# Partially Dynamic Single-Source Shortest Paths on Digraphs with Positive Weights

Wei Ding[1] and Guohui Lin[2,⋆]

[1] Zhejiang University of Water Resources and Electric Power
Hangzhou, Zhejiang 310018, China
dingweicumt@163.com
[2] Department of Computing Science, University of Alberta
Edmonton, Alberta T6G 2E8, Canada
guohui@ualberta.ca

**Abstract.** We examine several structural properties of single-source shortest paths and present a local search algorithm for the partially dynamic single-source shortest paths problem. Our algorithm works on both deterministic digraphs and undirected graphs. For a deterministic digraph with positive arc weights, our algorithm handles a single arc weight increase in $O(n + \frac{n^2 \log n}{m})$ expected time, where $n$ is the number of nodes and $m$ is the number of edges in the digraph. Specifically, our algorithm is an $O(n)$ expected time algorithm when $m = \Omega(n \log n)$. This solves partially an open problem proposed by Demetrescu and Italiano (Journal of the ACM. 51(2004), 968–992).

**Keywords:** partially dynamic, single-source shortest paths, local search, expected time.

## 1 Introduction

An *all-pairs shortest paths* (APSP) algorithm computes the shortest paths between every pair of nodes in a given digraph, and a *single-source shortest paths* (SSSP) algorithm computes the shortest paths from a given source node to all the other nodes. When dynamic changes occur to the digraph, a *dynamic* APSP (SSSP, respectively) algorithm updates the shortest paths. One can recompute the shortest paths using the *static* algorithms, but a truly dynamic algorithm seeks for updating operations using fundamental properties of the shortest paths, and is expected to run faster than recomputation by the static algorithms. We present a *partially* dynamic SSSP algorithm for digraphs with arbitrary positive arc weights. We use arc and edge interchangeably in this paper.

Dynamic changes of a digraph (a.k.a. *edge update*) include *topology update* and *edge weight update*. Topology update includes *edge insertions* and *deletions*, and edge weight update includes weight *increase* and *decrease*. Note that topology update can be realized by edge weight update, and vice versa. When dealing with

⋆ Correspondence author.

Q. Gu, P. Hell, and B. Yang (Eds.): AAIM 2014, LNCS 8546, pp. 197–207, 2014.

only edge weight updates, an algorithm is said to be *fully dynamic* if it can handle both edge weight increases and decreases, but *partially dynamic* if it only can handle edge weight decreases or increases but not both [1,4,5,10,13,16,17]. When dealing with only topology updates, an algorithm is *fully dynamic* if it can handle both edge insertions and deletions, is *incremental* if it can handle only edge insertions but not deletions, and is *decremental* if it can handle only deletions but not insertions [2,3,7,8,11,14,15]. Incremental and decremental algorithms are sometimes collectively called *partially dynamic*.

Demetrescu and Italiano [4,5] studied a generalization of dynamic changes, in which the weights of all the edges incident at a given node are changed in one update. Such an update is called a *node update*, see also [16,17]. Clearly, dynamic APSP and SSSP algorithms for node updates work for edge updates too. Throughout this paper, $n$ and $m$ denote the numbers of nodes and edges (arcs) in the input digraphs, respectively.

The dynamic APSP problem with edge weight updates has been studied extensively since it was proposed [12]. Ausiello *et al.* [1] devised a partially dynamic APSP algorithm with an $O(Cn \log n)$ amortized update time for digraphs of which all edge weights are less than or equal to a constant $C$. King [11] studied the fully dynamic APSP problem on the same class of digraphs and presented an algorithm with $O(n^{\frac{5}{2}}\sqrt{C \log n})$ worst-case update time. Demetrescu and Italiano [4,5] considered node updates on digraphs with non-negative real-valued edge weights, and designed a fully dynamic APSP algorithm with an $O(n^2 \log^3 n)$ amortized time. Thorup [16,17] presented an improved algorithm with a worst-case $O(n^{2.75}\text{polylog}(n))$ time. The fully dynamic APSP problem on *random* graphs was considered by Friedrich and Hebbinghaus [10], who gave an $O(n^{\frac{4}{3}+\epsilon})$ expected time per update algorithm, for any $\epsilon > 0$, for random graphs $G(n, p)$ with uniform random edge weights, and by Peres *et al.* [13], who presented an $O(\log^2 n)$ expected time per update algorithm for complete digraphs with edge weights selected independently at random from the uniform distribution on interval $[0, 1]$. There are also a number of *approximate* incremental and decremental APSP algorithms [2,3,14] for undirected graphs with positive edge weights.

For the dynamic SSSP problem with topology updates, Even and Shiloach [8] and Dinitz [7] presented $O(n)$ amortized time decremental algorithms for unweighted and undirected graphs. King [11] presented a decremental algorithm to maintain shortest paths of distance up to $d$ in $O(md)$ time, for digraphs with positive integer edge weights. Bernstein and Roditty [3] devised an $O(\frac{n^2}{m})$ amortized time decremental algorithm to maintain $(1+\epsilon)$-approximate SSSP on unweighted and undirected graphs. This is the first algorithm that breaks the long-standing $O(n)$ update time barrier on decremental SSSP problem, on not-too-sparse graphs. On the other hand, Roditty and Zwick [15] showed that the incremental and decremental SSSP problems on edge weighted digraphs are at least as hard as the static APSP problem; by similar reductions they showed that the incremental and decremental SSSP problems on edge unweighted digraphs are at least as hard as the Boolean matrix multiplication problem. These hardness

results hint that it will be difficult to improve the known best algorithms of Even and Shiloach [8].

For the dynamic SSSP problem with edge weight updates, Fakcharoemphol and Rao [9] studied the fully dynamic variant on planar digraphs, and devised an $O(n^{\frac{4}{5}} \log^{\frac{13}{5}} n)$ amortized time algorithm. Demetrescu and Italiano [4] raised the open problem on whether or not we can solve efficiently (*i.e.*, better than recomputation) fully dynamic SSSP problem on general graphs. We address partially this open problem by presenting a partially dynamic SSSP algorithm for handling arc weight increase on digraphs with positive weights. For deterministic digraphs, our algorithm can handle a single arc weight increase in $O(n + \frac{n^2 \log n}{m})$ expected time. When $m = \Omega(n \log n)$, our algorithm is an $O(n)$ expected time algorithm. Moreover, our algorithm can also work on undirected graphs with positive weights.

The rest of the paper is organized as follows. In Sect. 2, we define some notations frequently used in this paper and the partially dynamic SSSP problem formally. In Sect. 3, we show several fundamental properties. In Sect. 4, we present a local search algorithm based on these properties. In Sect. 5, we analyze the worst-case expected update time of our algorithm handling a single arc weight increase for deterministic digraphs. In Sect. 6, we conclude the paper with some future work.

## 2 Problem Statements and Notations

Let $D = (V, A, w, s)$ be a weighted digraph, where $V$ is the node set, $A$ is the arc set, $s$ is a designated node called *source*, and $w(\cdot)$ is a weight function $w : A \to \mathbb{R}^+$. Suppose that the weight of an arc $a$ increases from $w(a)$ up to $w'(a)$ and all the other arc weights stay unchanged. Let $\delta = w'(a) - w(a)$. Note that $w'(a)$ always remains positive. The resultant digraph is denoted as $D' = (V, A, w', s)$. We use $d^*_D(s, v)$ (*resp.* $d^*_{D'}(s, v)$) to denote the shortest path distance from $s$ to node $v$ in $D$ (*resp.* $D'$), and use $T_s$ (*resp.* $T'_s$) to denote the *single-source shortest paths tree* in $D$ (*resp.* $D'$).

*Problem 1.* Given $D = (V, A, w, s)$ and $T_s$ in $D$, we replace $w(a)$ with $w'(a)$ for one arc $a$ to obtain a new digraph $D' = (V, A, w', s)$. The problem of updating $T_s$ to $T'_s$ in $D'$ is called the *partially dynamic SSSP problem with a single arc weight increase.*

The essence of Problem 1 is to maintain SSSP, that is, to update the given $T_s$ in $D$ to $T'_s$ in $D'$. Obviously, $T_s$ and $T'_s$ are both out-trees from $s$, and both can be taken as rooted trees at $s$. For any node of $V$, let $T_u$ (*resp.* $T'_u$) denote the subtree of $T_s$ (*resp.* $T'_s$) rooted at $u$. We use $\pi_T(s, v)$ to denote the $s$-to-$v$ simple path along $T_s$, and use $d_T(s, v)$ to denote the length of $\pi_T(s, v)$ which is equal to the sum of the weights on all the arcs of $\pi_T(s, v)$. Clearly, $d_T(s, v) = d^*_D(s, v)$ and $d_{T'}(s, v) = d^*_{D'}(s, v)$ for every $v \in V$. In addition, we use $V(\cdot)$ and $A(\cdot)$ to denote the node set and arc set of one digraph or its subgraph respectively. Let $S(D, U)$ denote the subgraph of $D$ induced by the subset $U \subseteq V$ of nodes.

For any subset $U \subset V$, we let $C[U, V \setminus U]$ denote the subset of arcs with tail in $U$ and head in $V \setminus U$.

Let $a = (u, v)$ denote the arc of $D$ from $u$ to $v$ and $w(u, v)$ (or sometimes $w(a)$) denote its weight. We call $u$ the *tail* of $a$ and denote $u$ as $t_a$, and call $v$ the *head* of $a$ and denote $v$ as $h_a$. Thus, $a = (t_a, h_a)$, which is called an *outgoing arc* from $t_a$ and an *incoming arc* to $h_a$. The set of all the outgoing arcs from a node $u$ is denoted as $\mathcal{O}(u)$ and the set of all the incoming arcs to $v$ is denoted as $\mathcal{I}(v)$.

## 3 Fundamental Properties

In the section, we show several fundamental properties which will play an important role in the design of our *local search algorithm*, described as PSAI, for updating SSSP under a single arc weight increase.

**Theorem 1.** *For any $a \notin T_s$, no matter how much $w(a)$ increases by to $w'(a)$, it always holds that $d^*_{D'}(s, v) = d_T(s, v)$ for any $v \in V$.*

*Proof.* No matter how much $w(a)$ increases by to $w'(a)$, the length of every simple $s$-to-$v_1$ path in $D'$ passing through $a$ is larger than its length in $D$, and the length of every simple $s$-to-$v_2$ path in $D'$ not passing through $a$ is the same as its length in $D$. So the length of every simple path in $D'$ is either larger than or equal to its length in $D$. When $a \notin T_s$, $\pi_T(s, v)$ obviously does not pass through $a$, and thus the length of $\pi_T(s, v)$ stays unchanged. Considering that $d_T(s, v) = d_D(s, v)$, we conclude that $\pi_T(s, v)$ is always the $s$-to-$v$ shortest path in $D'$ for any $v \in V$. So, $d^*_{D'}(s, v) = d_T(s, v), \forall v \in V$. The proof is complete. □

**Theorem 2.** *For any $a \in T_s$, it always holds that $d^*_{D'}(s, u) = d_T(s, u)$ for any $u \in V \setminus V(T_{h_a})$ regardless of how much $w(a)$ increases by to $w'(a)$.*

*Proof.* When $a \in T_s$, we observe that $\pi_T(s, u)$ does not pass through $a$ for any $u \in V \setminus V(T_{h_a})$. Thus, the length of $\pi_T(s, u)$ stays unchanged and thus $\pi_T(s, u)$ is always the $s$-to-$u$ shortest path in $D'$ regardless of how much $w(a)$ increases by to $w'(a)$. This implies that $d^*_{D'}(s, u) = d_T(s, u), \forall u \in V \setminus V(T_{h_a})$. The proof is complete. □

When $a \in T_s$ and its weight increases to $w'(a)$, one observes that since $\pi_T(s, v)$ passes through $a$ for any $v \in V(T_{h_a})$, the length of $\pi_T(s, v)$ in $D'$ is equal to its length in $D$ plus $w'(a) - w(a)$. So, the length of $\pi_T(s, v)$ in $D'$ is larger than $d_T(s, v)$. Since the length of every $s$-to-$v$ simple path not passing through $a$ stays unchanged, it may occur that some $s$-to-$v$ simple paths in $D'$ not passing through $a$ have a smaller length than $\pi_T(s, v)$. We conclude that every $s$-to-$v$ simple path in $D'$ having a smaller length than $\pi_T(s, v)$ is surely composed of one $s$-to-$u$ simple path in $D'$, followed by arc $(u, v')$ and the path $v'$-to-$v$ in $T_{h_a}$, where $u \in \mathcal{I}(v') \setminus V(T_{h_a})$. Such an $s$-to-$v'$ path can minimize its length by selecting $\pi_T(s, u)$ as the $s$-to-$u$ simple path. Let

$$b(v) = \arg \min_{u \in \mathcal{I}(v) \setminus V(T_{h_a})} \{d_T(s, u) + w(u, v)\}, \quad \forall v \in V(T_{h_a}), \tag{1}$$

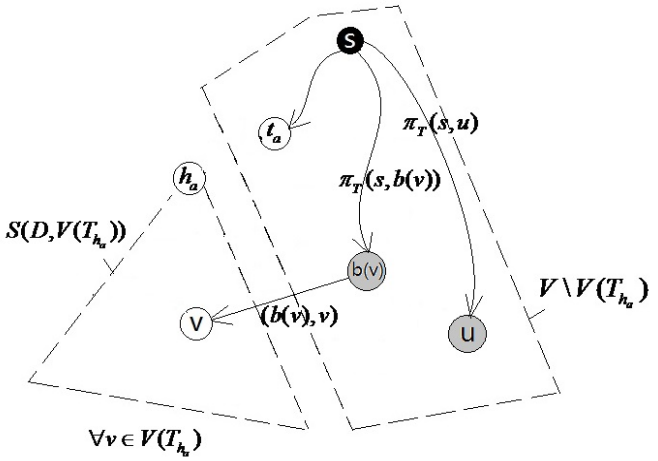


**Fig. 1.** Illustration of $D_a$

and $b(v)$ is called a *bridge node* of $v$. Let $b(v) = \text{null}$ if $b(v)$ does not exist in $D'$ and thus $(b(v), v) = \text{null}$. In addition, we use an arc $\epsilon(\pi_T(s, v))$ to represent $\pi_T(s, v)$.

Accordingly, we can construct an auxiliary graph $D_a = (V, A_a, w_a(\cdot))$ from $D$ based on $a$ in the following way, see Fig. 1. The new arc set $A_a$ is composed of all the arcs in the subgraph of $D$ induced by $V(T_{h_a})$ and all $\epsilon(\pi_T(s, u)), u \in V \setminus V(T_{h_a})$ and all $(b(v), v), v \in V(T_{h_a})$. For any arc $r \in A_a$, the weight of $r$, $w_a(r)$, is equal to the length of $\pi_T(s, u_0)$ if $r$ represents $\pi_T(s, u_0)$, or otherwise $w_a(r) = w(r)$. That is,

$$\begin{aligned} A_a = {} & A(S(D, V(T_{h_a}))) \cup \{(b(v), v) : v \in V(T_{h_a})\} \\ & \cup \{\epsilon(\pi_T(s, u)) : u \in V \setminus V(T_{h_a})\}, \end{aligned} \tag{2}$$

and

$$w_a(r) = \begin{cases} d_T(s, u_0) & \text{if} \quad r = \epsilon(\pi_T(s, u_0)), \\ w(r) & \text{otherwise}, \end{cases} \qquad \forall r \in A_a. \tag{3}$$

**Lemma 1.** *For any $a \in T_s$ and any $v \in V(T_{h_a})$, it always holds that any $s$-to-$v$ shortest path in $D'$ contains exactly one arc in $C[V \setminus V(T_{h_a}), V(T_{h_a})]$.*

*Proof.* We conclude from $s \in V \setminus V(T_{h_a})$ and $v \in V(T_{h_a})$ that any $s$-to-$v$ simple path in $D'$ contains at least one arc in $C[V \setminus V(T_{h_a}), V(T_{h_a})]$. Suppose that $\pi^{\triangle}(s, v)$ is an $s$-to-$v$ shortest path in $D'$ containing two arcs in $C[V \setminus V(T_{h_a}), V(T_{h_a})]$. In details, $\pi^{\triangle}(s, v)$ consists of $\pi_T(s, u_1)$, two arcs $(u_1, v_1)$ and $(u_3, v_3)$ in $C[V \setminus V(T_{h_a}), V(T_{h_a})]$ and $(v_2, u_2)$, two disjoint paths $\pi(v_1, v_2)$ and $\pi(v_3, v)$ in $S(D', V(T_{h_a}))$, and one path $\pi(u_2, u_3)$ in $S(D', V \setminus V(T_{h_a}))$, where $u_1, u_2, u_3 \in V \setminus V(T_{h_a})$ and $v_1, v_2 \in V(T_{h_a})$ (see Fig. 2). Note that it is

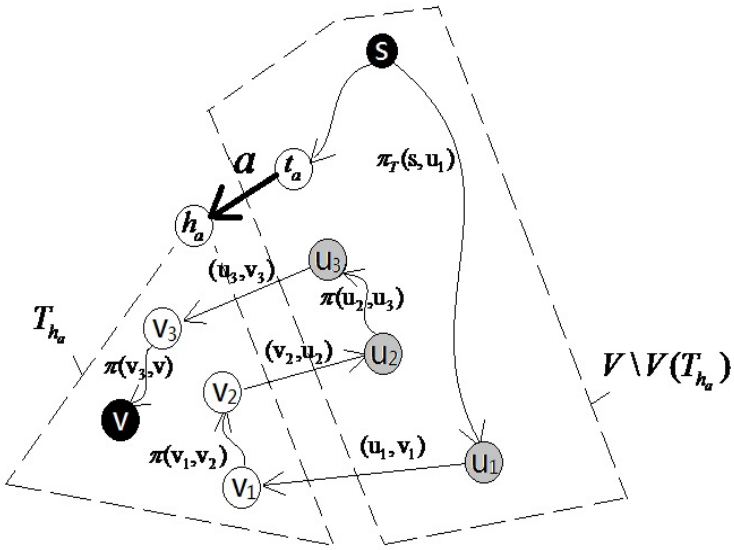


**Fig. 2.** Illustrate the proof of Lemma 1

possible that $v_1 = v_2, u_2 = u_3$ and $v_3 = v$. Clearly, the seven parts of $\pi^{\triangle}(s,v)$ are disjoint. The sub-path $\pi^{\triangle}(s,u_3)$ of $\pi^{\triangle}(s,v)$ is an $s$-to-$u_3$ simple path in $D'$. By Theorem 2, $\pi_T(s,u_3)$ is still an $s$-to-$u_3$ shortest path in $D'$. So, the length of $\pi_T(s,u_3)$ is less than the length of $\pi^{\triangle}(s,u_3)$. Therefore, a new path $\pi^{\star}(s,v)$ composed of $\pi_T(s,u_3)$, $(u_3,v_3)$ and $\pi(v_3,v)$ has a smaller length than $\pi^{\triangle}(s,v)$. This causes a contradiction. □

**Theorem 3.** *For any $a \in T_s$, it always holds that $d^*_{D'}(s,v) = d^*_{D_a}(s,v)$ for any $v \in V$ regardless of how much $w(a)$ increases by to $w'(a)$.*

*Proof.* When $a \in T_s$, Theorem 2 shows that $\pi_T(s,u)$ is always an $s$-to-$u$ shortest path in $D'$ for any $u \in V \setminus V(T_{h_a})$ and also an $s$-to-$u$ shortest path in $D_a$. So, $d^*_{D'}(s,u) = d^*_{D_a}(s,u)$. The work left is to prove $d^*_{D'}(s,v) = d^*_{D_a}(s,v)$ for any $v \in V(T_{h_a})$.

We observe that the weight of $a$ increases from $w(a)$ up to $w'(a)$ and thus the length of $\pi_T(s,v)$ in $D'$ increases by $w'(a) - w(a)$. So, it is certain that an $s$-to-$v$ shortest path in $D'$ is one of such simple paths as composed of three disjoint parts $\pi_T(s,b(v_0)), (b(v_0),v_0)$ and a $v_0$-to-$v$ path $\pi(v_0,v)$ where $v_0 \in V(T_{h_a})$. We conclude from Lemma 1 and $(b(v_0),v_0) \in C[V \setminus V(T_{h_a}), V(T_{h_a})]$ that $\pi(v_0,v)$ contains no arc in $C[V \setminus V(T_{h_a}), V(T_{h_a})]$. So, $\pi(v_0,v)$ is a $v_0$-to-$v$ shortest path in $S(D, V(T_{h_a}))$.

We need to visit all the incoming arcs to $v$ in order to find $b(v)$ for all $v \in V(T_{h_a})$ and need to traverse $S(D, V(T_{h_a}))$ to compute a $v_0$-to-$v$ shortest path in $S(D, V(T_{h_a}))$. Therefore, the problem of finding an $s$-to-$v$ shortest path in $D'$ is

equivalent to the problem of finding an $s$-to-$v$ shortest path in $D_a$. This implies that $d^*_{D'}(s,v) = d^*_{D_a}(s,v)$. The proof is complete. □

## 4 Local Search Algorithm

From the properties shown in Sect. 3, we need to discuss the situation of the arc $a$ with its weight increased, i.e., discuss whether $a \notin T_s$ or $a \in T_s$. When $a \notin T_s$, we conclude from Theorem 1 that $\pi_T(s,v)$ is also an $s$-to-$v$ shortest path in $D'$ for any $v \in V$. Therefore, $T_s$ is also a single-source shortest paths tree in $D'$ with $s$ as the origin. So we need no work. When $a \in T_s$, we conclude from Theorem 2 that $\pi_T(s,u)$ is also an $s$-to-$u$ shortest path in $D'$ for any $u \in V \setminus V(T_{h_a})$, and from Theorem 3 that $\pi(s,v)$ is an $s$-to-$v$ shortest path in $D'$ iff it is an $s$-to-$v$ shortest path in $D_a$ for any $v \in V(T_{h_a})$. So, we only need to update the $s$-to-$v$ shortest path for all $v \in V(T_{h_a})$. Above discussions can be described as a local search algorithm PSAI.

For every $v \in V(T_{h_a})$, we conclude from Eq. (1) that we need to visit all the nodes in $\mathcal{I}(v)$ and judge whether one node is in $T_{h_a}$ or not. We define a 0-1 variable $p(v)$. In details, $p(v) = 0$ means that $v$ is not in $T_{h_a}$ and $p(v) = 1$ means that $v$ is in $T_{h_a}$. Initially, we set $p(v) = 0$ for all $v \in V$. In order to make PSAI facilitate implementing its local search procedure, we use appropriate data structures to store the input digraph and single-source shortest paths trees.

Algorithm PSAI:
**Input:** $D = (V, A, w, s)$, $T_s$ and $w'(a)$;
**Output:** $T'_s$ in $D'$.
**Step_0**: If $a \notin T_s$, then return $T_s$; if $a \in T_s$, then goto Step_1;
**Step_1**: Use DFS to traverse $T_{h_a}$ twice from $h_a$; in the first one, we let $p(v) \leftarrow 1, \forall v \in V(T_{h_a})$; in the second one, for every $v \in V(T_{h_a})$, we visit all nodes in $\mathcal{I}(v)$ and find $b(v)$ using Eq. (1);
**Step_2**: Construct $D_a$ based on Eqs. (2) and (3);
**Step_3**: Use Dijkstra's algorithm in $D_a$ to compute the single-source shortest paths tree with $s$ as the origin, and record it as $T'_s$;
**Step_4**: Use DFS to traverse $T_{h_a}$ and reset $p(v) \leftarrow 0, \forall v \in V(T_{h_a})$;

**Theorem 4.** *PSAI takes $O(m + n \log n)$ time in the worst case.*

*Proof.* Let $|V(T_{h_a})| = n', |A(S(D, V(T_{h_a})))| = m'$ and $\sum_{v \in V(T_{h_a})} |\mathcal{I}(v)| = K$. Step_1 first takes $O(n')$ time to use DFS in $T_{h_a}$ to do preliminaries, and then uses DFS in $T_{h_a}$ the second time to find $b(v), \forall v \in V(T_{h_a})$ which takes $O(K)$ time. Obviously, $n' \leq K$. So, Step_1 runs $O(K)$ time. Since $T_{h_a}$ has at most $n'$ bridge nodes, $D_a$ has $n$ nodes and at most $m' + n$ arcs. Clearly, $m' \leq K$. So, Step_2 spends $O(K)$ time to construct $D_a$. Step_3 uses Dijkstra's algorithm [6] in $D_a$ to compute $T'_s$, whose running time is $O(|A(D_a)| + |V| \log |V|)$ and $O(K + n \log n)$. Step_4 takes $O(n')$ time to use DFS in $T_{h_a}$ to reset all the values of $p(v), v \in V(T_{h_a})$. Since $K \leq m$, PSAI takes $O(m + n \log n)$ time in the worst case. □

## 5 Average Case Analysis

Let $\{\cdot\}$ denote an *event*, e.g., $\{h_a = u\}$ represents the event of $a$ having a head $u$, and $\{a \in T_s\}$ (*resp.* $\{a \notin T_s\}$) represents the event that $T_s$ contains $a$ (*resp.* $T_s$ does not contain $a$). Let $\omega = \langle \omega_1, \omega_2 \rangle$ be an unchangeable couple, where $\omega_1 \in \Omega_1, \omega_2 \in \Omega_2$ and $\Omega_1, \Omega_2$ are two *event spaces* as follows

$$\Omega_1 = \{\{h_a = u\} : u \in V\}, \quad \Omega_2 = \{\{a \in T_s\},\ \{a \notin T_s\}\}. \tag{4}$$

Suppose that $\{\omega_1 \in \Omega_1\}$ and $\{\omega_2 \in \Omega_2\}$ are *independent*. Let

$$\Omega = \Omega_1 \times \Omega_2 = \{\langle \omega_1, \omega_2 \rangle :\ \omega_1 \in \Omega_1, \omega_2 \in \Omega_2\}. \tag{5}$$

Given any $D = (V, A, w, s)$, we denote by $G$ the topology of $D$ and by $T_s$ the SSSP tree of $D$ with $s$ as the origin. In fact, a given $D$ means that both $G$ and $T_s$ of $D$ are given. When $D$ is given, every $\omega \in \Omega$ represents a situation of a single arc weight increase, and thus acts as an elementary *conditional event* of time costs of PSAI. So, $\Omega$ is just the *conditional event space* induced by $D$.

Let $\mathbb{E}[Z|X = x, Y = y]$ denote the *conditional expectation* of $Z$ when $X = x$ and $Y = y$. Lemma 2 shows an important formula on conditional expectation.

**Lemma 2.** *Given two discrete random variables $X$, $Y$ and another random variable $Z$, provided that $\{X = x\}$ and $\{Y = y\}$ are the condition events of $Z$, we have*

$$\mathbb{E}[Z] = \sum_{x,y} \mathbb{E}[Z|X = x, Y = y] \cdot \Pr[X = x, Y = y]. \tag{6}$$

Let $\mathbb{E}[time]$ denote the expected time of PSAI dealing with a single arc weight increase in $D$, $\mathbb{E}[time|h_a = u, a \in T_s]$ denote the expected time of PSAI dealing with the increase of $a$ with $h_a = u$ and $a \in T_s$, and $\mathbb{E}[time|h_a = u, a \notin T_s]$ denote the expected time of PSAI dealing with the increase of $a$ with $h_a = u$ and $a \notin T_s$. Suppose that a single arc weight increase of $D$ occurs at random from the *uniform distribution* on $A$, i.e.,

$$\Pr[\text{an increase occurs to } a] = \frac{1}{m}, \quad \forall a \in A. \tag{7}$$

**Theorem 5.** *Given any $D = (V, A, w, s)$, the expected update time of PSAI dealing with a single arc weight increase is*

$$O\Big(\frac{1}{m} \sum_{u \in V} \sum_{v \in V(T_u)} |\mathcal{I}(v)| + \frac{1}{m} \sum_{u \in V} |V| \log |V| + \frac{1}{m} \sum_{u \in V} |\mathcal{I}(u)|\Big). \tag{8}$$

*Proof.* For any $u \in V$, $D$ has a single incoming arc to $u$ which is in $T_s$, and $|\mathcal{I}(u)| - 1$ incoming arcs to $u$ which are not in $T_s$ when $1 \le |\mathcal{I}(u)| \le n - 1$. According to Eq. (7), we get

$$\Pr[h_a = u, a \in T_s] = \frac{1}{m}, \tag{9}$$

and

$$\Pr[h_a = u, a \notin T_s] = \frac{|\mathcal{I}(u)| - 1}{m}. \tag{10}$$

Theorem 1 implies that PSAI needs no work when $h_a = u$ and $a \notin T_s$, and thus takes only $O(1)$ time in this case. So,

$$\mathbb{E}[time | h_a = u, a \notin T_s] = O(1). \tag{11}$$

From the description of PSAI and the proof of Theorem 4, we conclude the update time of PSAI for any $v \in V(T_u)$ when $h_a = u$ and $a \in T_s$ is

$$\mathbb{E}[time | h_a = u, a \in T_s] = O(\sum_{v \in V(T_u)} |\mathcal{I}(v)| + |V| \log |V|), \tag{12}$$

According to Eq. (5), we can further rewrite $\Omega$ to be

$$\Omega = \bigcup_{u \in V} \{h_a = u, a \in T_s\} \cup \{h_a = u, a \notin T_s\}. \tag{13}$$

and then derive from Lemma 2 that

$$\begin{aligned}
\mathbb{E}[time] &= \sum_{\omega \in \Omega} \mathbb{E}[time|\omega] \cdot \Pr[\omega] \\
&= \sum_{u \in V} \Big( \mathbb{E}[time | h_a = u, a \in T_s] \cdot \Pr[h_a = u, a \in T_s] \Big) + \\
&\quad \sum_{u \in V} \Big( \mathbb{E}[time | h_a = u, a \notin T_s] \cdot \Pr[h_a = u, a \notin T_s] \Big).
\end{aligned}$$

We take Eqs. (9), (10), (11) and (12) into above equality to obtain

$$\begin{aligned}
\mathbb{E}[time] &= \sum_{u \in V} \Big( O(\sum_{v \in V(T_u)} |\mathcal{I}(v)| + |V| \log |V|) \cdot \frac{1}{m} + O(1) \cdot \frac{|\mathcal{I}(u)| - 1}{m} \Big) \\
&= O\Big( \frac{1}{m} \sum_{u \in V} \sum_{v \in V(T_u)} |\mathcal{I}(v)| + \frac{1}{m} \sum_{u \in V} |V| \log |V| + \frac{1}{m} \sum_{u \in V} |\mathcal{I}(u)| \Big). \qquad \square
\end{aligned}$$

**Theorem 6.** *Given any $D = (V, A, w, s)$, the worst-case expected update time of PSAI dealing with a single arc weight increase is $O(n + \frac{n^2 \log n}{m})$.*

*Proof.* Obviously, we have

$$\sum_{u \in V} |\mathcal{I}(u)| = m \quad \text{and} \quad \frac{1}{m} \sum_{u \in V} |V| \log |V| = \frac{n^2 \log n}{m}.$$

Combining with

$$\frac{1}{m} \sum_{u \in V} \sum_{v \in V(T_u)} |\mathcal{I}(v)| = \sum_{u \in V} \Big( \frac{1}{m} \sum_{v \in V(T_u)} |\mathcal{I}(v)| \Big) \leq \sum_{u \in V} O(1) = O(n),$$

we conclude that algorithm PSAI runs in $O(n + \frac{n^2 \log n}{m})$ expected time in the worst case. $\square$

## 6 Concluding Remarks

We have presented a local search algorithm PSAI for handling a single arc weight increase to maintain SSSP on digraphs. The worst-case update time of PSAI is $O(m+n\log n)$, and the worst-case expected update time of PSAI is $O(n+\frac{n^2\log n}{m})$. When $m = \Omega(n\log n)$, PSAI has an $O(n)$ expected updated time. To the best of our knowledge, we are the first one to propose almost linear time algorithm for maintaining SSSP. Also, PSAI applies to undirected graphs with positive weights.

When a single arc weight reduces, whether a linear time algorithm exists for maintaining SSSP on digraphs remains open.

**Acknowledgement.** This research was supported in part by NSERC.

## References

1. Ausiello, G., Italiano, G.F., Marchetti-Spaccamela, A., Nanni, U.: Incremental algorithms for minimal length paths. Journal of Algorithms 12, 615–638 (1991)
2. Bernstein, A.: Fully dynamic $(2+\epsilon)$ approximate all-pairs shortest paths with fast query and close to linear update time. In: Proc. of 50th FOCS, pp. 693–702 (2009)
3. Bernstein, A., Roditty, L.: Improved dynamic algorithms for maintaining approximate shortest paths under deletions. In: Proc. of 22th SODA, pp. 1355–1365 (2011)
4. Demetrescu, C., Italiano, G.F.: A new approach to dynamic all pairs shortest paths. Journal of the ACM 51, 968–992 (2004)
5. Demetrescu, C., Italiano, G.F.: Experimental analysis of dynamic all pairs shortest path algorithms. ACM Transactions on Algorithms 2, 578–601 (2006)
6. Dijkstra, E.W.: A note on two problems in connection with graphs. Numerische Mathematik 1, 269–271 (1959)
7. Dinitz, Y.: Dinitz' algorithm: The original version and Even's version. In: Essays in Memory of Shimon Even, pp. 218–240 (2006)
8. Even, S., Shiloach, Y.: An on-line edge-deletion problem. Journal of the ACM 28, 1–4 (1981)
9. Fakcharoemphol, J., Rao, S.: Planar graphs, negative weight edges, shortest paths, and near linear time. In: Proc. of 42nd FOCS, pp. 232–241 (2001)
10. Friedrich, T., Hebbinghaus, N.: Average update times for fully-dynamic all-pairs shortest paths. In: Hong, S.-H., Nagamochi, H., Fukunaga, T. (eds.) ISAAC 2008. LNCS, vol. 5369, pp. 692–703. Springer, Heidelberg (2008)
11. King, V.: Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs. In: Proc. of 40th FOCS, pp. 81–99 (1999)
12. Murchland, J.: The effect of increasing or decreasing the length of a single arc on all shortest distances in a graph. Technical report, LBS-TNT-26. London Business School, Transport Network Theory Unit, London, UK (1967)
13. Peres, Y., Sotnikov, D., Sudakov, B., Zwick, U.: All-pairs shortest paths in $O(n^2)$ time with high probability. In: Proc. of 51th FOCS, pp. 663–672 (2010)
14. Roditty, L., Zwick, U.: Dynamic approximate all-pairs shortest paths in undirected graphs. In: Proc. of 45th FOCS, pp. 499–508 (2004)

15. Roditty, L., Zwick, U.: On dynamic shortest paths problems. In: Albers, S., Radzik, T. (eds.) ESA 2004. LNCS, vol. 3221, pp. 580–591. Springer, Heidelberg (2004)
16. Thorup, M.: Fully-dynamic all-pairs shortest paths: Faster and allowing negative cycles. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111, pp. 384–396. Springer, Heidelberg (2004)
17. Thorup, M.: Worst-case update times for fully-dynamic all-pairs shortest paths. In: Proc. of 37th STOC, pp. 112–119 (2005)