SENSITIVITY ANALYSIS FOR SHORTEST PATH PROBLEMS AND MAXIMUM CAPACITY PATH PROBLEMS IN UNDIRECTED GRAPHS

Ramkumar Ramaswamy, James B. Orlin, Nilopal Chakravarty

Sensitivity Analysis for Shortest Path Problems and Maximum Capacity Path Problems in Undirected Graphs

Ramkumar Ramaswamy
Infosys Technologies Limited
3rd Cross, Electronics City
Bangalore 561 229 India
ramkumarr@infy.com

James B. Orlin
MIT E40-147
Cambridge, MA 02139
jorlin@mit.edu

Nilopal Chakravarti
Integrated Decision Systems Consultancy
1 Jalan Kilang Timor
Singapore 159303
nilotpal@idsc.com.sg

March 2001

Revised, December 2003

Abstract. This paper addresses sensitivity analysis questions concerning the shortest path problem and the maximum capacity path problem in an undirected network. For both problems, we determine the maximum and minimum weights that each edge can have so that a given path remains optimal. For both problems, we show how to determine these maximum and minimum values for all edges in O(m + K log K) time, where m is the number of edges in the network, and K is the number of edges on the given optimal path.

Key words: sensitivity analysis, shortest path problem, bottleneck shortest path, maximum capacity path problem.

1. Introduction.

Let $G = (N, E)$ be an undirected graph with n nodes and m edges, and designated source node s and sink node t. This paper addresses sensitivity analysis questions concerning the shortest s-t path (SP) problem in G and the maximum capacity s-t path (MCP) problem in G. For each $(i, j) \in E$, let c_{ij} denote either the length of (i, j) or the capacity of (i, j) depending on whether we are solving the shortest path or the maximum capacity path problem.

Suppose that P^* is a shortest s-t path in G. For each edge $e \in E$, the *lower SP tolerance* of edge e is the minimum non-negative value that the length of edge e can take (with all other lengths staying fixed) so that P^* remains an optimal path. Similarly, the *upper SP tolerance* of edge e is the maximum value in $\mathbb{R} \cup \infty$ that the length of edge e can take so that P^* remains an optimal path. We show that the problem of finding all upper and lower tolerances of edges in E is computationally equivalent to the *Minimum Cost Interval Problem* which is as follows. For each $i = 1$ to r, let $[a_i, b_i]$ denote an interval with endpoints in $\{1, ..., q\}$, and let d_i be the associated cost. For each $k = 1$ to q, identify a minimum cost interval $[a_i, b_i]$ containing k. We provide an $O(r + q \log q)$ algorithm for the Minimum Cost Interval Problem, and provide an $O(m + |P^*| \log |P^*|)$ algorithm for finding all upper and lower tolerances of edges in E.

Related sensitivity analysis problems for tree solutions have been considered initially by Shier and Witzgall (1980). Examples of subsequent research in the same direction include an $O(m\alpha(m,n))$[1] algorithm for sensitivity analysis of the minimum cost spanning tree problem by Tarjan (1984) and an $O(m)$ algorithm for sensitivity analysis of minimum spanning trees and shortest path trees in planar graphs by Booth and Westbrook (1994). We refer the reader to Gal (1995), Greenberg (1998) and to Gal and Greenberg (1997) for extensive references to a variety of sensitivity analysis problems in combinatorial optimization.

We further consider the sensitivity analysis problem for the maximum capacity path problem. For a network with edge capacities, the capacity of a path is the minimum capacity of an edge on the path. Let Q^* be the maximum capacity s-t path in G with respect to capacity vector c. For each edge $e \in E$, the *lower* (resp., *upper*) *MCP tolerance* of the edge e is the minimum (resp., maximum) value in $\mathbb{R} \cup \{-\infty,\infty\}$. that the capacity of edge e can take so that Q^* remains a maximum capacity path. We show that the problem of finding all upper and lower tolerances of edges in E can be solved in $O(m + |Q^*| \log |Q^*|)$ time. Moreover, the problem of finding all tolerances is nearly computationally equivalent to the Minimum Cost Interval Problem.

The *most vital SP edge problem* is the problem of finding an edge e whose deletion from G increases the length of the shortest path from s to t as much as possible. The most vital MCP edge problem is the problem of finding an edge e whose deletion from G decreases the maximum capacity of a path from s to t as much as possible. The most vital SP edge is the edge with the largest upper tolerance, and the most vital MCP edge is the edge with the lowest MCP tolerance. Algorithms for the most vital edge problems on shortest paths include Bar-Noy et al (1995), Malik et al (1989), Venema et al (1996). Algorithms for finding the most vital edges in minimum cost spanning trees have been developed by Hsu et al (1991), Hsu et al. (1992), Iwano and Kato (1993), and Banerjee and Saxena (1997). The most vital edge problem for minimum cost spanning trees is not directly related to the problem of identifying edge tolerances, although both are fundamental questions of sensitivity analysis.

[1] $\alpha(m, n)$ is the inverse Ackermann function, and is VERY slowly growing in m and n.

A different approach for determining upper SP-tolerances in O($m + n \log n$) time for edges in a shortest path P^* was independently developed by Hershberger and Suri (2001). They incorrectly claimed to have solved the sensitivity analysis problem for directed graphs, and pointed out their error in Hershberger and Suri (2002).

This paper extends previous results in the unpublished work of Ramaswamy (1994), who gave O(n^2) algorithms for the upper and lower tolerance problems considered here. In Sections 2, 3, and 4, we consider sensitivity analysis for the shortest path problem. In Section 5, we address the Minimum Cost Interval problem; we show its equivalence to the sensitivity analysis problem for SP and evaluate its computational complexity. In Sections 6, and 7, we consider the maximum capacity path problem. We offer a summary and conclusions in Section 8.

2. Upper and Lower Tolerances for Combinatorial Optimization Problems.

In this section, we give procedures for calculating the upper and lower tolerances of variables for combinatorial optimization problems with linear costs as well as for problems with bottleneck (or maximin) costs. We first consider linear objectives.

Combinatorial Optimization Problems with Linear Objectives.

Consider a combinatorial optimization problem **X** where costs are assumed to be non-negative. Let P(c) denote the instance of **X** min (cx: $x \in F$), where $F \subseteq \{0, 1\}^n$, and where $c \geq 0$. We assume that **X** is *closed under substitution of linear objectives*. That is, the instance P(d) = min (dx: $x \in F$) is also an instance of **X** so long as c and d have the same number of components and $d \geq 0$. Let $z(c)$ denote the optimal objective value for P(c).

Suppose that x^* is an optimal solution to P(c).

We are interested in how much the cost coefficients of c can change with x^* remaining optimal. To this end, for each index i and for each $k \in \mathbb{R}$, we let $c^{i,k}$ denote the vector derived from c as follows:

$$c_j^{i,k} = \begin{cases} k, & \text{if } j = i; \\ c_j, & \text{if } j \neq i. \end{cases} .$$

For each component i, we define the *lower tolerance* α_i := min { k: $k \geq 0$ and x^* is optimal for P($c^{i,k}$)} to be the least non-negative value that the cost of component i can take so that x^* remains optimal. Similarly, we define the *upper tolerance* β_i to be max { k: x^* is optimal for P($c^{i,k}$)}. If x^* is optimal for P($c^{i,k}$) for all finite values of k, then $\beta_i := \infty$.

The next theorem characterizes the upper and lower tolerances of x^* in terms of optimal solution values for related problems. The theorem is easily established, and we omit the proof.

Theorem 1. *Let x^* be optimal for P(c), and let $i \in \{1, 2, ..., n\}$. If $x_i^* = 1$, then $\alpha_i = 0$, and $\beta_i = z(c^{i,\infty}) - c^{i,0}x^*$. If $x_i^* = 0$, then $\beta_i = \infty$, and $\alpha_i = cx^* - z(c^{i,0})$.* ♦

The Shortest Path Problem

We now return to the shortest path problem on an undirected graph $G = (N, E)$. Let $n = |N|$ and let $m = |E|$. For each edge $e \in E$, c_e denotes the length of edge e. We assume that $c_e \geq 0$ for all edges e. We permit costs to be infinite. If S is a subset of edges, then $c(S) = \sum_{e \in S} c_e$. We let $G^{e,k}$ denote the graph $G = (N, E)$ in which the cost vector c is replaced by $c^{e,k}$.

Let P^* denote a shortest path from node s to node t in G, and let $d(s, t) = c(P^*)$. In general, let $d^{e,k}(s,t)$ be the length of the shortest s-t path in $G^{e,k}$. We note that if $e \in P^*$, then $c^{e,0}(P^*) = c(P^*) - c_e$.

If we interpret Theorem 1 in terms of tolerances for the shortest path problem, we get the following corollary.

Corollary 1. *Let P^* be a shortest path in G = (N, E). If $e \in P^*$, then $\alpha_e = 0$, and $\beta_e = d^{e,\infty}(s,t) - c(P^*) + c_e$. If $e \notin P^*$, then $\beta_e = \infty$ and $\alpha_e = c(P^*) - d^{e,0}(s,t)$.*

By Corollary 1, the lower tolerance of an edge $e \in P^*$ is 0, and the upper tolerance can be calculated by solving a single shortest path problem. Also by Corollary 1, the upper tolerance of an edge $e \notin P^*$ is infinity, and the lower tolerance can be calculated by solving a single shortest path problem. Thus, we could find all of the edge tolerances by solving at most m shortest path problems.

In the next sections, we will show how to find all edge tolerances in $O(m + |P^*| \log |P^*|)$ time, which is nearly a factor of m improvement in running time. We will also show that the problem of computing all tolerances reduces to the Minimum Cost Interval Problem, as defined in Section 1.

Combinatorial Optimization Problems with Bottleneck Objectives.

For a given n-vector c, and a 0-1 n-vector x, let $c_{\min}(x) = \min\{c_i : x_i = 1\}$. In this subsection, the vector c denotes a vector of capacities, and $c_{\min}(x)$ is the *capacity* of solution x.

Consider the following instance of a *bottleneck combinatorial optimization* problem **Y:** max $(c_{\min}(x)$: $x \in F)$, where $F \subseteq \{0, 1\}^n$, and where $c_j \in \mathbb{R} \cup \{-\infty, \infty\}$ for each $j = 1$ to n. We assume **Y** is closed under substitution of linear objectives, and we let $\mathsf{B}(c)$ denote the instance max $(c_{\min}(x)$: $x \in F)$. (Here, we permit all linear objectives, with positive and negative coefficients.) Let $v(c)$ denote the optimal objective value for $\mathsf{B}(c)$. Let x^* be an optimal solution to $\mathsf{B}(c)$. Analogously to before, we let the lower tolerance α_i and the upper tolerance β_i be defined as follows:

$\alpha_i = \min \{ k : x^*$ is optimal for $\mathsf{B}(c^{i,k})\}$.

$\beta_i = \max \{ k : x^*$ is optimal for $\mathsf{B}(c^{i,k})\}$.

If x^* is optimal for $\mathsf{B}(c^{i,k})$ for all negative values of k, then $\alpha_i = -\infty$. If x^* is optimal for $\mathsf{B}(c^{i,k})$ for all positive values of k, then $\beta_i = \infty$.

The following theorem is analogous to Theorem 1. It can be proved in a straightforward manner, and we omit the proof.

Theorem 2. *Let x^* be optimal for B(c), and let $i \in \{1, 2, ..., n\}$. If $x_i^* = 1$, then the following are true:*

1. $\alpha_i = v(c^{i,-\infty})$.
2. *If x^* is optimal for $\mathrm{B}(c^{i,\infty})$, then $\beta_i = \infty$.*
3. *If x^* is not optimal for $\mathrm{B}(c^{i,\infty})$, then $\beta_i = c_{\min}^{i,\infty}(x^*)$*

If $x_i^* = 0$, then the following are true:

4. $\alpha_i = -\infty$.
5. *If $v(c^{i,\infty}) = c_{\min}(x^*)$, then $\beta_i = \infty$.*
6. *If $v(c^{i,\infty}) > c_{\min}(x^*)$, then $\beta_i = c_{\min}(x^*)$.*

The Maximum Capacity Path Problem

We now return to the Maximum Capacity Path Problem. We let P^* denote a maximum capacity path from *s* to *t* in *G*.

The following corollary is a translation of Theorem 2 to the MCP problem.

Corollary 2. *Let P^* be a maximum capacity path for G = (N, E). If $e \in P^*$, then the following are true:*

1. $\alpha_e = v(G^{e,-\infty})$ *and*
2. *If P^* is a maximum capacity path for $G^{e,\infty}$, then $\beta_e = \infty$;*
3. *If P^* is not a maximum capacity path for $G^{e,\infty}$, then β_e is the minimum capacity of an edge of $P^* \backslash e$.*

If $e \notin P^*$, then the following are true:

4. $\alpha_e = -\infty$;
5. *If P^* is a maximum capacity path for $G^{e,\infty}$, then $\beta_e = \infty$;*
6. *If P^* is not a maximum capacity path for $G^{e,\infty}$, then, $\beta_e = c_{\min}(P^*)$.* ♦

3. Lower S-P tolerances.

In this section, we show how to determine lower S-P tolerances for all edges in O(*m*) time. We first give pseudo-code for the algorithm, and subsequently establish its correctness and running time.

Algorithm 1. Compute Lower S-P Tolerances.

```
begin
    for each $e \in P^*$, $\alpha_e$ := 0;
    determine the shortest path length $d(s, i)$ in $G$ from $s$ to $i$ for all $i \in N$;
    determine the shortest path length $d(j, t)$ in $G$ from $j$ to $t$ for all $j \in N$;
    for each edge $e = (i, j) \notin P^*$, $\alpha_e = c(P^*) - \min(c(P^*), d(s, i) + d(j, t), d(s, j) + d(i, t))$;
end
```

Before establishing the correctness of Algorithm 1, we introduce some more notation. Let T^s denote a tree of shortest paths from node s to all other nodes, and let T^t be a tree of shortest paths to node t. Let $P(s, i)$ denote the path in T^s from node s to node i. Let $P(j, t)$ denote the path in T^t from node j to node t. For any edge $(i, j) \in E$, let $W(i,j) = P(s,i), (i,j), P(j,t)$, which is the s-t walk obtained by concatenating $P(s,i)$, (i, j), and $P(j,t)$.

Theorem 3. *Algorithm 1 correctly computes the lower S-P tolerances for an undirected network G = (N, E), and can be implemented to run in O(m) time.*

Proof. By Corollary 1, it suffices to show that for edges $e \notin P^*$,

$$d^{e,0}(s,t) = \min\,(c(P^*), d(s, i) + d(j, t), d(s, j) + d(i, t)),$$

If there is a shortest s-t path in $G^{e,0}$ that does not contain edge e, then $d^{e,0}(s,t) = c(P^*)$. If there is a shortest s-t path in $G^{e,0}$ that does contain edge e, then $d^{e,0}(s,t) = \min\,(d(s, i) + d(j, t), d(s, j) + d(i, t))$.

Computing $d^{e,0}(s,t)$ for all $e = (i, j) \notin P^*$ requires only that we compute $d(s,i)$ for all nodes i and that we compute $d(j,t)$ for all nodes j. This requires only two shortest path computations on undirected networks, which takes $O(m)$ time using the algorithm by Thorup (1997). ♦

4. Upper tolerances and the Minimum Cost Interval Problem

In this section, we give an algorithm for computing upper *S-P* tolerances in G with respect to a minimum length path P^*. We first give pseudo-code for solving the upper tolerance problem. We later show that the bottleneck step is equivalent to the Minimum Cost Interval Problem.

Algorithm 2. Compute Upper S-P Tolerances.

```
begin
    for each $e \notin P^*$, $\beta_e := \infty$;
    choose $\varepsilon > 0$ so that for any subsets S and S' of edges, if $c(S) < c(S')$ then $c(S) + \varepsilon < c(S')$;[2]
    for each $e \in P^*$, let $c'_e = c_e + \varepsilon/n^2$;
    for each $e \notin P^*$, let $c'_e = c_e + \varepsilon/n$;
    let $T^s$ be a tree of shortest paths from node $s$ to all other nodes with respect to
        costs $c'$;
    let $T^t$ be a tree of shortest paths to node t from all other nodes with respect to
        costs $c'$;
    for each $i \in N$, let $P(s, i)$ denote the path in $T^s$ from $s$ to $i$;
    for each $j \in N$, let $P(j, t)$ denote the path in $T^t$ from $j$ to $t$;
    for each edge $e \in P^*$, $\beta_e := c_e - c(P^*) + \min\,(c(W(i, j)): (i, j) \in E \backslash P^*$, and $e \notin W(i, j))$;
end
```

We will soon establish the correctness of Algorithm 2. However, we first make a brief comment on the running time. It might appear that there are four potential bottleneck operations. First of all, there is the calculation of ε in the second line. Second, there is the calculation of the shortest path trees. Third,

[2] The purpose of ε is to perturb the problem so that shortest path P^* is unique and such that $P^* \subseteq T^s \cap T^t$

there is the definition of $W(i, j)$ for all $(i,j) \in E\backslash P^*$. Fourth, there is evaluating for all e in P^* the following: $\min\{c(W(i,j)) : (i,j) \in A, \textit{ and } e \notin W(i,j)\}$.

As for the calculation of ε, this can be accomplished by choosing any positive $\varepsilon < 1$ if all data are integral. If data is permitted to be irrational, then one can represent the perturbation of costs implicitly as a second component of the costs, and calculate the shortest path trees using lexicography. The calculation of the shortest path trees using lexicography is O(m) using the technique of Thorup (1997).

We also do not directly determine $W(i, j)$. Instead, we show how to determine for all e in P^* $\min\{c(W(i,j)) : (i,j) \in E \setminus P^*, \textit{ and } e \notin W(i,j)\}$ by reducing it to the minimum cost interval problem. This is accomplished in Theorem 4 below. Thus, it is the calculation of $\min\{c(W(i,j)) : (i,j) \in E \setminus P^*, \textit{ and } e \notin W(i,j)\}$ that is the bottleneck step of Algorithm 2.

We begin proving the correctness of Algorithm 2 with the following lemma concerning the spanning trees T^s and T^t.

Lemma 1. *Suppose that T^s, T^t, $P(s, j)$, and $P(j, t)$ are defined as in Algorithm 2. Then for each edge $e \in P^*$, $e \notin P(s, j) \cap P(j, t)$.*

Proof. We first note that for any two paths P and P' in G, if $c'(P) \le c'(P')$, then $c(P) \le c(P')$ by our choice of ε. So T^s and T^t are shortest path trees with respect to c. Moreover, $P^* \subseteq T^s \cap T^t$.

Let $P(s, j) = P(s, i_1), Q_1$, where i_1 is the last node of P^* on the path $P(s, j)$. Let $P(j, t) = Q_2, P(i_2, t)$, where i_2 is the first node of $P(j, t)$ on path P^*. Suppose $e \in P(s, i_1) \cap P(i_2, t)$. In that case, let $P'(s, j) = P(s, i_2), Rev(Q_2)$, where $Rev(Q_2)$ is obtained from Q_2 by visiting the edges in opposite order. Similarly, let $P'(j, t) = Rev(Q_1), P(i_1, t)$. Then $P'(s, j)$ is an s-j path, $P'(j, t)$ is a j-t path, and $c'(P'(s, j)) + c'(P'(j, t)) < c'(P(s, j)) + c'(P(j, t))$, contradicting that $P(s, j)$ and $P(j, t)$ are both shortest paths with respect to c'. ♦

Theorem 4. *Algorithm 2 correctly computes the upper S-P tolerances for an undirected graph $G = (N, E)$.*

Proof. By Corollary 1, for each $e \notin P^*$, $\beta_e = \infty$. So, we henceforth consider the case that $e \in P^*$.

Also, by Corollary 1, it suffices to show the following:

$$d^{e,\infty}(s,t) = \min\ \{c(W(i,j)) : (i,j) \in E \setminus P^*, \textit{ and } e \notin W(i,j)\}, \tag{1}$$

If there is no s-t path in $G\backslash e$, then $d^{e,\infty}(s,t) = \infty$, and the minimization in the right hand side of (1) is over the null set, and so (1) is valid.

We now consider the case that P^e is some shortest s-t path in $G\backslash e$ with respect to costs c. Since the length of the shortest s-t path is also the length of the shortest s-t walk, it follows that

$$d^{e,\infty}(s,t) \le \min\ \{c(W(i,j)) : (i,j) \in E \setminus P^*, \textit{ and } e \notin W(i,j)\}$$

We next prove the reverse inequality.

Let $S = \{i \in N : e \notin P(s,i)\}$. Let i denote the last node on P^e that is also in S, and let j be the subsequent node on P^e. We know that nodes i and j exist because P^e is an s-t path, $s \in S$, and $t \notin S$. By assumption, $i \in S$ and $j \notin S$, and so $e \in P(s, j)$. By Lemma 1, $e \notin P(j, t)$. Because $e \notin P^e$, it follows that $(i, j) \neq e$. So, $e \notin W(i, j)$. Path P^e is a concatenation of a path from node s to node i, edge (i, j), and a path from node j to node t. Therefore, $c^{e,\infty}(P^e) = c(P^e) \geq d(s,i) + c_{ij} + d(j,t) = c(W(i,j))$, completing the proof. ♦

Theorem 4 does not generalize to directed graphs. We give a counterexample to the generalization of Theorem 4 to directed graphs in Figure 1. It is the failure of Theorem 4 to generalize to directed graphs that, in our opinion, makes it more difficult to compute edge tolerances in directed graphs.

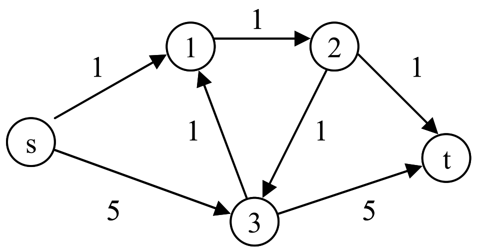

Figure 1. A counterexample to the directed version of Theorem 4, where $e = (1, 2)$.

Also, Theorem 4 would not be true if edge costs could be 0 and if we did not require paths to be shortest paths with respect to the perturbed costs c'. Consider an undirected version of Figure 1 where each edge has a cost of 0. If we do not require paths to have the fewest number of edge, we can let each of $P(s, 1)$, $P(s, 2)$, $P(s, 3)$, $P(1, t)$, $P(2, t)$, and $P(3, t)$ contain edge $e = (1, 2)$. In this case, the minimization in Theorem 1 would be over the empty set.

We now use Theorem 4 to transform the problem of computing upper tolerances for edges in P^* to the Minimum Cost Interval Problem. We accomplish this in Algorithm 3.

Algorithm 3. Transform SP Upper Tolerance Problem to Min Cost Interval Problem

```
begin
    relabel edges so that $P^* = e_1, e_2, \ldots, e_K$;
    let $T^s$, $T^t$, $P(s, j)$, $P(j, t)$ be defined as in Algorithm 2;
    $d(s, j) := c(P(s, j))$ for $j = 1$ to $n$;
    $d(j, t) := c(P(j, t))$ for $j = 1$ to $n$;
    for each $i \in N$ do
        begin
            if $P(s, i) \cap P^* = \emptyset$, then $a(i) := 1$; else choose a(i) so that $e_{a(i)-1}$ is the last edge of $P^*$ that is
                also on $P(s, i)$;
            if $P(i, t) \cap P^* = \emptyset$, then $b(i) := K$; else choose $b(i)$ so that $e_{b(i)+1}$ is the first edge of $P^*$ that is
                also on $P(i, t)$;
        end
    for each edge $(i, j) \notin P^*$ with $a(i) \leq b(j)$, create an interval $[a(i), b(j)]$ with cost $c(W(i, j)) =$
        $d(s, i) + c_{ij} + d(j, t)$;
end
```

Given the shortest path trees from Algorithm 2, we can easily compute $a(i)$ in Algorithm 3 for all nodes i in $O(n)$ time by scanning nodes of T^s in depth first search order. If j is the predecessor of i in T^s, and if $j \in P^*$, then $a(i)$ is the index of the edge following j on P^*. If $j \notin P^*$, then $a(i) = a(j)$. Similarly, we can compute $b(i)$ for all nodes i in $O(n)$ time by scanning nodes of T^t in depth first search order.

Because $P^* \subseteq T^s \cap T^t$, it follows that all edges preceding $e_{a(i)-1}$ in $P(s, i)$ and all edges succeeding $e_{b(j)+1}$ on $P(j, t)$ are also on P^*.

Theorem 5. For each edge $(i, j) \in P^*$ with $a(i) \le b(j)$, create an interval $[a(i), b(j)]$ with cost $c(W(i, j)) = d(s, i) + c_{ij} + d(j, t)$ as in Algorithm 3. Then the minimum cost of an interval covering k is

$$d^{e_k,\infty}(s,t) = min\ \{c(W(i,j)) : (i,j) \in E \setminus P^*, and\ e \notin W(i,j)\}.$$

Proof. By Theorem 3 and Algorithm 2, $d^{e_k,\infty}(s,t) = \min\{c(W(i,j)) : (i,j) \in E \setminus P^*\ and\ e_k \notin W(i,j)\}$. We will complete the proof by showing that for any edge e_k, we have $e_k \notin W(i, j)$ if and only if $k \in [a(i), b(j)]$. If $e_k \in W(i, j)$, then $e_k \in P(s, i)$ or $e_k \in P(j, t)$ or both. Hence $k < a(i)$ or $k > b(j)$ or both, and thus $k \notin [a(i), b(j)]$. Conversely, if $e_k \notin W(i, j)$, then $e_k \notin P(s, i)$ and $e_k \notin P(j, t)$. Hence $k > a(i)$-1 and $k < b(j) + 1$, and thus $k \in [a(i), b(j)]$. ♦

Let $TOL(K, m)$ denote the time to find the upper and lower tolerances for a shortest path problem on an undirected graph with m edges and with K edges in the given shortest s-t path. Let $INT(q, r)$ denote the time to solve the Minimum Cost Interval Problem over r intervals with endpoints in $\{1, ..., q\}$. The next theorem states the computational equivalence of the Minimum Cost Interval Problem and the problem of finding upper and lower tolerances for a shortest path problem on an undirected graph.

Theorem 6. Suppose that $m \ge n$. Then $TOL(K, m) = O(INT(K, m))$, and $INT(q, r) = O(TOL(q, r))$.

Proof. Theorems 3 and 4 show that computing the tolerances of edges not on P^* requires the computation of two shortest path problems, and this takes $O(m)$ time (Thorup (1997)). Theorems 1 and 5 show that the additional time to compute tolerances of edges on P^* is $O(INT(K, m))$. Therefore $TOL(K, m) = O(INT(K, m))$.

Now suppose that $r \ge q$, and consider the Minimum Cost Interval Problem in which the intervals are $[a(i), b(i)]$, with cost $d(i)$. Without loss of generality, assume that there is at most one interval with endpoints i and j for all i and j. If there is an interval $[j, j]$, let z_j denote its cost. Otherwise, let $z_j = \infty$. We create a tolerance problem as follows. We create a graph $G = (N, E)$, with $q + 1$ nodes, where the source node is node 1, and the sink node is node q+1. Let P^* be the path 1, 2, ..., q+1, and suppose that each edge of P^* has a cost of 0. Let e_i denote the edge $(i, i+1)$. For each i = 1 to r with $a(i) \ne b(i)$, there is an edge $e_{r+i} = (a(i), b(i)+1)$ with a cost of $d(i)$. If an interval i is such that $a(i) = b(i)$, we call it a *zero length interval.* We do not create edges in G corresponding to zero-length intervals.

Let β_j denote the upper tolerance of edge e_j in G. The minimum cost of an interval covering j is $\min(d(i) : j \in [a(i), b(i)])$, which we will show is equal to min $\{\beta_j, z_j\}$. If $j \in [a(i), b(i)]$ and $a(i) \ne b(i)$ then $e_j \notin P(1, a(i))$, and $e_j \notin P(b(i)+1, q+1)$, and so $e_j \notin W(a(i), b(i)+1)$. Similarly, if $j \notin [a(i), b(i)]$, then either $j < a(i)$, and $e_j \in P(1, a(i))$ or else $j > b(i)$, and $e_j \in P(b(i)+1, q+1)$. So, j is contained in a nonzero length interval $[a(i), b(i)]$ if and only if $e_j \notin W(a(i), b(i)+1)$. By Theorem 5, the cost of a minimum cost nonzero length interval covering j is the upper tolerance of e_j in G. This establishes that $INT(q, r) = O(TOL(q+1, q+r))$. However, since the tolerance problem can be solved in polynomial time and since $q \leq r$ by assumption, it follows that $TOL(q+1, q+r) = O(TOL(q, r))$, and accordingly $INT(q, r) = O(TOL(q, r))$. ♦

5. Solving the Minimum Cost Interval Problem.

Here we provide an $O(r + q \log q)$ algorithm for solving the Minimum Cost Interval Problem. Shigeno and Uno (2002) independently solved the Minimum Cost Interval Problem with the same running time.

Let F denote a set of r intervals. Let c_{ij} denote the cost of the interval $[i, j]$ for each $[i, j] \in F$. The minimum cost of an interval covering integer l is $g(l) = \min \{ c_{ij} : i \leq l \leq j\}$. Let $f_k(j) = \min \{ c_{ij} : 1 \leq i \leq k \}$. Then $g(k) = \min \{ f_k(j) : k \leq j \leq q\}$. Moreover, for each k and for each j, $f_k(j) = \min \{f_{k-1}(j), c_{kj}\}$.

The following algorithm computes the minimum cost of each interval.

Algorithm 4. The Minimum Cost Interval Algorithm

```
begin
    for j =1 to q,  f(j) = min {c_{1j} ,∞};
    g(1) := min{ f(j): j = 1 to q};
    for k = 2 to q do
    begin
        for j = k to q  do f(j) = min {f(j), c_{kj}};
        g(k) = min{ f(j): j = k to q};
    end
end
```

The first "for loop" computes $f_1(j)$ for each j. When the index is k, the second "for loop" computes $f_k(j)$ for each j. The correctness of the algorithm follows from the fact that minimum cost of an interval covering integer k is $\min \{f_k(j): j = k \text{ to } q\}$.

The algorithm can be implemented in $O(r + q \log q)$ using Fredman and Tarjan's (1984) Fibonacci Heap. At each iteration, we store the values of $f(\cdot)$ in the heap. To initialize the heap takes $O(q)$ steps. To update the heap in the line "$f(j) = \min \{f(j), c_{kj}\}$" takes $O(r)$ steps in total since each interval causes f to either stay the same or decrease, and the decrease operation takes O(1) steps. Finally, there are q operations of finding the min and q options of deleting $f(k)$ at the end of iteration k. Each of these operations takes $O(\log q)$ time using Fibonacci Heaps.

We state our conclusions in the next theorem.

Theorem 7. A Fibonacci Heap implementation of Algorithm 4 solves the Minimum Cost Interval Problem in $O(r + q \log q)$ steps. ♦

6. Upper Tolerances for the Maximum Capacity Path Problem

In this section, we return to the Maximum Capacity Path Problem on a network $G = (N, E)$, where c_e denotes the capacity of edge $e \in E$.

The MCP problem arises in several domains. For example, one method for implementing the augmenting path algorithm for the maximum flow problem is to send flow along a path with maximum capacity. This was first analyzed by Edmonds and Karp (1972). Additional details can be found in Ahuja *et. al.,* (1993). The maximum augmenting path problem is mathematically equivalent to the bottleneck shortest path problem. An example of the bottleneck shortest path problem is the problem of finding a path from s to t such that the minimum reliability of an edge is maximized.

Let P^* denote a maximum capacity path from s to t in G. We next use the results of Corollary 2 to provide algorithms for computing the upper tolerances of all edges in O(m) time.

```
Algorithm 5.  Compute Upper MCP tolerances for edges not in P*
begin
    $H := \{a \in E: c_a > c_{\min}(P^*)\}$;
    let $G_H$ denote the graph $(N, H)$;
    let $S$ be the nodes in the same connected component as $s$ in $G_H$;
    let $T$ be the nodes in the same connected component as $t$ in $G_H$;
    for each edge $e = (i, j) \in E \backslash P^*$ do
        if $i \in S$ and $j \in T$ or if $i \in T$ and $j \in S$, then $\beta_e = c_{\min}(P^*)$;  else $\beta_e = \infty$;
end
```

```
Algorithm 6.  Compute Upper MCP tolerances for edges in P*.
begin
    for each $e \in P^*$ such that $c_e \neq c_{\min}(P^*)$, $\beta_e = \infty$;
    let $e'$ be any minimum capacity edge in $P^*$;
    let $H := \{a \in E: c_a > c_{\min}(P^* \backslash e')\}$;
    let $G_H$ denote the graph $(N, H)$;
    let $S$ be the nodes in the same connected component as $s$ in $G_H$;
    let $T$ be the nodes in the same connected component as $t$ in $G_H$;
    for each $e = (i, j) \in P^*$ such that $c_e = c_{\min}(P^*)$ do;
        begin
            if $i \in S$ and $j \in T$ or if $i \in T$ and $j \in S$, then $\beta_e = c_{\min}(P^* \backslash e')$;
            else $\beta_e = \infty$;
        end
end
```

Theorem 8. Algorithm 5 correctly computes the upper MCP tolerances for edges not in P^*. Algorithm 6 correctly computes the upper MCP tolerances for edges in P^*. Each algorithms can be implemented to run in O(m) time.

Proof. For each edge $e = (i, j) \notin P^*$, Algorithm 5 determines whether there is an s-t path in $G^{e,\infty}$ whose capacity exceeds $c_{\min}(P^*)$, and sets β_e correctly using the results of Corollary 2.

For each edge $e = (i, j) \in P^*$ with $c_e > c_{\min}(P^*)$, Algorithm 6 sets $\beta_e = \infty$, as per Corollary 2. We now consider each edge $e = (i, j) \in P^*$ with $c_e = c_{\min}(P^*)$. The capacity of P^* in $G^{e,\infty}$ is $c_{\min}(P^* \backslash e)$, which is the second smallest capacity of an edge of P^*. Algorithm 6 determines if P^* is a maximum capacity path in $G^{e,\infty}$ by checking whether there is some path with capacity greater than $c_{\min}(P^* \backslash e)$, and then sets sets β_e correctly using the results of Corollary 2.

For Algorithms 5 and 6, determining H and the connected components takes O(m) time, as does computing $c_{\min}(P^* \backslash e)$, completing the proof. ♦

7. Efficient Computation of Lower Tolerances for the MCP Problem

In this section, we compute lower tolerances for edges $e \in P^*$. Our results rely on a close connection between maximum capacity paths and the maximum capacity spanning tree.

The maximum capacity spanning tree (MST) problem for G is to find a spanning tree T for which $\sum_{e \in T} c_e$ is maximum. We will refer to an optimum solution as a *maximum capacity spanning tree*.

The following lemma is easily established, and is well known.

Lemma 2. Let T^* denote a maximum capacity spanning tree of $G = (N, E)$. For any pair of nodes i and j, the path in T^* from i to j is a maximum capacity path from i to j in G. ♦

Henceforth, we let T^* denote some maximum capacity spanning tree of G. We let $b(s, j)$ denote the capacity of the path in T^* from s to j, and we let $b(j, t)$ denote the capacity of the path in T^* from j to t.

The tree T^* can be calculated in linear time via a randomized algorithm as per the technique of (Karger *et. al.,* 1995). The best deterministic algorithm for computing the minimum cost spanning tree is due to Gabow *et. al.,* (1984), and the running time is $O(m\, f(n,m))$, where $f(n,m) = \min(i : \log^{(i)} n \le m/n)$. The values $b(s, j)$ and $b(j, t)$ can be computed for all j in an additional O(n) time. The notation b was selected since the maximum capacity of a path is the capacity of its "bottleneck" edge.

The following algorithm computes lower tolerances in G. We will subsequently prove its correctness and show how its running time reduces to that of finding a maximum capacity spanning tree and solving an instance of the Minimum Cost Interval Problem.

```
Algorithm 7. Compute Lower MCP Tolerances
begin
    for each $e \notin P^*$, $\alpha_e = -\infty$;
    let $T^*$ be a maximum capacity spanning tree;
    for each edge $e \in P^* \backslash T^*$, $\alpha_e = c_{min}(P^*)$;
    let $P(i, j)$ denote the path in $T^*$ from node $i$ to node $j$;
    let $b(i, j) = c_{min}(P(i, j))$ denote the capacity of the path from $i$ to $j$;
    for each $e \in P^* \cap T^*$, $\alpha_e = max\{ (min(b(s, i), c_{ij}, b(j, t)) : (i, j) \neq e$, and $T^* + (i, j) - e$ is a tree $\}$;
end
```

Theorem 9. The algorithm Compute Lower MCP Tolerances correctly computes the lower tolerances of capacities in the network $G = (N, E)$, given a maximum capacity s-t path P^*.

Proof. For each edge $e \notin P^*$, $\alpha_e = -\infty$ by Corollary 2. Also by Corollary 2, for each edge $e \in P^*$, α_e is the maximum capacity of a path in $G^{e,-\infty}$, which is denoted as $v(c^{e,-\infty})$. If $e \in P^*$ and $e \notin T^*$, then by Lemma 2, $P(s, t)$ is a maximum capacity path in G, and thus it is also a maximum capacity path in $G^{e,-\infty}$. It follows that $\alpha_e = c_{min}(P^*)$.

If $e \in P^* \cap T^*$ and if there is no path in $G \backslash e$ from s to t, then Algorithm 7 correctly sets α_e to $-\infty$ as there is no arc $(i, j) \neq e$ for which $T + (i, j) - e$ is a tree.

Finally we consider edges $e \in P^* \cap T^*$ such that there is a path in $G\backslash e$ from s to t. For each edge $(i, j) \notin T^*$, let $W(i, j) = P(s, i), (i, j), P(j, t)$, which is a walk from s to t containing edge (i, j). If $W(i, j)$ contains edge e, then $c_{\min}^{e,-\infty}(W(i,j)) = -\infty$. Otherwise $c_{\min}^{e,-\infty}(W(i,j)) = \min(b(s,i), c_{ij}, b(j,t))$. Since $W(i, j)$ contains a path from s to t whose capacity is at least as great as $c_{\min}(W(i,j))$, it follows that

$$v(c^{e,-\infty}) \geq \min\{b(s, i), c_{ij}, b(j, t)) : W(i, j) \text{ does not contain edge } e\}.$$

Moreover, $W(i, j)$ does not contain edge e precisely when $T^* + (i, j) - e$ is a tree. Therefore,

$$\alpha_e \geq max\{ (min(b(s, i), c_{ij}, b(j, t)) \ : T^* + (i, j) - e \text{ is a tree } \}.$$

We now prove the reverse inequality in the case when $e \in P^* \cap T^*$ and there is a path from s to t in $G\backslash e$. Let P' be a maximum capacity s-t path in $G^{e,-\infty}$. Let S^e denote the nodes in $T^*\backslash e$ in the same component as s. Then $N\backslash S^e$ are the nodes of $T^*\backslash e$ in the same component as t. Then P' contains some edge (i, j) with $i \in S^e$, and $j \notin S^e$. Also, $e \notin P'$. So $T^* + (i, j) - e$ is a tree. Moreover, $c_{\min}^{e,-\infty}(P') = c_{\min}(P') \leq \min(b(s,i), c_{ij}, b(j,t))$, with the latter inequality following from the fact that the path in P' from s to i has capacity at most $b(s, i)$ from Lemma 2, and the path in P' from j to t has capacity at most $b(j, t)$. We conclude that

$$\alpha_e = c_{\min}(P') \leq \ \max\{ (\min(b(s, i), c_{ij}, b(j, t)) \ : T^* + (i, j) - e \text{ is a tree } \},$$

and so the theorem is proved. ♦

We now use Theorem 9 to transform the problem of computing tolerances for edges in P^* to the Maximum Cost Interval Problem. This is equivalent to the Minimum Cost Interval Problem except that we want to determine the maximum cost interval containing k for each $k = 1$ to q. Our transformation is very similar to the one contained in Algorithm 3.

Let T^* be the maximum capacity spanning tree. For each node $i \notin N$, recall that $P(s, i)$ is a maximum capacity path from s to i in T^*, and $P(j, t)$ is a maximum capacity path from j to t in T^*. We assume that the edges of E are ordered so that $P(s, t) \cap P^*$ consists of the edges $e_1, e_2, ..., e_r$ in the order that they appear on the path $P(s, t)$. For each node i, if P^* has no edge in common with $P(s, i)$, then let $a(i) = 1$. Otherwise, let $e_{a(i)-1}$ denote the highest index edge of $P(s, t) \cap P^*$ that is also on $P(s, i)$. If P^* has no edge in common with $P(j, t)$, then let $b(j) = r$. Otherwise, let $e_{b(j)+1}$ denote the least index edge of $P(s, t) \cap P^*$ that is also on $P(j, t)$. We can compute $a(i)$ and $b(i)$ for all $i \in N$ in O(n) time in a similar manner to the way indices are computed for Algorithm 3.

Lemma 3. Consider the Maximum Cost Interval Problem, defined as follows:

For each edge $(i, j) \notin P^*$, with $a(i) \leq b(j)$, there is an interval $[a(i), b(j)]$ with cost $c_{\min}(W(i,j))$. Then the minimum cost of an interval covering k is

$$v(c^{e_k,-\infty}) = \max\ (c_{min}(W(i,j) : e_k \notin W(i,j)\ ).$$

Proof. Essentially the same as the proof of Theorem 4. ♦

We let TOLCAP(K, m) denote the time to find tolerances for the maximum capacity path problem, where K is the number of arcs of the given path. Let MST(n, m) be the time to solve a minimum cost spanning tree problem on n nodes and m edges.[3]

Theorem 10. Suppose that $m \geq n$. Then TOLCAP(K, m) = O(INT(K, m) + MST(n, m)). Moreover, INT(r, q) = O(TOLCAP(r, q)).

Proof. The proof that TOLCAP(K, m) = O(INT(K, m) + MST(n, m)) relies on the fact that one can compute tolerances by solving a maximum cost interval problem and by finding a maximum cost spanning tree. The other details are the same as in the proof of Theorem 6.

Now suppose that $r \geq q$, and consider the Maximum Cost Interval Problem in which the intervals are [$a(i)$, $b(i)$], with cost $d(i)$. The proof relies on the same construction as in the proof of Theorem 6, except that here P^* is a path 1, 2, ..., q + 1, where each edge of the path has a capacity of $M > d_{\max}$. ♦

8. Summary and conclusion

In this paper we have considered sensitivity analysis questions for the shortest s-t path (SP) and maximum capacity s-t path (MCP) problems and presented algorithms for answering these questions that are far superior to successive reoptimization. Table 1 summarizes our contribution.

Table 1. Summary of results

	Problem	**Complexity**
1	Minimum [Maximum] cost interval	INT(r, q) = $O(r + q \log q)$
	Shortest path sensitivity	
2	Lower tolerances of edges	$O(m)$
3	Upper tolerances of edges	O(INT(n, m)) = $O(m + n \log n)$
	Maximum capacity path sensitivity	
4	Lower tolerances of edges	O(MST(n, m) + INT(n, m)) = $O(m + n \log n)$
5	Upper tolerances of edges	$O(m)$

Some open questions include the following. (1) What is the computational complexity for the sensitivity analysis questions addressed in this paper if one permits negative cost edges in the SP, but no negative cost cycle? (2), What is the computational complexity for the sensitivity analysis questions addressed in this paper if considers directed rather than undirected graphs? (Some partial results have recently been obtained by Hershberger et al. (2003)) and (3) Are there superior algorithms for solving the Minimum Cost Interval Problem?

Acknowledgments

[3] The maximum capacity spanning tree is mathematically equivalent to the minimum cost spanning tree problem, and so has the same running time.

This work was supported in part by Office of Naval Research grant ONR N00014-98-1-0317. We thank Dan Stratila and two anonymous referees for their numerous suggestions leading to improvements in exposition.

References

Ahuja R., T. Magnanti and J. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, NJ.

Banerjee, S, and S. Saxena. 1997. Parallel algorithm for finding the most vital edge in weighted graphs. *Journal of Parallel and Distributed Computing* **46**, 101-104.

Bar-Noy, A., S. Khuller, and B. Schieber. 1995. The complexity of finding most vital edges and nodes. Technical Report CS-TR-3539, University of Maryland Institute for Advanced Studies, College Park.

Booth H. and J. Westbrook. 1994. A linear algorithm for analysis of minimum spanning and shortest path trees of planar graphs. *Algorithmica* **11**, 341-352.

Edmonds J. and R. M. Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, **19**, 248-264.

Fredman, M.L., and R.E. Tarjan 1984. Fibonacci heaps and their uses in improved network optimization algorithms, Proceedings of the 25th Annual IEEE Symposium on the Foundation of Computer Science, 338-446. Full paper in Journal of the ACM 34 (1987) 596-615.

Gabow H. N., Z. Galil, and T.H. Spencer. 1984. Efficient implementations of graph algorithms using contraction. *Proceedings of the 25th Annual Symposium on the Foundations of Computer Science* (FOCS '84), 347-357.

Gal. T. 1995. *Sensitivity Analysis, Parametric Programming, and Related Topics: Degeneracy, Multicriteria Decision Making, Redundancy* W.deGruyter, Berlin and New York.

Gal, T. and H. J. Greenberg (eds.). 1997. *Advances in Sensitivity Analysis and Parametric Programming. Volume 6 of International Series in Operations Research and Management Science,* Kluwer Academic Publishers, Boston.

Greenberg H. J. 1998. An annotated bibliography for post-solution analysis in mixed integer programming and combinatorial optimization, in D. Woodruff (Ed.), *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search*, Kluwer Academic Publishers, 97-148.

Hershberger, J. and S. Suri. 2001. Vickrey prices and shortest paths: What is an edge worth? In *Proc. 42nd Annu. IEEE Symp. Found. Comput. Sci.*, pages 252–259, 2001.

Hershberger, J. and S. Suri. 2002. Erratum to "Vickrey prices and shortest paths: What is an edge worth?" Proceedings of the 43 rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'02), 809-810.

Hershberger, J. and S. Suri, and A. Bhosle 2003. On the difficulty of some shortest path problems. Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS 2003), 343-354.

Hsu, L. H., R.H. Jan, Y.C. Lee, C.N. Hung, and M.S. Chern. 1991. Finding the most vital edge with respect to minimum spanning tree in weighted graphs. *Information Processing Letters* **39**, 277-281.

Hsu, L.H., P.F. Wang, and C.T. Wu. 1992. Parallel algorithms for finding the most vital edge with respect to minimum spanning tree. *Parallel Comput.* **18**, 1143-1155.

Karger D. R., P. N. Klein, and R. E. Tarjan. 1995. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, **42**, 321-328.

Malik, K., A.K. Mittal, and S.K. Gupta. 1989. The *k* most vital edges in the shortest path problem. *Operations Research Letters* **8**, 223-227.

Ramaswamy R. 1994. *Sensitivity Analysis in Combinatorial Optimization*. Unpublished Fellow Programme dissertation, Indian Institute of Management Calcutta, India.

Shier D. R. and C. Witzgall. 1980. edge tolerances in shortest path and network flow problems. *Networks* **10**, 277.

Shigeno, M, and T. Uno. 2002. Personal Correspondence.

Tarjan R.E. 1982. Sensitivity Analysis of Minimum Spanning Trees and Shortest Path Trees. *Information Processing Letter*s **14**, 30-33.

Tarjan R. E. 1984. A simple version of Karzanov's blocking flow algorithm. *Operations Research Letters* **2**, 265-268.

Thorup M. 1997. Undirected Single Source Shortest Paths in Linear Time. *Proceedings of the 38th Annual Symposium on the Foundations of Computer Science* (FOCS '97).

Venema, S., H. Shen and F. Suraweera. 1996. NC Algorithms for the Single Most Vital Edge Problem with Respect to Shortest Paths, *Information Processing Letters* **60**, 243-248.