Global Journal of Technology and Optimization

Hosseini, Global J Technol Optim 2017, 8:2
DOI: 10.4172/2229-8711.1000214

Research Article OMICS International

The Widest Path Postoptimality Analysis

Ahmad Hosseini*

East Institute of Science and Technology, Tehran, Iran

***Corresponding author:** Ahmad Hosseini, East Institute of Science and Technology, Tehran, Iran, Tel: 98-935-442-1022; E-mail: ahmad.s.hosseini@gmail.com

Received date: June 28, 2017; **Accepted date:** August 04, 2017; **Published date:** August 11, 2017

Abstract

The notion of postoptimality analysis is a well-established topic in operations research. This paper studies one aspect of the robustness of optimal solutions to the widest path (WP), more precisely, postoptimality analysis of the WP problem. The WP postoptimality analysis here deals with infimum and supremum perturbations which determine multiplicative changes each individual arc can tolerate preserving the optimality of a given WP. It is also elucidated how to determine these marginal values for all arcs, and an algorithm for computing all such values is proposed, which is shown to have complexity of O(m) or O(|P*|m)for general networks with arcs and optimal WP P*.

Keywords: Operations Research; Path Finding; Networks/Graphs; Widest Path; Postoptimality

Introduction

Path-finding problem, such as widest path problem (also known as the bottleneck shortest path problem), is a fundamental component of many important applications in different fields including logistics, networking, shipping, emergency response, project scheduling, cable routing, robotics, message in communication systems, QoS (quality-of-service) routing, electrical routing, cycle routing, and molecular biology [1-6]. Assuming the relationships between nodes in a network are weighted by a capacity of some sort, the widest path problem involves finding the path between a source node s and a sink node t that maximizes the minimum capacity in the path Ahuja et al. [2]

In many cases the data used are inexact or uncertain. In such cases, postoptimality analysis is necessary to determine the credibility of the optimal solution and conclusions based on that solution. In fact, postoptimality analysis is an important element in making decisions and it investigates the effect of changes on a given optimal solution to a problem. Therefore, such a study can be useful in assessing the `robustness' of an optimal solution to inaccuracy or variability in the given input data. The basic topic of sensitivity analysis is the special case when the value of a single element is subject to change. The goal of such perturbation is to determine the maximum and minimum additive changes of a given individual weight preserving the optimality of a given optimal solution [7]. Such tolerance calculations have been previously investigated for different problems in different contexts such as transportation, minimum spanning tree, traveling salesman, shortest path, dynamic graph techniques, Vickrey payments, and maximally reliable path related traditional sensitivity analysis problems for shortest path tree and MST have been consideredinitially by Shier and Witzgall [7-11].

A thirty-year-old result of Tarjan shows that MST sensitivity analysis can be solved in O (mα (m, n)) time, where m is the number of arcs, n the number of nodes, and α the inverse-Ackermann function. Moreover, he gave a linear time reduction from shortest path sensitivity analysis to MST sensitivity analysis. Tarjan's algorithm has not seen any unqualified improvements, though Dixon et al. presented two MST sensitivity algorithms, one running in expected linear time and another whose complexity is known to be bounded by O(mα(m, n)). In particular, Harel showed that if the arc costs are polynomially bounded inn, then on a unit-cost random access machine the MST verification and sensitivity analysis problems can be solved deterministically in linear time. Pettie presents a deterministic algorithm for computing the sensitivity of an MST or shortest path tree in O (m log α(m, n)) time [12]. This work improves upon the long standing bound of O (m α (m, n)) established by Tarjan. Shier and Witzgall (1980) propose several algorithms for sensitivity analysis of shortest path trees. Gus field (1983) shows that two of their algorithms can be implemented in O (m log n) time and O (m) space, and noted that his techniques also apply to sensitivity analysis of minimum spanning trees. Spira and Pan (1975) give lower bounds on the amount of computation required to update shortest path trees [13,14]. Kurzhanski and Varaiyab [15] study the problem of reachability for linear systems in the presence of uncertain input disturbances. Hershberger and Suri [16] focused on the algorithmic aspect of computing the Vickrey payments in the context of shortest path routing. In a case where multiple self-interested agents owned portions of the network. They showed that the Vickrey prices (upper tolerances in a shortest path problem) can be computed in O (m + n log n) time. For an extensive account on computational issues of tolerances in combinatorial optimization, such as MST problem, minimum Hamiltonian path problem and the traveling salesman problem (TSP), published after 1980 [17-19].

To complement previous works, this paper addresses the issue of postoptimality analysis of WP problem in networks to deal with determining the supremum and infimum multiplicative changes that an arc can tolerate preserving the optimality of a pre-obtained WP. More precisely, we aim to determine how much the arc's capacity can be multiplicatively changed without affecting the WP between two given nodes in the system. We propose an algorithm to do the postoptimality analysis for a given WP restricted to the case when a single arc is changed by a multiplicative factor. The algorithm runs in time O (m |P*|) (or even O (m) if one is only interested in the sup tolerances) instead of O (m2) using a naive approach. Examples are also provided to help understand the algorithm and relations.

Preliminaries

Let N=(V, A) describe a network, either directed or undirected, with V and $A \subseteq V \times V$ representing the set of nodes and arcs, respectively. We let n=|V | and m=|A| denote the number of nodes and arcs, respectively. Further, we designate two special nodes, the source node s ∈V and the sink node $t \in V$. Each arc $(i, j) \in A$ has a positive capacity parameter $c_{ij} \in (0, +\infty)$ associated with it. The capacity measures the maximum amount of flow that can be transmitted through the arc. A path P from v1 to vk is defined by a sequence of nodes v1; v2.vk-1; vk with the property that every consecutive pair of nodes vi and vi+1 in the sequence is connected by an arc, more precisely $P = \cup_{i=1}^{k-1} (v_i, v_{i+1})$. The capacity of a (directed) path P is the minimum arc capacity in P. That is, the capacity C (P) of a path P i

$$C(P) = \min_{(i,j) \in P} c_{ij} = \min_{(i,j) \in A} \left\{ c_{ij} \chi_{ij}^{P} + M(1 - \chi_{ij}^{P}) \right\}$$

where $x_{ij}^{p} = 1$ if arc (i, j) belongs to path P and zero otherwise, and M is a constant such that $M \geq \max_{(i,j \in A)} \{c_{ij}\}$. The MCP problem is dealing with the maximization of function (1). Let P={Pk} denote the set of all s-t-paths in N, i.e., all paths from source s to sink t. The set P does not depend on the capacity parameters. We are specifically interested in two subsets of P, namely the sets $P_{(i,j)}^{+}$ and $P_{(i,j)}^{-}$ that comprise all s-t-paths in N that does and does not include arc (i, j), respectively.We also let C (P):= max P∈pC(P) denote the optimal value for the MCP problem, $P_{(i,j)}^{-}$ denote the capacity of an MCP in $P_{(i,j)}^{+}$ and denote the capacity of an MCP in $P_{(i,j)}^{-}$. In the following sections, we assume that the network is s-t-connected and that an MCP is already given/ obtained and we are asked to investigate the postoptimality analysis issue with respect to it. It is possible to adapt most shortest path algorithms to compute widest paths, by modifying them to use the bottleneck distance instead of path length. However, in many cases even faster algorithms are possible [2]. To obtain a WP, an WP algorithm can be established by modifying, e.g., Dijkstra's algorithm. To do so, we need to initialize the label d (.) of each node to 0 and the source node s to 1. Further, we update the distance label of a node j if and only if for some node $i \in V, (i, j) \in A$ and $d(j) < min\{d(i), c_{i,j}\}$ i.e., we set $d(j) := max\{d(j), min\{d(i), c_{ij}\}\}$. The complexity of this MCP algorithm is O(m + n log n) for directed networks using a Fibonacci or hollow heap (Hansen et al., 2015) and O(m) for undirected networks using Thorup's algorithm [20].

Widest path and post optimality

This section discusses the multiplicative perturbations for the WP problem on N=(V, A) with a given WP P*. Here, we establish approaches for computing the inf and sup tolerances of all arcs with respect to P*. Henceforth, we let $N_{(i,j)}^{\times \lambda}$ denote the network N in which the capacity of arc (i, j) is replaced by λc_{ij} with all other capacity parameters staying unchanged, and $C(N_{(i,j)}^{\times \lambda})$ represents the capacity of the WP in $N_{(i,j)}^{\times \lambda}$. We define the multiplicative inf tolerance Ip* (I, j) and multiplicative sup tolerance Sp* (i,j) of arc (i, j) with respect to P* as

$$I_{p*}(i,j) = \inf_{\lambda \in R} \left\{ \lambda \in (0,1] \,\middle|\, P^* \text{ is an WP in } N_{(i,j)}^{\times \lambda} \right\}$$

$$Sp^*(i,j) = \sup_{\lambda \in R} \left\{ \lambda \in [1, +\infty) \,\middle|\, P^* \text{is an WP in } N_{(i,j)}^{\times \lambda} \right\}$$

We remark that the multiplicative inf and sup tolerances do depend on a particular WP. For better illustration, let us consider the WP instance presented in Figure 1. As it is seen, there are three s-t-paths from s=1 to t=6 among which paths P*=P1-2-3-6 and P**=P1-2-4-6 are the network's WPs with capacity 3. It is easy to verify that the following relations hold:

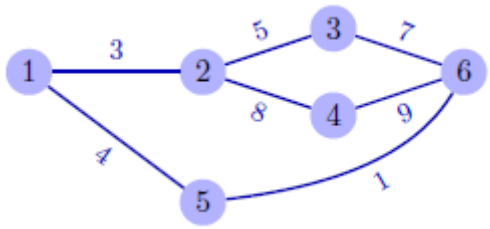

Figure 1: An WP instance.

$$1/3 = I_{p*}(1,2) = I_{p**}(1,2) = 1/3 \qquad 5/3 = S_{p*}(1,2) \neq S_{p**}(1,2) = +\infty$$

$$3/5 = I_{p*}(2,3) \neq I_{p**}(2,3) = 0 \qquad +\infty = S_{p*}(2,3) = S_{p**}(2,3) = +\infty$$

Property 3.1. (a) Let arc $(i, j) \in P^*$ where P* is a WP in N. Then, $(i, j) \in \cap_k P_k \Leftrightarrow I_{p*}(i, j) = 0$

(b $(i, j) \notin \cup_k P_k \Rightarrow S_{p*}(i, j) = +\infty$.

Note that only the direction " ⇒ " of Property 3.1(b) holds in general, but not the reverse direction " ⇐ "

This is illustrated in the WP instance presented in Figure 2. Here, there are three paths from node 1 to

node 6 among which the path P*=P1-2-4-6 is the unique WP with capacity of 3. It is easy to see that

$S_{p*}(5,6) = +\infty$, but on the other hand, (5, 6) ∈ P 1-5-6

Theorem 3.2. Let N=(V, A) be an WP instance and P* a WP in N.

$$(a)\ if\ (i,j) \in P^*,\ then\ S_{p*}(i,j) = \begin{cases} +\infty & if\ C(P) \geq \lim_{\lambda \to +\infty} C(N_{(i,j)}^{X\lambda}) \\ \frac{1}{c_{ij}} \min_{(i',j') \in P^* \setminus (i,j)} \{c_{i',j}\} & if\ C(P) < \lim_{\lambda \to +\infty} C(N_{(i,j)}^{X\lambda}) \end{cases}$$

(b) *If* $(i,j) \notin P^*$, *then* $S_{p*}(i,j) =$

$$\begin{cases} +\infty & \text{if } C(P) \geq \lim\limits_{\lambda \to +\infty} C(N^{X\lambda}_{(i,j)}) \\ \frac{1}{c_{ij}} C(P) & \text{if } C(P) < \lim\limits_{\lambda \to +\infty} C(N^{X\lambda}_{(i,j)}) \end{cases}$$

(c) $C(P^{-}_{(i,j)}) = \lim\limits_{\lambda \to 0} C(N^{x\lambda}_{(i,j)})$ *and* $C(P^{+}_{(i,j)}) =$

$$\max_{P \in P^{+}_{(i,j)}} \min_{(i',j') \in P} \left\{ c_{i'j'} \right\}$$

(d) $I_{p*}(i,j) = \begin{cases} \frac{1}{c_{ij}} C(P^{-}_{(i,j)}) & \text{if } (i,j) \in P^* \\ 0 & \text{otherwise} \end{cases}$

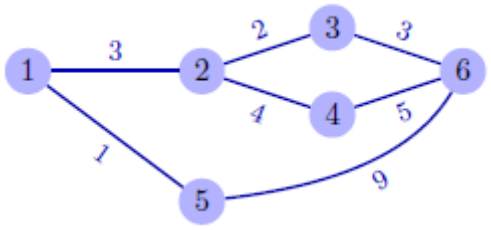

Figure 2: An WP instance for Property 3.1(b).

Proof. For part (a), it should be noted that there can be only two possibilities when dealing with an arc$(i,j) \in P^*$:either P* will remain a WP for $N^{\times\lambda}_{(i,j)}$ when $\lambda \to +\infty$or it will not. P* is a WP for $N^{\times\lambda}_{(i,j)}$while $\lambda \to +\infty$if and only if the condition $C(P) \geq \lim\limits_{\lambda \to +\infty} C(N^{\times\lambda}_{(i,j)})$holds. By definitions of multiplicative sup tolerances, P* remains a WP under such circumstance, and setting $S_{p*}(i,j) = +\infty$ is justified. The other case is when P* is no more an WP in $N^{\times\lambda}_{(i,j)}$ while$\lambda \to +\infty$, and this happens if and only if the condition $C(P) < \lim\limits_{\lambda \to +\infty} C(N^{\times\lambda}_{(i,j)})$holds. In such situation, in order for P* to remain a WP for N (justifying the definitions of sup tolerances), any increase more than $\frac{1}{c_{ij}} \min_{(i'j') \in P^* \setminus (i,j)} \left\{ c_{i'j'} \right\}$on cij will create another WP (violating the definitions of sup tolerances). Hence, our settings are validated. For illustrative proof, we refer to Figures 1-5.The proof for Part (b) is established along the same lines as in the proof for Part (a). Part (c) is establish by considering the fact that the WP instance $N^{\times\lambda}_{(i,j)}$ actually represents N (V,A \ (i, j))when λ approaches zero. For Part (d), let $(i,j) \in P^*$By the definition of inf tolerance (2), we have $I_{p*}(i,j) = \inf\limits_{\lambda \in R} \left\{ \lambda \in (0,1) \middle| \lambda_{c_{i,j}} \geq C(P^{-}_{(i,j)}) \right\} = \frac{1}{c_{ij}} C(P^{-}_{(i,j)})$

By employing Theorem 3.2, we can calculate the exact values of all multiplicative inf and sup tolerances of an arc (i, j) with respect to a given WP in the same asymptotic time complexity as at most two WP algorithms. Therefore, the total computational effort will be O (m2 + mn log n) for directed networks and O (m2) for undirected networks. However, we now use the previous results to develop the following WP-IST algorithm for computing the tolerances of all arcs in a reduced computational time. To proceed, we need to define an auxiliary network, called residual network.

Let N=(V, A) be an WP instance and let P be an arbitrary s-t-path in N with capacity C (P). The residual network with respect to path P is $N^r_p = (V, A^r_p)$where$A^r_p = \left\{ (i,j) \in A \middle| c_{ij} > C(P) \right\}$. Consequently, $N^r_{p\setminus(k,l)} = (V, A^r_{p\setminus(k,l)})$, where $A^r_{p\setminus(k,l)} = \left\{ (i,j \in A \middle| c_{ij} > C(P\setminus(k,l)) \right\}$. It is easy to see that with respect to some WP P* that the residual network $N^r_{P^*} = (V, A^r_{p^*})$ is an s-t-disconnected network. More precisely, the node set V can be partitioned into at least two disconnected components. Therefore, we define a possible cut. Let Vs denote the set of nodes reachable from s in $N^r_{P^*}$ and Vt be the set of nodes that are reachable from t in $N^r_{P^*}$, and let Vs and Vt be non-empty. Then in an undirected network, define Cut (N^r_{p*}, V_s, V_t) as the set of pairs (i, j) satisfying either that $i \in V_s$and $j \in V_t$or $i \in V_t$ and j. Similarly, in a directed network define C ($N^r_{P^*}$, Vs, Vt) as the set of pairs (i, j) satisfying that i ∈ Vs and$j \in V_t$. Cut ($N^r_{P^*}$, Vs, Vt) can also be translated in the original network N. Namely, Vs consists of all nodes i for which there is a path $P_{s \to i}$ whose capacity is strictly larger than C(P) in N, and Vt contains all nodes j from which there is a path $P_{j \to t}$whose capacity is strictly larger than C(P) in N.

Now, assuming that a WP P* is already obtained, we perform the WP-IST Algorithm to efficiently calculate all arcs' tolerances for an WP instance N=(V, A). We give the algorithm in a pseudo code (WP-IST Alg.) which runs in O (m) (if only the sup tolerances are concerned) or O (|P*| m) time (if both the sup and inf tolerances are concerned). The WP-IST algorithm originally was developed to calculate the sup tolerances; however, it is capable to calculate the inf tolerances also [21].

WP-IST algorithm

Step 0: Preparation

The WP instance N=(V, A) is at hand. So is a WP P*

Step 1: Construction

1.1. Set (k, l)=arg $\min_{(i,j) \in P^*} \left\{ c_{ij} \right\}$ (note: (k, l) may not be unique)1.2. Set C (P)=C(P*)=$\min_{(i,j) \in p*} \left\{ c_{ij} \right\}$

1.3. Construct $N^r_{P^*} = (V, A^r_{p^*})$ and Cut $(N^r_{p^*}, V_8, V_t)$ on $N^r_{P^*}$

1.4. Set C (P*\ (k, l))=$\min_{(i,j \in P* \setminus (k,l))} \left\{ c_{ij} \right\}$

1.5.

$Construct\ N^{r}_{P*\setminus(k,l)} = (V,\ A^{r}_{p*\setminus(k,l)})\ and\ Cut\ (N^{r}_{p*\setminus(k,l)},\ V_s,\ V_t)\ on\ N^{r}_{p*\setminus(k,l)}$

Step 2: Search over arc set A\ P*

2.1. For $(i,j) \in A$and $(i,j) \notin P^*$set $I_{p*}(i,j)$= 02.2. For$(i,j) \in A$, $(i,j) \notin P^*$, and (i,j) $\in$Cut(N^{r}_{p*}, V_s, V_t)set $S_{p^*}(i,j) = \frac{1}{c_{ij}} C(P)$

2.3. For$(i,j) \in A$,$(i,j) \notin P^*$, and $(i,j) \notin$ Cut (N^{r}_{p*}, V_s, V_t) set $S_{p^*}(i,j) = +\infty$

Step 3: Search over arc set $A \cap P^*$

3.1. For $(i,j) \in A$and $(i,j) \in P^*$:set $I_{p*}(i,j) = \frac{1}{c_{ij}} C(P^{-}_{(i,j)})$

3.2. For$(i,j) \in A$,$(i,j) \in P*$, and cij > C(P) set $S_{p^*}(i,j) = +\infty$

3.3. For$(i,j) \in A$,$(i,j) \in P*$,cij=C(P) and (i, j) $\in$ Cut $(N^{r}_{p*\setminus(k,l)}, V_s, V_t))$ $set\ S_{p*}(i,j) = \frac{1}{c_{ij}} \min_{(i',j') \in P*\setminus(k,l)} \{c_{i'j'}\}$

3.4. For$(i,j) \in A$, (i, j)$\in P^*$, cij=C (P), and $(i,j) \notin$Cut $(N^{r}_{p*\setminus(k,l)}, V_s, V_t))$ set $S_{p^*}(i,j) = +\infty$

At Step 2, the algorithm makes use of Cut (N^{r}_{p*}, V_s, V_t) for any arc $(i,j) \in A\setminus P^*$to determine the arcs whose capacities' changes impact the optimality of P*. Those arcs are exactly the ones that belong to Cut (N^{r}_{p*}, V_s, V_t)\ P* and were discussed in Theorem 3.2. Note that any arc (i, j) $\in$ Cut (N^{r}_{p*}, V_s, V_t)can be a bottleneck arc whose capacity's increase may affect the optimality of the already obtained WP, because it may create a path of capacity strictly larger than C (P*) = C (P). Having detected those bottleneck arcs, we correctly set the tolerances' values for all arcs $(i,j) \in A\setminus P^*$ using the results of corresponding properties and theorems established previously.Analogously, at Step 3, the algorithm sets the inf tolerances for all arcs (i,j) $\in A \cap P^*$again using the results of Theorem 3.2. Then it sets the sup tolerances for arcs (i, j) $\in A \cap P^*$whose capacities are larger than C (N). A closer scrutiny and another use of Theorem 3.2 reveal that the arcs in $A \cap P^*$ that may affect the optimality of P* are exactly those belonging to Cut $(N^{r}_{p*\setminus(k,l)}, V_s, V_t))$ with capacity C (P). In other words, any arc $(i,j) \in A \cap P*$ with cij=C (P) can be a bottleneck arc. Indeed, arc (i,j) $\in$ Cut$N^{r}_{p*\setminus(k,l)}, V_s, V_t \cap P*$whose capacity is C(P) can create a better WP, so we should limit it (by sup tolerance) using Theorem 3.2.Taking the fact $A\setminus P^* \cup A \cap P^*$= A into account, the algorithm determines the multiplicative sup tolerances of all arcs in O (m) time. Therefore, if only the sup tolerances are required, the running time of the WP-IST algorithm is O (m), which outperforms the naive O (m2)-implementation discussed earlier. If both inf and sup tolerances are concerned, the complexity is O (m) + O (|P*|m)=O (|P*|m). The bottleneck operation of the algorithm is the scanning of arcs (i, j) $\in A \cap P^*$at Step 3.1, which takes O (|P*|m) time. The algorithm performs the construction step in O (m) time.

Example 3.3. Let us consider the WP instance presented in Figure 3. There exist several s-t-paths from s=1 to t=9 among which the path P*=P1-5-8-9 is a WP of capacity 5. It can be seen that

(k, l)=(8, 9)= $\arg \min_{(i',j') \in P*} \{c_{i',j'}\}$ C (P)=C (P*)=5

$\arg \min_{(i',j') \in P*\setminus(k,l)} \{c_{i',j'}\} = (1,5)$ C (P*\ (k,l))=6

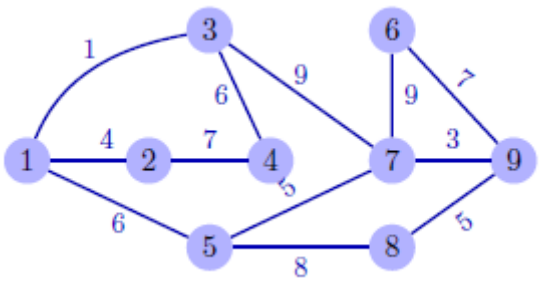

Figure 3: An WP instance N=(V, A) for the WP-IST algorithm.

Having this information at hand, we can now construct the residual networks as follows:

$N^{r}_{P^*} = (V, A^{r}_{p^*})$ $\quad where A^{r}_{p*} = \{(i,j) \in A \mid c_{ij} > 5\}$,

$N^{r}_{p*\setminus(k,l)} = V,\ A^{r}_{p*\setminus(k,l)}$, $\quad where\ A^{r}_{p*\setminus(k,l)} = \{(i,j) \in A \mid c_{ij} > 6\}$

Finally, employing the algorithm over sets (A \ P*) and $A \cap P^*$we can calculate all arcs' tolerances.

To this end, we need the following quantities:

$C(P^{-}_{(1,5)}) = 4$ $\quad C(P^{-}_{(5,8)}) = 5,$ $\quad C(P^{-}_{(8,9)}) = 5,$

We denote the multiplicative inf and sup tolerances of each arc (i; j) by tolerance interval [Ip*(i,j), Sp*(i.j)] in the network (Figure 5). As examples, we show the calculations and considerations taken by the algorithm.

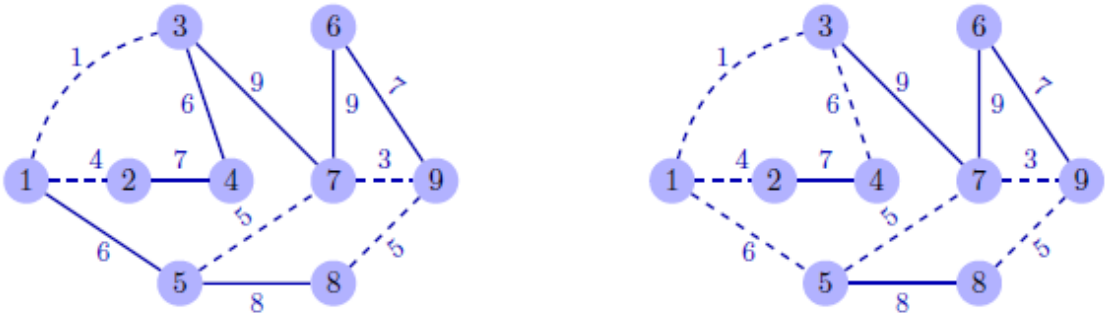

Figure 4: The residual $N^r_{P^*} = (V, A^r_{p^*})$ (left side) and the residual network $N^r_{p\setminus(k,l)} = (V, A^r_{p\setminus(k,l)})$ (right side).

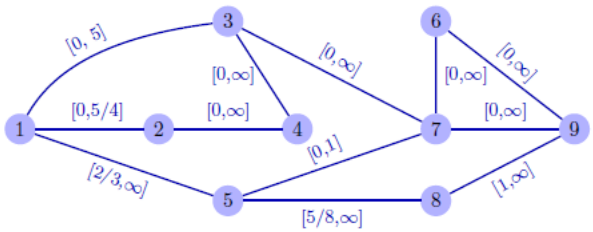

Figure 5: The WP instance of Figure 3 with all arcs' multiplicative inf and sup tolerances.

The validity of the obtained values can be checked by Theorem 3.2.

Vs={1} and Vt={3, 4, 6, 7, 9}

Cut $N^r_{P^*} = \{(1, 3)\}$:

$(1,5) \in P^* \Rightarrow I_{p^*}(1,5) = 2/3, \quad c_{15} > C(P) = 5 \quad S_{p^*}(1,5) = \infty$

$(5,8) \in P^* \Rightarrow I_{p^*}(5,8) = 5/8 \quad c_{58} > C(P) = 5 \Rightarrow S_{p^*}(5,8) = \infty$

$(8,9) \in P^* \Rightarrow I_{p^*}(8,9) = 5/5, \quad c_{89} = C(P)5\&(8,9) \notin Cut(N^r_{p^*\setminus(k,l)}, V_8, V_t) \Rightarrow S_{p^*}(8,9) = \infty$

$V_s = \{1,5,8\}$ and $V_t = \{2, 3, 4, 6, 7, 9\}$

$Cut\ (N^r_{p^*}, V_s, V_t) = \{(1,2), (1,3), (5,7), (8,9)\}$:

$(1,2) \notin P^* \Rightarrow I_{P^*}(1,2) = 0, (1,2) \in Cut(N^r_{p^*}, V_s, V_t) \Rightarrow S_{p^*}(1,2) = 5/4,$

$(1,3) \notin P^* \Rightarrow I_{P^*}(1,3) = 0, (1,3) \in Cut(N^r_{p^*}, V_s, V_t) \Rightarrow S_{p^*}(1,3) = 5,$

$(5,7) \notin P^* \Rightarrow I_{P^*}(5,7) = 0, (5,7) \in Cut(N^r_{p^*}, V_s, V_t) \Rightarrow S_{p^*}(5,7) = 1$

For all other arcs {(2, 4); (3, 4); (3, 7); (6, 7); (6, 9); (7, 9)}, it holds that they are not in P* and thus their inf tolerance is set to zero. Moreover, they are not in Cut($N^r_{p^*}, V_s, V_t$), and thus their sup tolerance is set to ∞ Summary and Discussion

This paper addresses the issue postoptimality analysis to deal with infimum and supremum multiplicative perturbations in widest path (WP) problem in networks. We propose an algorithm to do the WP postoptimality analysis for given WPs restricted to the case when an arc is changed by a multiplicative factor. We however believe that there is room for improvement in our algorithm, in particular, Step 3. Moreover, modern computers can also well take advantage of the algorithm's inherent parallelism at Step 3.

References

1. Berman O, Handler GY (1987) Optimal minimax path of a single service unit on a network to nonservice destinations. Transportation Science 21: 115-122.
2. Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory, algorithms, and applications. Prentice Hall, Upper Saddle River (N.J.).
3. Dietrich B, Vohra R (1993) Mathematics of the Internet. Prentice Hall. Upper Saddle River (N.J.).
4. Gal T (1995) Sensitivity analysis, parametric programming, and related topics: Degeneracy, multicriteria decision making redundancy. Walter de Gruyter, Berlin, Germany.
5. Chang A, Amir E (2007) Reachability under uncertainty. In: Proceedings of the Twenty-Third Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-07). AUAI Press, Corvallis, Oregon 41-48.

6. Hosseini A (2015) Time-dependent optimization of a multi-item uncertain supply chain network: A hybrid approximation algorithm. Discrete Optimization 18: 150-167.
7. Muller-Merbach H (1968) Sensitivity analysis of transport problems of the linear planning calculation (mit ALGOLProgramm).
8. Elmaghraby SE (1964). Sensitivity analysis of multiterminal ow networks. Operations Research 12: 680-688.
9. Dixon B, Rauch M, Tarjan RE (1992) Verification and sensitivity analysis of minimum spanning trees in linear time. SIAM Journal on Computing 21: 1184-1192.
10. Pettie S (2003) On the shortest path and minimum spanning tree problems. Ph.D. thesis.
11. Hosseini A, Baiki BK (2017) An addendum on postoptimality of maximally reliable path. Journal of Mathematics Research 9: 23-29.
12. Pettie S (2005) Sensitivity analysis of minimum spanning trees in sub-inverse-ackermann time. In: International Symposium on Algorithms and Computation. Springer 964-973.
13. Spira PM, Pan A (1975) On finding and updating spanning trees and shortest paths. SIAM Journal on Computing 4: 375-380.
14. Libura M (1991) Sensitivity analysis for minimum hamiltonian path and traveling salesman problems. Discrete Applied Mathematics 30 2:197-211.
15. Kurzhanski A, Varaiyab P (2005). Reachability under uncertainty and measurement noise. Mathematical and Computer Modeling of Dynamical Systems 11: 183-194.
16. Hershberger J, Suri S (2001) Vickrey prices and shortest paths: What is an edge worth? In: Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on. IEEE 252-259.
17. Hsu LH, Jan RH, Lee YC, Hung CN, Chern MS (1991) Finding the most vital edge with respect to minimum spanning tree in weighted graphs. Information Processing Letters 39: 277-281.
18. Venema S, Shen H, Suraweera F (2000) NC algorithms for the single most vital edge problem with respect to all pairs shortest paths. Information Processing Letters 10: 51-58.
19. Thorup M (1997) Undirected single source shortest paths in linear time. In: 38th Annual Symposium on the Foundations of Computer Science. Miami Beach, FL 12-21.
20. Hansen T, Kaplan H, Tarjan R Zwick U (2015). Hollow heaps. In: Proceedings of ICALP 2015. Vol. 9134 of Lecture Notes In Computer Science. Springer 689-700.
21. Kurzhanski A, Varaiyab P (2006) On reachability under uncertainty. SIAM Journal on Control and Optimization 41: 181-216.