Improved Algorithms for Replacement Paths Problems in Restricted Graphs*

Amit M. Bhosle†
YASU Technologies Pvt. Ltd.
1-10-98/16, Raj Plaza, Dwarkadas Colony
Begumpet, Hyderabad - 500 016, India.
Email: bhosle@cs.ucsb.edu

Abstract

We present near optimal algorithms for two problems related to finding the replacement paths for edges with respect to shortest paths in sparse graphs. The problems essentially study how the shortest paths change as edges on the path fail, one at a time. Our technique improves the existing bounds for these problems on directed acyclic graphs, planar graphs, and non-planar integer-edge-weighted graphs.

KEYWORDS: *Replacement Paths, Most Vital Arcs, Single Link Failure Recovery, Minimum Spanning Trees Sensitivity.*

1 Introduction

The *Single Source Shortest Paths* ($SSSP$) and the *Minimum Spanning Tree* (MST) problems are classical graph theoretic problems and have been widely studied for decades. Efficient algorithms for computing these trees are long known. For graphs with arbitrarily weighted edges (henceforth called *arbitrary graphs*), the fastest $SSSP$ algorithm is Dijkstra's shortest path algorithm, which runs in $O(m + n \log n)$ time using Fredman and Tarjan's Fibonacci heaps[8]. The same bound holds for constructing the MST of such graphs using the F-Heap implementation of the algorithms by Dijkstra[5] and Prim[20]. The MST algorithm by Chazelle[4] runs in $O(m \cdot \alpha(m, n))$. Pettie and Ramachandran's optimal MST algorithm[19] has a (yet unknown) running time of $\Omega(m)$ and $O(m \cdot \alpha(m, n))$, where $\alpha(\cdot)$ is the extremely slowly growing inverse Ackermann's function.

For graphs with integral edge weights (henceforth called *integer graphs*), linear time algorithms are known for both, $SSSP$[22] and MST[9]. For arbitrarily weighted planar graphs, a linear time $SSSP$ algorithm was presented by Klien, et al.[16].

We study the following two versions of the replacement shortest paths problems:

*The author thanks Abhishek Gupta, John Hershberger, Atri Rudra, and the annonymous reviewers for comments on a preliminary version of this paper, which have significantly improved the presentation and clarity of the same.

†Part of this work was done while the author was at Department of Computer Science, University of California, Santa Barbara, CA - 93106.

(1) Most Vital Arc of a Shortest Path (MVA-SP): Given a weighted graph $G(V, E)$, two nodes s and t, let a shortest path from s to t be $\mathcal{P}_{s,t} = \{e_1, e_2, \ldots, e_k\}$. Compute a shortest s-t path in each of the k graphs $G(V, E \setminus e_i)$ for $1 \leq i \leq k$.

(2) Single Link Failure Recovery (SLFR): Given a weighted graph $G(V, E)$, let a shortest paths tree of node s be $\mathcal{T}_s = \{e_1, e_2, \ldots, e_{n-1}\}$, where $e_i = (x_i, y_i)$ and $x_i = parent_{\mathcal{T}_s}(y_i)$ is the parent of y_i in $\mathcal{T}_s$. Compute a shortest path from y_i to s in each of the $n-1$ graphs $G(V, E \setminus e_i)$ for $1 \leq i \leq n-1$.

A naive algorithm for either of these problems invokes the $SSSP$ algorithm $O(n)$ times, and thus takes $O(mn + n^2 \log n)$ time. We now discuss some applications, related work, and the background of the replacement paths problems, followed by a summary our results, which are listed in Table 1.

1.1 Motivation and Applications

The MVA-SP and SLFR problems find direct applications in *single-link-failure-recover* protocols in communication networks.

The change in a given shortest path between a specified source and destination pair as a shortest path edge fails, is the central focus of the MVA-SP problem. However, a protocol making use of an algorithm for this problem is based on the notion of *global knowledge* of the failed link. That is, the information about the failure of a link e should be propagated to every node of the network so that a message originating from a node s to a node t is now routed along a shortest $s-t$ path in $G(V, E \setminus e)$ rather than the original shortest path. Nisan and Ronen[18] noticed applications of this problem in the area of Algorithmic Mechanism Design and formulated the problem from a new perspective. They describe a network, such as the Internet, which is decentralized entity where multiple self-interested agents own different parts of the network. In such a setting, auction-based pricing seems appropriate for determining the payments to be made to the agents in exchange of using the links. The Vickrey pricing mechanism motivates the owners (agents) of the network resources to bid truthfully, where each agent is compensated in proportion to the marginal utility she brings to the auction. In context of shortest path routing, an edge's utility is the value by which it lowers the shortest path distance - the difference between shortest path lengths with and without the edge. See [11] for a detailed discussion on this. Other applications include finding the k shortest (simple) paths between a pair of vertices[10] and finding k-most-vital-arcs of a network with respect to a given $s-t$ shortest path [17].

The SLFR problem is better suited for the single-link-failure-recovery protocols for networks where the knowledge of the link failure is not propagated across the network, and the failure is discovered only when one is about to use the failed link. As shown in [2], a solution to the SLFR problem can be processed in linear additional time and used for the Alternate Path Routing mechanism used in ATM networks. The reader is referred to [2] for further details.

1.2 Related Work

The MVA-SP was studied for arbitrary undirected graphs by Malik, et al.[17] in their work related to finding the *most vital arc* (*mva*) of an s-t shortest path. The *mva* is defined as the arc (edge) of the graph whose deletion produces the largest increase in the shortest path distance from s to t. Their algorithm was rediscovered by Hershberger and Suri[11], who studied the exact problem defined above in their work on computing the Vickrey payments

for the edges participating in an s-t shortest path. The bound achieved for the MVA-SP problem is $O(m + n \log n)$, which is optimal for arbitrarily weighted graphs in the sense that this much time needs to be invested in computing the original s-t shortest path (a minor flaw in *proof of correctness* of the algorithm of [17] was pointed out and corrected in [1]). The same algorithm works for *directed acyclic graphs* (DAGs). However, this bound is not optimal for planar graphs, DAGs, and integer graphs since shortest paths computations need only linear time for such graphs. Using Thorup's *integer priority queues*[23], the known algorithms of [11, 17] can be implemented to run in $O(m + n \log \log n)$ time for integer graphs. Note that this problem is much different from the *sensitivity analysis of shortest paths* studied in [3, 6, 21]. For instance, see [13] which presents a lower bound of $\Omega(min(m\sqrt{n}, n^2))$ for the directed version of the MVA-SP algorithm in the *path comparison* model for shortest paths algorithms (the directed graphs algorithm presented in [11] had a flaw, as pointed out in [12]). On the other hand, the sensitivity analysis of shortest paths can be performed in near-linear $O(m \cdot \alpha(m, n))$ time[6, 21] even on directed graphs.

The SLFR problem was studied by Bhosle and Gonzalez[2] in their work related to *single link failure recovery* in ATM networks, and an $O(m + n \log n)$ bound was achieved for arbitrarily weighted undirected graphs. For integer graphs, an $O(m+n \log \log n)$ algorithm was presented for the problem. Also, DAGs were shown to admit a linear time algorithm for the SLFR problem. For unweighted undirected graphs, it was shown that the problem can be solved in linear time. For directed graphs, the lower bound construction of [13] applies directly to this problem and imposes the same lower bound of $\Omega(min(m\sqrt{n}, n^2))$ in the path comparison model for shortest path algorithms.

A related problem, termed the *replacement edges with respect to MSTs* is the following:

RE-MST: Given an undirected weighted graph $G(V, E)$, let $\mathcal{T}_{mst} = \{e_1, e_2, \ldots, e_{n-1}\}$ be the MST of G. For each edge $e_i \in \mathcal{T}_{mst}$, find the edge $r(e_i) \in E \setminus e_i$ which connects the two disconnected components of $\mathcal{T}_{mst} \setminus e_i$ in the MST of $G(V, E \setminus e_i)$.

This problem has been studied under the name *sensitivity analysis of MSTs* and efficient algorithms have been developed for it. Tarjan's $O(m \cdot \alpha(m, n))$ time algorithm[21] has possibly been improved by Dixon, et al.[6], who presented an algorithm with a (yet unknown) running time of $\Omega(m)$ and $O(m \cdot \alpha(m, n))$. For planar graphs, a linear time algorithm for this problem was developed by Booth and Westbrook[3].

1.3 Main Results

We reduce a given instance of an MVA-SP or SLFR problem to an instance of the RE-MST problem, which helps us achieve improvements in certain restricted graphs, while matching the current best bounds for others. The reduction takes linear time for planar graphs, and $O(T_{sssp}(m, n) + T_{mst}(m, n))$ time for non-planar graphs, where $T_{sssp}(m, n)$ and $T_{mst}(m, n)$ are the time complexities for computing a single-source-shortest-paths tree and a minimum-spanning-tree respectively of a graph on n nodes and m edges. The general expression for the overall time complexity of our algorithms is $O(T_{sssp}(m, n) + T_{mst}(m, n) + T_{msts}(2n, n))$ for non-planar graphs, and linear (actually, $O(T_{sssp}(m, n) + T_{msts}(m, n))$) for planar graphs. Here, $T_{msts}(m, n)$ is the time required to solve the RE-MST problem defined above (also known as the MST Sensitivity problem) on a graph with m edges, and n nodes.

Our main results are tabulated in Table 1. We improve upon the current best bounds for the MVA-SP problem for arbitrarily weighted planar graphs (undirected or DAGs), integer graphs (undirected or DAGs) and arbitrarily weighted (non-planar) DAGs. For the SLFR

problem, we achieve improvements for arbitrarily weighted planar (undirected) graphs and integer (undirected) graphs. Our algorithms for both the problems take optimal (linear) time for planar graphs, and are within a factor of the inverse Ackermann's function of optimal for integer graphs and arbitrary DAGs.

Note that for non-planar graphs, the improvements hold only for sparse graphs, i.e., $m = o(n \log\log n)$ for integer graphs and $m = o(n \log n)$ for arbitrary DAGs.

As a direct consequence of the `MVA-SP` results, we get improved results for the *k-shortest* (*simple*) *paths* problem for sparse undirected graphs using the recent k shortest paths algorithm of [10], which can be used to solve the undirected version of the problem in $O(k(m + n \log n))$ time. For arbitrarily weighted planar graphs, using our `MVA-SP` algorithm, this algorithm can be implemented to run in $O(kn)$ time, improving the $O(kn \log n)$ previous best bound[15, 10]. For non-planar undirected integer graphs, the bound of $O(k(m + n \log\log n))$ (using integer priority queues[23]) can be improved to $O(k(m + n \cdot \alpha(2n, n)))$.

Problem	**Graph Category**	**Edge Weights**	**Current Results**	**Our Results**
`MVA-SP`	Planar, Undirected or DAG	Arbitrary	$O(n \log n)$	$O(n)$
`MVA-SP`	Undirected or DAG	Integral	$O(m + n \log\log n)$	$O(m + n \cdot \alpha(2n, n))$
`MVA-SP`	DAG	Arbitrary	$O(m + n \log n)$	$O(m \cdot \alpha(m, n))$
`SLFR`	Planar, Undirected	Arbitrary	$O(n \log n)$	$O(n)$
`SLFR`	Undirected	Integral	$O(m + n \log\log n)$	$O(m + n \cdot \alpha(2n, n))$

Table 1: Main Results

2 Preliminaries

We use $G(V, E)$ to denote a graph with vertex set V and edge set E, with $n = |V|$ and $m = |E|$. Each edge $e \in E$ has an associated cost (or weight), $cost_G(e)$, which is a non-negative number. A cut in a graph is the partitioning of the set of vertices V into V_1 and V_2 and is denoted by (V_1, V_2). $E(V_1, V_2)$ represents all the edges across the cut (V_1, V_2).

A shortest path tree $\mathcal{T}_s(G)$ for a node s is a collection of $n-1$ edges of G, $\{e_1, e_2, \ldots, e_{n-1}\}$ where $e_i = (x_i, y_i)$, $x_i = parent_{\mathcal{T}_s}(y_i)$ and the path from node v to s in $\mathcal{T}_s$ is a shortest path from v to s in G. Note that under our notation a node $v \in G$ is the x_i component of as many tuples as the number of its children in $\mathcal{T}_s$ and it is the y_i component in one tuple (if $v \neq s$). Nevertheless, this notation facilitates an easier formulation of the problem. Moreover, the algorithms do not depend on this labeling.

An x-y shortest path $\mathcal{P}_{x,y}(G) = \{e_1, e_2, \ldots, e_k\}$ is a collection of k edges of G forming a shortest path from x to y. The path $\mathcal{P}_{x,y}(G)$ can also be defined as a sequence of nodes lying on the path. The length (total weight) of $\mathcal{P}_{x,y}$ is denoted by $d_G(x, y)$. Note that $\mathcal{P}_{x,y} \cap \mathcal{T}_x = \mathcal{P}_{x,y} \cap \mathcal{T}_y = \mathcal{P}_{x,y}$.

A minimum spanning tree (MST) of a graph $G(V, E)$ is a collection of $n-1$ edges of G: $\mathcal{T}_{mst}(G) = \{e_1, e_2, \ldots, e_{n-1}\}$ spanning all nodes $\in G$ such that the total weight of the tree is less than (or equal to) the weight of any other spanning tree of the graph.

With respect to MSTs, the replacement edge for an edge $e \in \mathcal{T}_{mst}(G)$ is defined as an edge $r(e) \in E \setminus e$ of *minimal* weight which connects the two disconnected components of $\mathcal{T}_{mst}(G) \setminus e$.

When the graph G is understood from the context, we use only $cost(e)$, $\mathcal{P}_{x,y}$, $\mathcal{T}_s$, $d(x,y)$ and $\mathcal{T}_{mst}$ in the above notations.

2.1 Organization

We describe the algorithm for the `MVA-SP` problem in Section 3.1, followed by the algorithm for the `SLFR` problem in Section 3.2. A precomputation step which speeds up the basic algorithms presented in these sections is presented in Section 3.3. The k-shortest simple paths problem is briefly discussed in Section 4. We conclude in Section 5.

3 Fast Algorithms for `MVA-SP` and `SLFR` Problems

3.1 `MVA-SP` Problem

Consider the cut induced in the graph G by deleting an edge $e \in \mathcal{P}_{s,t}$ from $\mathcal{T}_s$, a shortest paths tree of s. The following property of the replacement path for e was proved in [11, 17]. For brevity, we omit its simple proof.

Property 3.1 [**11**, **17**] *The replacement path for an edge $e \in \mathcal{P}_{s,t}$, which is defined as a shortest path from s to t in the graph $G(V, E \setminus e)$, uses exactly one edge from $E(V_s, V_t)$, where V_s is the component of $\mathcal{T}_s \setminus e$ containing s and V_t is the component containing t, and has weight equal to*

$$MIN_{(u,v)\in E(V_s,V_t)}\{d(s,u) + cost(u,v) + d(v,t)\}$$

We borrow from [11], the definition of the "block" of a node v with respect to $\mathcal{T}_s$.

Definition 3.1 [**11**] *If $\mathcal{P}_{s,t} = \{s = x_0, x_1, \ldots, x_k = t\}$ is a shortest path from s to t in $G(V,E)$, and the vertex v belongs to the component in the forest $\mathcal{T}_s \setminus E(\mathcal{P}_{s,t})$ containing x_i, then $block(v) = i$.*

Computing the $block(v)$ values for all vertices $v \in V$ can be easily done in linear time.
Now we are ready to describe our algorithm for the `MVA-SP` problem.

First we eliminate all edges $e = (u,v)$ which have $block(u) = block(v)$. We call such edges *useless edges*. We then construct a new graph $G'(V', E')$ from $G(V,E)$, $\mathcal{T}_s$ and $\mathcal{T}_t$ as follows:

$V' = V$

$E' = \{E \setminus useless\ edges\}$

The cost of an edge $e = (u,v) \in E'$ is assigned as follows:

$if(e \in \mathcal{T}_s), cost_{G'}(e) = 0$

$if(e \notin \mathcal{T}_s), cost_{G'}(e) = d_G(s,u) + cost_G(u,v) + d_G(v,t)$

Once $\mathcal{T}_s$ and $\mathcal{T}_t$ are available, the $cost_{G'}(\cdot)$ values can be computed in constant time per edge. The following two lemmas capture critical properties of the graph $G'(V', E')$.

Lemma 3.1 *The edges of $\mathcal{T}_s$ form the MST of G', and a shortest paths tree of s. That is, $\mathcal{P}_{s,t}(G') = \mathcal{P}_{s,t}(G)$, and $\mathcal{T}_{mst}(G') = \mathcal{T}_s(G') = \mathcal{T}_s(G)$.*

Proof: Trivial, since by the new cost function for the edges of G', the total weight of all the edges of $\mathcal{T}_s$ and $\mathcal{P}_{s,t}$ is 0. □

Lemma 3.2 *The MST of the graph $G'(V', E' \setminus e)$ has weight exactly equal to the weight of the shortest path distance from s to t in the graph $G(V, E \setminus e)$ (which is the weight of the replacement path for e in the graph $G(V, E)$).*

Proof: The proof follows from the Property 3.1 and the constructions of G'. □

The Property 3.1 can be restated as follows:

Property 3.2 *The replacement edge $r(e)$ with respect to MST in G' of an edge $e \in \mathcal{P}_{s,t}(G')$ has weight exactly equal to the weight of the replacement path for e in the graph $G(V, E)$, which is defined as the shortest s-t path in the graph $G(V, E \setminus e)$.*

In effect, we have reduced the `MVA-SP` problem on $G(V, E)$ to the replacement edges problem with respect to MSTs on $G'(V', E')$, and a solution to the latter provides a solution to the former.

The problem of finding the replacement edges with respect to MST in a graph $G(V, E)$ with $|V| = n$ and $|E| = m$ can be solved in time $O(m \cdot \alpha(m, n))$ using the MST sensitivity analysis results presented in [6, 21]. For planar graphs, Booth and Westbrook's algorithm[3] takes only linear time.

We have thus proved the following theorem:

Theorem 3.1 *The `MVA-SP` problem can be solved in $O(T_{sssp}(m, n) + m \cdot \alpha(m, n))$ time for non-planar graphs. For planar graphs, it can be solved in linear time.*

Proof: $O(T_{sssp}(m, n))$ time needs to be invested to compute $\mathcal{T}_s$ and $\mathcal{T}_t$. Construction of $G'(V', E')$ takes only linear time once $\mathcal{T}_s$ and $\mathcal{T}_t$ are available. For planar graphs, $T_{sssp}(m, n)$ is linear[16]. The rest of the proof follows from the discussion above. □

`MVA-SP` for *DAG*s

For *DAG*s, $\mathcal{T}_s$ and $\mathcal{T}_t$ are respectively the single-source shortest paths tree of s and the single-*destination* shortest paths tree of t. Both these trees can be built in linear time. Since the Lemma 3.1 holds for *DAG*s (though not for arbitrary directed graphs), the graph $G'(V', E')$ can be constructed in exactly the same manner as for undirected graphs, the only difference being that the set $\{$*useless edges*$\}$ now also includes all edges of the form $e = (u, v)$ with e being directed from u to v such that $block(u) > block(v)$. In other words, we consider only those edges in the cut $E(V_s, V_t)$ which are directed from a node in V_s to a node in V_t. See [11] for a discussion on this issue.

Furthermore, the constructed graph G' is made *undirected*. This critical change does not affect the solution, since the costs assigned to the edges of G' capture all the information relevant to the replacement paths.

The remaining part of the algorithm is exactly the same as that for undirected graphs and we achieve the time bound of $O(T_{mst}(m, n) + m \cdot \alpha(m, n))$ for the `MVA-SP` problem for arbitrarily weighted *DAG*s.

3.2 `SLFR` Problem

The approach to the `SLFR` problem is essentially the same as that for the `MVA-SP` problem. Consider the cut in the graph G induced by deleting an edge $e = (x, y) \in \mathcal{T}_s$, with $x = parent_{\mathcal{T}_s}(y)$, from $\mathcal{T}_s$. The following property of the replacement path was proved in [2]. For brevity, we omit the proof.

Property 3.3 [2] *The replacement path for $e = (x, y) \in \mathcal{T}_s$ (with $x = parent_{\mathcal{T}_s}(y)$), which is defined as a shortest path from y to s in the graph $G(V, E \setminus e)$ uses exactly one edge across the cut (V_x, V_y), where V_x is the component of $\mathcal{T}_s \setminus e$ containing x and V_y is the component containing y, and has weight equal to*

$$MIN_{(u,v)\in E(V_x,V_y)}\{d_G(s,u) + cost_G(u,v) + d_G(v,s)\}$$

In other words, if we assign the cost to edge $e = (u, v)$ as

$$cost_{G'}(e) = d_G(s,u) + cost_G(u,v) + d_G(v,s)$$

the weight of the replacement path is exactly equal to the cost of the cheapest edge (according to the new $cost_{G'}(\cdot)$ function defined above) across the induced cut.

Note that once $\mathcal{T}_s$ is available, the new costs for the edges can be computed in constant time per edge. We build a new graph $G'(V', E')$ as follows.

$V' = V$.

$E' = E$.

An edge $e = (u, v) \in E'$ is assigned a cost as follows

$if(e \in \mathcal{T}_s),\ cost_{G'}(e) = 0$

$if(e \notin \mathcal{T}_s),\ cost_{G'}(e) = d_G(s,u) + cost_G(u,v) + d_G(v,s)$

It is easy to see that the Lemmas 3.1 and 3.2 hold for the graph $G'(V', E')$ constructed above. That is, $\mathcal{T}_s(G') = T_{mst}(G') = \mathcal{T}_s(G)$.

Restating the Property 3.3 above, we have

Property 3.4 *The replacement edge $r(e)$ (with respect to MST in G') of an edge $e = (x, y) \in MST(G')$ (with $x = parent_{\mathcal{T}_s(G)}(y)$) has weight exactly equal to the weight of the replacement path for e in the graph $G(V, E)$, which is defined as a shortest path from y to s in $G(V, E \setminus e)$.*

We have thus reduced the `SLFR` problem to the replacement edges problem with respect to MSTs, which sets the stage for us to use the sensitivity analysis results from [6, 21] and arrive at the solution in $O(m \cdot \alpha(m, n))$ additional time for non-planar graphs. For planar graphs, the sensitivity analysis algorithm of [3] requires only linear additional time.

We are now ready to state the main theorem of this subsection.

Theorem 3.2 *The* `SLFR` *problem can be solved in linear time on planar graphs. For non-planar graphs, it can be solved in $O(T_{sssp}(m, n) + m \cdot \alpha(m, n))$ time. If a shortest paths tree $\mathcal{T}_s$ of the node s is supplied as part of the input, the $T_{sssp}(m, n)$ term can be dropped from the expression.*

Proof: For planar graphs, $T_{sssp}(m, n)$ is linear[16]. The rest of the proof follows from the discussion above. □

3.3 Fine Tuning the Algorithms

The number of edges in the graph $G'(V', E')$ are reduced from $O(m)$ to at most $2n$ using the following Lemma. The reader is referred to [2] for the proof.

Lemma 3.3 *The replacement edge $r(e)$ of any edge $e \in MST(G)$ with respect to the MST of the graph $G(V, E)$ belongs to the set of edges in $MST(G(V, E \setminus \mathcal{T}_{mst}))$, where $\mathcal{T}_{mst}$ is the MST of $G(V, E)$.*

The edges in $MST(G(V, E \setminus \mathcal{T}_{mst}))$ can be identified in $T_{mst}(m, n)$ time. For integer graphs, this bound in linear in m and n[9]. Thus, the graph $G'(V', E')$ contains only the edges in the set $\mathcal{T}_{mst} \cup MST(G(V, E \setminus \mathcal{T}_{mst}))$. Now, the sensitivity analysis algorithm [6, 21] requires only $O(n \cdot \alpha(2n, n))$ time instead of $O(m \cdot \alpha(m, n))$ since G' has at most $2n$ edges. Hence, our algorithms require only $O(T_{mst}(m, n) + T_{sssp}(m, n) + n \cdot \alpha(2n, n))$ time.

4 k-Shortest (Simple) Paths

The k *shortest simple paths* problem takes as input, an undirected, weighted graph $G(V, E)$, two special nodes, $s, t \in V$, and an integer k. One is required to list out k *simple* paths from s to t in order of increasing weight. We briefly outline the k shortest paths algorithm of [10].

At any point of time, the algorithm maintains in a priority queue data structure, the set of candidates for the next shortest s-t path. The total weight of the path determines its priority. When the i^{th} shortest path is extracted (via a $deleteMin()$ operation), the algorithm generates two new candidates for the $(i+1)^{th}$ shortest path by invoking the `MVA-SP` algorithm twice (on two different graphs). Hence, the size of the priority queue can grow only as large as $2k$, and the time complexity is $O(k(T_{\texttt{MVA-SP}}(m, n) + T_{deleteMin(k)})) = O(k(m + n \log n))$ since $k = O(n!)$ (for planar graphs, $k = O(c^n)$ for some constant c) and $T_{deleteMin(k)} = O(\log k)$, which represents the time required to perform a $deleteMin()$ operation on a priority queue with $O(k)$ keys, and $T_{\texttt{MVA-SP}}(m, n) = O(m + n \log n)$, is achieved. For integer graphs, a time bound of $O(k(m + n \log \log n))$ can be achieved using integer priority queues[23], which guarantee $T_{deleteMin(N)} = O(\log \log N)$.

Replacing the `MVA-SP` algorithm of [11] used in [10] by our algorithm, we can achieve the following bounds for the k shortest simple paths problem: for arbitrarily weighted planar undirected graphs, $O(kn)$, and for non-planar undirected integer graphs, $O(k(m+n \cdot \alpha(2n, n)+ \log \log k)) = O(k(m + n \cdot \alpha(2n, n)))$ (again, using integer priority queues).

The version of the k shortest paths problem which does not require the paths to be *simple* has been solved in optimal time of $O(T_{sssp}(m, n) + k)$[7]. For DAGs, the two versions of the problem are identical.

5 Concluding Remarks

A generalization of the the `MVA-SP` algorithm is the k *most vital arcs with respect to shortest paths* (`k-MVA-SP`) problem, which asks one to find the $k \geq 2$ edges of the graph, whose deletion causes the largest increase in the shortest path distance from s to t in the graph than deleting any other set of k edges from the graph. The problem was shown to be NP-Complete for arbitrary k in [1]. Also, the exponential algorithm for the same presented in [17] was shown to be incorrect in [1]. The `MVA-SP` algorithm can easily be generalized to an algorithm for the `k-MVA-SP` problem. For *fixed* $k \geq 2$, the time complexity of this algorithm would be $O(n^{k-1} \cdot T_{\texttt{MVA-SP}}(m, n))$, which boils down to $O(n^k)$ for planar graphs, $O(n^k(m + n \cdot \alpha(2n, n)))$ for (non-planar) integer graphs, and $O(n^k m \cdot \alpha(m, n))$ for arbitrary DAGs. We omit further details from this abstract due to space limitations.

An improvement in the `RE-MST` problem for finding the replacement edges with respect to MSTs for integer graphs would imply the corresponding improvements to the problems studied here. The fine-tuning trick described in Section 3.3 can be used to improve the

sensitivity analysis results of [6, 21] for MSTs on integer graphs from $O(m \cdot \alpha(m,n))$ to $O(m + n \cdot \alpha(2n,n))$.

The algorithms of [6, 21] were designed for arbitrary graphs, thus suggesting that for integer graphs, it may be possible to eliminate the $\alpha(\cdot)$ term from the time complexity expression.

The directed versions of both, `MVA-SP` and `SLFR`, are significantly more difficult as they admit a lower bound of $O(min(m\sqrt{n}, n^2))$[2, 13] for a certain *path comparison* class of shortest paths algorithms (a flaw in the directed graphs algorithm of [11] was pointed out in [12]). See [13, 14] for the details on the model.

We believe that the technique of using MSTs in shortest path problems will also find applications in shortest path problems apart from those studied here.

References

[1] A. BarNoy, S. Khuller, and B. Schieber. The complexity of finding most vital arcs and nodes. *Technical Report CS-TR-3539, University of Maryland, Institute for Advanced Computer Studies, MD*, 1995.

[2] Amit M. Bhosle and Teofilo F. Gonzalez. Efficient algorithms for single link failure recovery and its application to atm networks. In *Proc. 15th IASTED Intl. Conf. on PDCS*, volume 1, pages 87 – 92. ACTA Press, 2003.

[3] Heather Booth and Jeffery Westbrook. A linear algorithm for analysis of minimum spanning and shortest-path trees of planar graphs. *Algorithmica*, 11(4):341–352, 1994.

[4] B. Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *JACM*, 47:1028-1047, 2000.

[5] E. W. Dijkstra. A note on two problems in connection with graphs. In *Numerische Mathematik*, pages 1:269-271, 1959.

[6] B. Dixon, M. Rauch, and R. E. Tarjan. Verification and sensitivity analysis of minimum spanning trees in linear time. *SIAM J. Comput.*, 21(6):1184–1192, 1992.

[7] D. Eppstein. Finding the k shortest paths. In *35^{th} IEEE FOCS*, pages 154-165, 1994.

[8] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *JACM*, 34:596-615, 1987.

[9] M. L. Fredman and D. E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *JCSS*, 48:533-551, 1994.

[10] J. E. Hershberger, M. Maxel, and S. Suri. Finding the k shortest simple paths: A new algorithm and its implementation. In *SIAM-ALENEX*, 2003.

[11] J. E. Hershberger and S. Suri. Vickrey prices and shortest paths: What is an edge worth? In *42^{nd} IEEE FOCS*, pages 252-259, 2001.

[12] J. E. Hershberger and S. Suri. Erratum to "Vickrey prices and shortest paths: What is an edge worth?". In *43^{rd} IEEE FOCS*, pages 809–809, 2002.

[13] J. E. Hershberger, S. Suri, and A. M. Bhosle. On the difficulty of some shortest path problems. In *20^{th} STACS*, pages 343–354. Springer-Verlag, 2003.

[14] D. R. Karger, D. Koller, and S. J. Phillips. Finding the hidden path: Time bounds for all-pairs shortest paths. In *32nd IEEE FOCS*, pages 560-568, 1991.

[15] N. Katoh, T. Ibaraki, and H. Mine. An efficient algorithm for k shortest simple paths. In *Networks.*, pages 12:411-427, 1982.

[16] Philip Klein, Satish Rao, Monika Rauch, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. In *26th ACM STOC*, pages 27–37. ACM Press, 1994.

[17] K. Malik, A. K. Mittal, and S. K. Gupta. The k most vital arcs in the shortest path problem. In *Oper. Res. Letters*, pages 8:223-227, 1989.

[18] N. Nisan and A. Ronen. Algorithmic mechanism design. In *31st Annu. ACM STOC*, pages 129-140, 1999.

[19] S. Pettie and V. Ramachandran. An optimal minimum spanning tree algorithm. In *Automata, Languages and Programming*, pages 49-60, 2000.

[20] R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36:1389-1401, 1957.

[21] R. E. Tarjan. Sensitivity analysis of minimum spanning trees and shortest path problems. *Inform. Proc. Lett.*, 14:30–33, 1982.

[22] Mikkel Thorup. Undirected single source shortest path in linear time. In *38th IEEE FOCS*, pages 12–21, 1997.

[23] Mikkel Thorup. Integer priority queues with decrease key in constant time and the single source shortest paths problem. In *35th ACM STOC*, pages 149–158. ACM Press, 2003.