)$, i.e., $E(H^{j+1}) = E(H^j) \setminus \delta_{H^j}(y^j)$. Otherwise (neither (a) nor (b) is satisfied), we have $E(H^{j+1}) = E(H^j)$. Hence, $E(H^1) \supseteq E(H^2) \supseteq \cdots \supseteq E(H^{n_0})$ holds.

Suppose that y^i is not a root. Then it has a unique parent, say p, and $p = y^l$ holds for some $l \in \{i+1, \ldots, n_0\}$. Note that there exists an edge $e^* \in E(H^1)$ satisfying $e^* = (y^i, y^l)$ or $e^* = (y^l, y^i)$. If $i = 1$, then we have $|\delta_{H^1}(y^1)| \geq 1$ since $e^* \in \delta_{H^1}(y^1)$. Consider the case of $i > 1$. We find that $e^* \notin \delta_{H^j}(y^j)$ holds for each $j \in \{1, \ldots, i-1\}$, since the initial and terminal vertices of e^* are y^i and $y^l (= p)$. Hence, we have $e^* \in E(H^i)$, from which we derive $e^* \in \delta_{H^i}(y^i)$. Therefore, we obtain $|\delta_{H^i}(y^i)| \geq 1$. □

Lemma 4.3. Suppose that **ModifiedDijkstra2** terminates after N executions of (S1.2.1) in **FindPivot2**. Let T_k $(k = 1, \ldots, N)$ be the T found immediately after the k-th (S1.2.1) in **FindPivot2**. Then we see $\Upsilon = \bigcup_{k=1}^{N} T_k$ and $T_k \cap T_l = \emptyset$ $(k, l \in \{1, \ldots, N\},\ k \neq l)$.

Proof. Let y_k and H_k $(k = 1, \ldots, N)$ be the y and H found immediately after the k-th (S1.2.1) in **FindPivot2**, respectively. Also, H_0 denotes the H found immediately before the first (S1.2.1). Note that $T_{k+1} = \delta_{H_k}(y_{k+1})$ holds for each $k = 0, 1, \ldots, N-1$. It is easy to see that $\bigcup_{k=1}^{N} T_k \subseteq \Upsilon$. In fact, it follows from $E(H_0) \supset E(H_1) \supset \cdots \supset E(H_N)$ that

$$T_k = \delta_{H_{k-1}}(y_k) \subseteq \delta_{H_0}(y_k) \subseteq E(H_0) = \Upsilon$$

for $k = 1, \ldots, N$. Let us show $\Upsilon \subseteq \bigcup_{k=1}^{N} T_k$. Assume that an edge $e^* \in \Upsilon$ does not belong to $\bigcup_{k=1}^{N} T_k$. Let $v := \partial^+_{H_0}(e^*)$ and $w := \partial^-_{H_0}(e^*)$. From Lemma 4.1, each of v and w is popped from S in some execution of (S1.1) in **FindPivot2**. Without loss of generality, we can assume that v is popped before w. Then v is not a root (otherwise, v must be the parent of w, which contradicts the assumption). Immediately after v is popped in (S1.1), Lemma 4.2 implies $|\delta_{H_k}(v)| \geq 1$ for some $k \in \{0, 1, \ldots, N\}$. Moreover, $|\delta_{H_k}(v)| = 1$ must hold. In fact, if $|\delta_{H_k}(v)| \geq 2$, then we find from $\rho(v) = 0$ that condition (b) in (S1.2) is satisfied. Then $e^* = (v, w) \in T_{k+1}$ occurs, which contradicts $e^* \notin \bigcup_{k=1}^{N} T_k$. Immediately after w is popped in (S1.1), $|\delta_{H_l}(w)| \geq 1$ holds for some $l \in \{0, 1, \ldots, N\}$ since $e^* \in \delta_{H_l}(w)$. Let us consider the following two cases:

(A) w is a root (i.e., $\rho(w) = 1$).
(B) w is not a root (i.e., $\rho(w) = 0$).

In case (A), w satisfies condition (a) in (S1.2). In case (B), there exist the parent, say p, of w and an edge e' with $e' = (w, p)$ or $e' = (p, w)$. Then we have $|\delta_{H_l}(w)| \geq 2$, since distinct edges e^* and e' belong to $\delta_{H_l}(w)$, hence w satisfies condition (b) in (S1.2). Since both cases satisfy (a) or (b) in (S1.2), execution of (S1.2.1) occurs (i.e., $e^* \in T_{l+1}$), which is a contradiction. Therefore, $\Upsilon \subseteq \bigcup_{k=1}^{N} T_k$ is proved. Finally, it is obvious that $T_k \cap T_l = \emptyset$ holds for all $k, l \in \{1, \ldots, N\}$ $(k \neq l)$, since $T_{k+1} = \delta_{H_k}(y_{k+1}) \subseteq E(H_k) = E(H_{k-1}) \setminus T_k$ for each $k = 1, \ldots, N-1$. □

Theorem 4.1. If $H := G[\Upsilon]$ is an undirected forest, then the number of iterations of the loop (S2)'–(S4) in **ModifiedDijkstr2** does not exceed $n_0/2$.

Proof. Let N be the number defined in Lemma 4.3. Since the number of iterations of the loop does not exceed N, it is sufficient to show $N \leq n_0/2$. We define y_k and T_k $(k = 1, \ldots, N)$ in a similar way to those in the proof of Lemma 4.3. Let $H_1, \ldots, H_p$ be connected components of H, and $n_i := |V(H_i)|$ $(i = 1, \ldots, p)$. Then $|V(H)| = n_0 = \sum_{i=1}^{p} n_i$

and $|E(H_i)| = n_i - 1$ $(i = 1, \ldots, p)$, since each H_i is an undirected tree. Consider H_i for a fixed index $i \in \{1, \ldots, p\}$. Among the vertices in $V(H_i) \cap \{y_1, \ldots, y_N\}$, let y_{p_i} (resp. y_{q_i}) be the vertex popped from S first (resp. last). Also, let $N_i := q_i - p_i + 1$. Then $y_{p_i}, y_{p_i+1}, \ldots, y_{q_i} \in V(H_i)$ obviously holds. Note that y_{q_i} may be a root. Also, it follows from Lemma 4.3 that $E(H_i) = \bigcup_{k=p_i}^{q_i} T_k$ and $T_k \cap T_l = \emptyset$ $(k, l \in \{p_i, \ldots, q_i\},\ k \neq l)$ hold. It is easy to see, from conditions (a) and (b) in (S1.2) in **FindPivot2**, that $|T_k| \geq 2$ $(k = p_i, \ldots, q_i - 1)$ and $|T_{q_i}| \geq 1$. Hence, we have

$$|E(H_i)| = n_i - 1 = \sum_{k=p_i}^{q_i} |T_k| \geq 2N_i - 1,$$

that is, $N_i \leq n_i/2$ holds. Since $N = \sum_{i=1}^{p} N_i$ and $n_0 = \sum_{i=1}^{p} n_i$, we obtain $N \leq n_0/2$. □

Finally, let us show a numerical example of applying **ModifiedDijkstra2** to a conservative network $\mathcal{N} := (G, l)$ in Figure 4 (a) with $s = 1$.

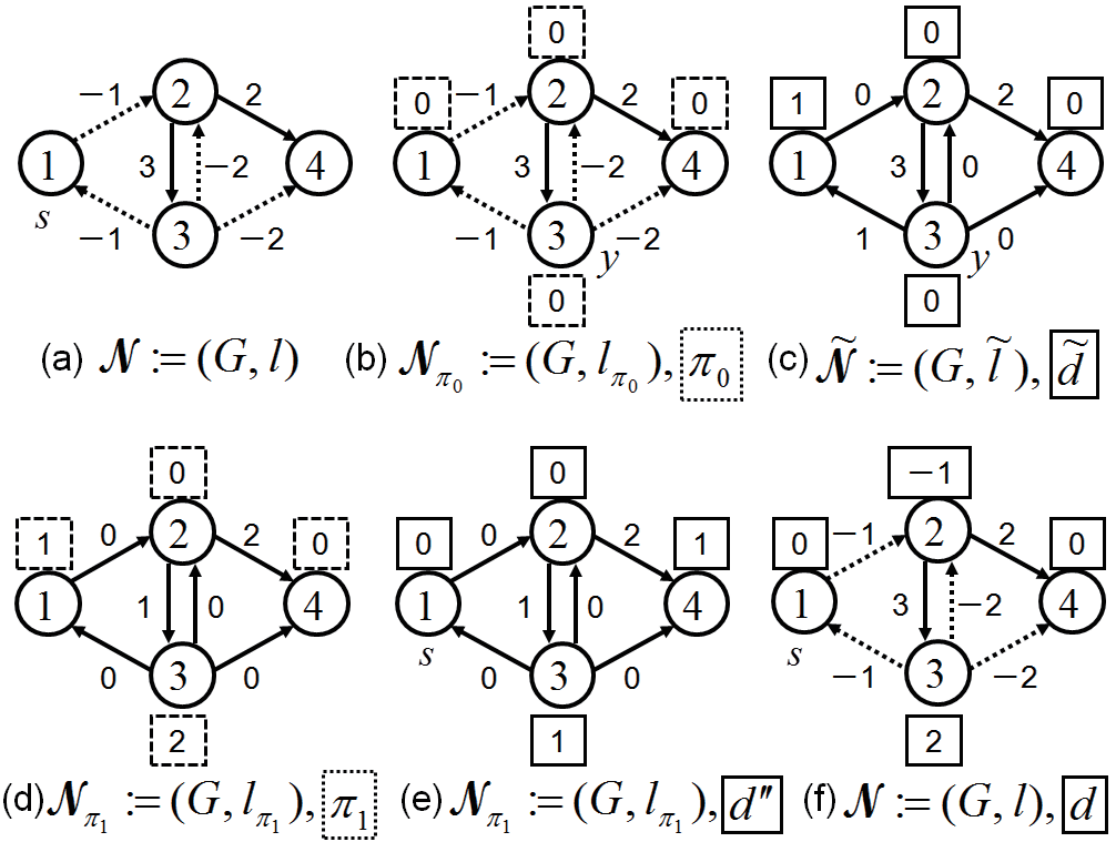

Figure 4: A numerical example (dotted lines denote edges with negative length; small rectangles in (b)–(f) indicate π_0, $\tilde{d}$, π_1, d'', and d, respectively; recall that d'' is a shortest-length function from s)

In this example, since (S3) and (S4) are done only once, let π_0 (resp. π_1) be π immediately before (resp. after) the unique execution of (S4) in **ModifiedDijkstra2**. By (S1)', $\pi_0(v) = 0$ holds for each $v \in V(G) = \{1, 2, 3, 4\}$. Since $\Upsilon = \{(1, 2), (3, 1), (3, 2), (3, 4)\} \neq \emptyset$, H is set as $H = G[\Upsilon]$, and **Ordering** is called in (S1)'. Suppose that, by **Ordering**, we obtain a stack $S = (4, 3, 2, 1)$ and a function ρ with $\rho(1) = 1$ and $\rho(v) = 0$ $(v = 2, 3, 4)$.

FindPivot2 in the first (S2)' returns $H = (\{1,2,3,4\}, \{(1,2)\})$, $S = (2,1)$, and $y = 3$. Immediately before (S3), $\mathcal{N}_{\pi_0} := (G, l_{\pi_0})$ is obtained as is shown in Figure 4 (b). By (S3), $\tilde{\mathcal{N}} := (G, \tilde{l})$ is constructed as is shown in Figure 4 (c), where $e^+ = (3,2)$ (or $(3,4)$), $|l_{\pi_0}(e^+)| = 2$, $\Upsilon^-_{\pi_0,y} = \Upsilon^-_{\pi_0,3} = \emptyset$, and $\overline{\Upsilon_{\pi_0,y}} = \overline{\Upsilon_{\pi_0,3}} = \{(1,2)\}$. $\tilde{l}$ is calculated as, for example, $\tilde{l}((3,1)) = l_{\pi_0}((3,1)) + |l_{\pi_0}(e^+)| = -1 + 2 = 1$, $\tilde{l}((2,4)) = l_{\pi_0}((2,4)) = 2$, and $\tilde{l}((1,2)) = \pi_0(1) - \pi_0(2) = 0$. By **Dijkstra** in (S4), $\tilde{d}$ is obtained as is shown in Figure 4 (c). π is updated as $\pi_1(y) = \pi_1(3) = \pi_0(3) + |l_{\pi_0}(e^+)| = 2$, $\pi_1(1) = \pi_0(1) + \tilde{d}(1) = 1$, $\pi_1(2) = \pi_0(2) + \tilde{d}(2) = 0$, and $\pi_1(4) = \pi_0(4) + \tilde{d}(4) = 0$. Hence, l_{π_1} is calculated as is shown in Figure 4 (d), and we have $\Upsilon_{\pi_1} = \emptyset$. Since **FindPivot2** in the second (S2)' does (S1.2.2) without satisfying the condition "$\Upsilon_\pi \cap T \neq \emptyset$", it returns $y =$ **null** with $H = (\{1,2,3,4\}, \emptyset)$ and $S = \emptyset$. Hence, we proceed to (S5) and obtain, by **Dijkstra**, d'' as is shown in Figure 4 (e). Finally, we obtain a solution d as is shown in Figure 4 (f).

5. Conclusion

We considered the problem of finding the shortest paths from a source s to every vertex $v \in V(G)$ in a conservative network $\mathcal{N} := (G, l)$ with edges of negative length, and we proposed two efficient algorithms for the problem. Both proposed algorithms have the same complexity $O(n_0 S(m,n))$, where n_0 and $S(m,n)$ are as defined above. Further, we pointed out that the revised versions of the algorithms shown in sections 2 and 3 iterate Dijkstra's method at most $n_0/2$ times for a specific class of instances whose subgraphs induced by negative edges are undirected forests by incorporating an additional routine based on the breadth-first search. If we delete an assumption of the conservativeness, then it may be checked by a method shown in [14] in advance. As our future work, we may develop an algorithm containing a routine for finding a negative cycle in $\mathcal{N}$, or accelerate the proposed algorithms by devising an order of vertices y chosen in **FindPivot** or **FindPivot2**.

Acknowledgments

The authors express deep thanks to the two anonymous referees for their constructive suggestions and comments.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin: *Network Flows* (Prentice Hall, New Jersey, 1993).

[2] R.E. Bellman: On a routing problem. *Quarterly of Applied Mathematics*, **16** (1958), 87–90.

[3] N. Deo and C. Pang: Shortest path problems: Taxonomy and annotation. *Networks*, **14** (1984), 275–323.

[4] E.W. Dijkstra: A note on two problems in connexion with graphs. *Numerische Mathematik*, **1** (1959), 269–271.

[5] Y. Dinitz and R. Itzhak: *Hybrid Bellman-Ford-Dijkstra Algorithm* (Technical Report CS-10-04, Department of Computer Science, Ben-Gurion University of the Negev, Israel, 2010).

[6] L.R. Ford: *Network Flow Theory* (Report P-923, RAND Corporation, Santa Monica, CA, 1956).

[7] S. Fujishige: A note on the problem of updating shortest paths. *Networks*, **11** (1981), 317–319.

[8] A.V. Goldberg: A simple shortest path algorithm with linear average time. *Lecture Notes in Computer Science*, **2161** (2001), 230–241.

[9] S. Goto and A. Sangiovanni-Vincentelli: A new shortest path updating algorithm. *Networks*, **8** (1978), 341–372.

[10] J. Hershberger, M. Maxel, and S. Suri: Finding the k shortest simple paths: A new algorithm and its implementation. *ACM Transactions on Algorithms*, **3** (2007), Article No.45.

[11] M. Holzer, F. Schulz, D. Wagner, and T. Willhalm: Combining speed-up techniques for shortest path computations. *Lecture Notes in Computer Science*, **3059** (2004), 269–284.

[12] P.N. Klein, S. Mozes, and O. Weimann: Shortest paths in directed planar graphs with negative lengths: A linear-space $O(n \log_2 n)$-time algorithm. *ACM Transactions on Algorithms*, **6** (2010), Article No.30.

[13] E. Kohler, R.H. Mohring, and H. Schilling: Acceleration of shortest path and constrained shortest path computation. *Lecture Notes in Computer Science*, **3503** (2005), 126–138.

[14] B. Korte and J. Vygen: *Combinatorial Optimization* (Springer-Verlag, New York, 2000).

[15] E.F. Moore: The shortest path through a maze. In *Proceedings of the International Symposium on the Theory of Switching, Part II* (Harvard University Press, 1959), 285–292.

[16] S. Peyer, D. Rautenbach, and J. Vygen: A generalization of Dijkstra's shortest path algorithm with applications to VLSI routing. *Journal of Discrete Algorithms*, **7** (2009), 377–390.

[17] R. Sedgewick and J.S. Vitter: Shortest paths in euclidean graphs. *Algorithmica*, **1** (1986), 31–48.

[18] D.D. Sleator and R.E. Tarjan: A data structure for dynamic trees. *Journal of Computer and System Sciences*, **26** (1983), 362–391.

[19] A.J.V. Skriver and K.A. Andersen: A label correcting approach for solving bicriterion shortest path problems. *Computers & Operations Research*, **27** (2000), 507–524.

[20] M. Thorup: Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM*, **46** (1999), 362–394.

[21] D. Wagner and T. Willhalm: Geometric speed-up techniques for finding shortest paths in large sparse graphs. *Lecture Notes in Computer Science*, **2832** (2003), 776–787.

Akira Nakayama
Graduate School of Symbiotic Systems Science
Fukushima University
1 Kanayagawa, Fukushima 960-1296, Japan
E-mail: nakayama@sss.fukushima-u.ac.jp