J Comb Optim
DOI 10.1007/s10878-007-9110-4

Finding an anti-risk path between two nodes in undirected graphs

Peng Xiao · Yinfeng Xu · Bing Su

Abstract Given a weighted graph $G = (V, E)$ with a source s and a destination t, a traveler has to go from s to t. However, some of the edges may be blocked at certain times, and the traveler only observes that upon reaching an adjacent site of the blocked edge. Let $\mathcal{P} = \{P_G(s, t)\}$ be the set of all paths from s to t. The risk of a path is defined as the longest travel under the assumption that any edge of the path may be blocked. The paper will propose the Anti-risk Path Problem of finding a path $P_G(s, t)$ in $\mathcal{P}$ such that it has minimum risk. We will show that this problem can be solved in $O(mn + n^2 \log n)$ time suppose that at most one edge may be blocked, where n and m denote the number of vertices and edges in G, respectively.

Keywords Shortest path · Shortest path tree · Most vital real time edge · Anti-risk path

1 Introduction

Let $G = (V, E)$ be an undirected graph (a map) with $|V| = n$ vertices and $|E| = m$ edges, where the set of nodes V corresponds to the set of sites, and the set of edges E corresponds to the set of roads between sites. In addition, each edge $e = (x, y) \in E$

This research is supported by NSF of China under Grants 70525004, 60736027, 70121001 and Postdoctoral Science Foundation of China under Grant 20060401003.

P. Xiao (✉) · B. Su
School of Management, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
e-mail: xiaopeng@mail.xjtu.edu.cn

B. Su
e-mail: subing@mail.xjtu.edu.cn

Y. Xu
State Key Lab for Manufacturing Systems Engineering, Xi'an, Shaanxi 710049, China
e-mail: yfxu@mail.xjtu.edu.cn

is associated with a positive real length $d(x, y)$ interpreted as the time it takes to traverse the road e. Suppose that a traveler has to go from site s to site t. However, some of the roads may be blocked for travel at certain times, and the traveler only observes that upon reaching an adjacent site of the blocked edge. This paper considers the problem of finding a path that results in the optimal travel (according to some predefine measure) from s to t given this uncertainty.

The main difficulty in choosing a good travel route is based on partial information with no knowledge of future blockages before starting travel at site s. Notice that it is simple to choose a travel route when all road blockages are known in advance and the optimal travel route is given by the shortest path from site s to site t which does not make use of the set of known blocked edges. In this paper, we assume that at most one road blockage might occur during a travel.

The problem of computing the replacement path while an edge of the shortest path is eliminated between two given nodes has been studied extensively (see Ball et al. 1989; Bar-Noy et al. 1995; Corley and Sha 1982). Malik et al. (1989) gave an $O(m + n \log n)$ time algorithm to compute all the replacement paths between source and sink in the presence of edge failures along the original shortest path, and Nardelli et al. (2001) further improved to $O(m \cdot \alpha(m, n))$ time, where α is the functional inverse of the Ackermann function defined in (Tarjan 1975). Some variants of the replacement path problem were also investigated. Nardelli et al. (1998) presented the longest detour (LD) problem of computing the detour from x to the destination node with removal of each edge $e = (x, y)$ on the shortest path, and gave an algorithm with the same time and space complexity as that of computing the replacement path. The same bound for undirected graphs is also achieved by Hershberger and Suri (2001), who solved the Vickrey payment problem with the similar algorithm of solving the LD problem, then Nardelli et al. (2001) improved the result to $O(m \cdot \alpha(m, n))$ time bound. Furthermore, Hershberger et al. (2003) showed that the LD problem requires $\Omega(m\sqrt{n})$ time in the worst case whenever $m = n\sqrt{n}$ in directed graphs.

Previous studies on the replacement path problem are based on the assumption that all blocked edges are known in advance, but in fact if we encounter an blockage edge that upon reaching an adjacent site of the blocked edge, we may not return to source s and re-choose the shortest path from s to t. Hence, we should consider that if we observe the blocked edge upon reaching an adjacent site, how to choose a path from s to t before we start in site s in order to cope with sudden failure of an edge. Our problem is revealed its importance in traffic network applications by the following example. Suppose that the traveler choose the shortest path between site s and site t with ten days travel. Assume that in one possible scenario removal of an edge (which results in the largest travel) results in twenty days travel. Hence, following the worst-case criterion, the traveler would spend twenty days, and we call the cost in worst case as the *risk* of that path, equaling to twenty days. On the other hand, assume that the traveler choose another path with twelve days travel (longer than the length of the shortest path) between s and t, however in another possible scenario removal of any edge of that path at most results in fifteen days travel. Following the worst-case criterion, the traveler would spend fifteen days, and the *risk* of that path is fifteen days. As consider the risk of a path, the traveler would prefer the latter path with less risk that guarantees at most fifteen days to travel from s to t.

The *anti-risk path* (ARP) problem is defined as the problem of finding a path between source and sink with the least *risk*. More precisely, we are going to find a path which has the smallest (among all paths from s to t) maximum (among all possible edge failures or no any edge failure) risk.

The paper is organized as follows: Sect. 2 gives some definitions, including the concept of the ARP problem. In Sect. 3, we first present some properties of the ARP problem in a metric space, and then put forward an efficient algorithm. In Sect. 4, we then put forward an efficient algorithm for the ARP problem in a general space. Finally, Sect. 5 concludes the paper.

2 Basic definitions

Given a source node s and a destination node t in G. Let $\mathcal{P} = \{P_G(s,t)\}$ denote the set of all paths from s to t and $P_G(s,t) = \{v_0, e_1, v_1, \ldots, v_{j-1}, e_j, v_j\} \in \mathcal{P}$ denote an path from s to t in G, where $v_0 = s, v_j = t$. Let $d_G(s,t|P)$ denote the length of $P_G(s,t)$. Here, for notation convenience, sometimes we simply write P instead of $P_G(s,t)$. Let $SP_G(s,t)$ be the shortest path from s to t in G, and let $d_G(s,t|SP)$ be its length.

Definition 1 A *detour path* $SP_{G-e_i}(v_{i-1},t)$ at a node $v_{i-1} \in P_G(s,t) = \{s, \ldots, v_{i-1}, v_i, \ldots, t\}$ is the shortest path from v_{i-1} to t which does not make use of edge $e_i = (v_{i-1}, v_i) \in P_G(s,t)$, and the length of the detour path is denoted by $d_{G-e_i}(v_{i-1},t|SP)$, where $G - e_i = (V, E - e_i)$.

Definition 2 A *real time detour path* $P_{G-e_i}(s,t)$ is the path from s to v_{i-1} along path $P_G(s,t)$ joining the path from v_{i-1} to t along *detour path* $SP_{G-e_i}(v_{i-1},t)$. Let $D_{G-e_i}(s,t|P)$ denote the length of $P_{G-e_i}(s,t)$, i.e., $D_{G-e_i}(s,t|P) = d_G(s,v_{i-1}|P) + d_{G-e_i}(v_{i-1},t|SP)$.

Definition 3 The *most vital real time edge* with respect to $P_G(s,t) \in \mathcal{P}$ is the edge $e^* = (x^*, y^*)$ such that $D_{G-e^*}(s,t|P) \geq D_{G-e_i}(s,t|P)$ for every edge $e_i \in P_G(s,t)$.

Definition 4 The *risk* $R(P)$ of a path $P_G(s,t)$ is the longest travel from s to t under the assumption that at most one edge of the path may be blocked.

According to above definition of the *risk* of a path, the longest travel may occur in two cases. One case is that the traveler chooses a longer path in G but no one edge is blocked in the path, and in the other case is that the traveler choose a shorter path in G and the *most vital real time edge* is blocked. Hence, the *risk* of a path $R(P)$ is the maximum between the length of original path and the length of real time detour path with removal of the *most vital real time edge*, i.e., $R(P) = \max\{d_G(s,t|P), D_{G-e^*}(s,t|P)\}$.

The *anti-risk path* (ARP) problem in this paper is to find a path $P_G^*(s,t)$ such that $R(P^*) \leq R(P)$ holds for any $P_G(s,t) \in \mathcal{P}$. First of all, we can give the upper and lower bounds of the risk of the *anti-risk path* $P_G^*(s,t)$.

Lemma 2.1 *For the anti-risk path $P^*_G(s,t)$, its risk is at most the length of real time detour path with removal of the most vital real time edge e^*_{SP} with respect to the shortest path, that is, $d_G(s,t|SP) \le R(P^*) \le D_{G-e^*_{SP}}(s,t|SP)$.*

Proof First, an arbitrary path has the length at least that of the shortest path, so the *anti-risk path* satisfies $R(P^*) \ge d_G(s,t|SP)$. Secondly, we easily have the length of $SP_G(s,t)$ is less than $D_{G-e^*_{SP}}(s,t|SP)$. Hence, by the definition of $R(P)$, we have $R(SP) = D_{G-e^*_{SP}}(s,t|SP)$. And the shortest path is a candidate for P^*. So, $R(P^*) \le D_{G-e^*_{SP}}(s,t|SP)$ holds. □

3 Solving efficiently the ARP problem in a metric space

In this section, we will discuss the properties of the risk of a path in G which is in a metric space, and present a polynomial time algorithm to solve the ARP problem.

3.1 The properties of the risk of a path in a metric space

Lemma 3.1 *For an arbitrary edge $e = (x,y) \in E$ in G, $d(x,y) \le d_G(x,y|p)$ holds, where $d_G(x,y|p)$ is the length of an arbitrary path p from x to y.*

The above lemma holds due to the triangular inequality in a metric space, and we omit the proof. If an edge is blocked, the length of route between the two ends of the edge will increase.

The following lemma shows that the length of the real time detour path with removal of the *most vital real time edge* is longer than that of original path in a metric space, namely, the risk of that path is the length of the real time detour path.

Lemma 3.2 *For an arbitrary path $P_G(s,t) \in \mathcal{P}$ in G, the following equality holds*:

$$R(P) = \max\{d_G(s,t|P),\, D_{G-e^*}(s,t|P)\} = D_{G-e^*}(s,t|P).$$

Proof Let $e^* = (x^*, y^*)$ be the *most vital real time edge* with respect to an arbitrary path $P_G(s,t) = \{s, e_1, v_1, \ldots, v_{j-1}, e_j, t\}$. Suppose, for the sake of contradiction, that $D_{G-e^*}(s,t|P) < d_G(s,t|P)$. According to the definition of the *most vital real time edge*, we have $D_{G-e_i}(s,t|P) \le D_{G-e^*}(s,t|P)$ for every edge $e_i \in P_G(s,t)$. For the last edge $e_j = (v_{j-1}, t)$, the following inequality holds:

$$D_{G-e_j}(s,t|P) \le D_{G-e^*}(s,t|P) < d_G(s,t|P). \tag{1}$$

According to the definition of real time detour path, we have

$$D_{G-e_j}(s,t|P) = d_G(s,v_{j-1}|P) + d_{G-e_j}(v_{j-1},t|SP). \tag{2}$$

On the other hand,

$$d_G(s,t|P) = d_G(s,v_{j-1}|P) + d_G(v_{j-1},t|P). \tag{3}$$

Hence, substitute (2) and (3) into formula (1), we have

$$d_{G-e_j}(v_{j-1}, t|SP) < d_G(v_{j-1}, t|P),$$

which is a contradiction with Lemma 3.1, completing the proof. □

Remark Lemma 3.2 still holds even if $P_G(s,t)$ is the longest simple path from s to t in a metric space. If the last edge of $P_G(s,t)$ is failure when moving to the adjacent site of that edge, the length of detour path is longer than that of the last edge by Lemma 3.1. Hence the detour path and the longest path must be partially overlapped (otherwise, $P_G(s,t)$ is not the longest path). And the length of real time detour path must be longer than that of $P_G(s,t)$. Note that the real time detour path which does not make use of the last edge of the longest path $P_G(s,t)$ is not a simple path. Hence, the length of the real time detour path with removal of the last edge is longer than that of the longest path.

Lemma 3.3 *For an arbitrary path $P_G(s,t) \in \mathcal{P}$, if $e^* = (x^*, y^*)$ is the most vital real time edge of that path, then the shortest path $SP_G(x^*, t)$ will pass through e^*, where x^* is closer to s on $P_G(s,t)$.*

Proof Suppose, for the sake of contradiction, that the shortest path $SP_G(x^*, t)$ does not pass through the edge e^*, then the following inequality holds:

$$d_G(x^*, t|SP) < d_{G(e^*)}(x^*, t|SP),$$

where $d_G(x^*, t|SP)$ is the length of the shortest path $SP_G(x^*, t)$, and $d_{G(e^*)}(x^*, t|SP)$ is the length of the shortest path $SP_{G(e^*)}(x^*, t)$ which has to pass through e^*.

So, the detour path at the node of x^* will be $SP_G(x^*, t)$ after removing $e^* = (x^*, y^*)$, and the following equality holds:

$$d_{G-e^*}(x^*, t|SP) = d_G(x^*, t|SP) < d_{G(e^*)}(x^*, t|SP).$$

On the other hand, we have $d_G(s,t|P) = d_G(s, x^*|P) + d_G(x^*, t|P)$, and $D_{G-e^*}(s,t|P) = d_G(s, x^*|P) + d_{G-e^*}(x^*, t|SP)$, then the following inequality holds:

$$D_{G-e^*}(s,t|P) < d_G(s,t|P),$$

where $d_G(x^*, t|P)$ is not less than $d_{G(e^*)}(x^*, t|SP)$, which is in contradiction with Lemma 3.2, completing the proof. □

Based on the above lemma, we have the following theorem that characterizes the *most vital real time edge* of a path in a metric space.

Theorem 3.1 *Given an arbitrary path $P_G(s,t) \in \mathcal{P}$, if $e^* = (x^*, y^*)$ is the most vital real time edge of the path, then we have $e^* \in SPT_G(t)$, where $SPT_G(t)$ is the shortest path tree $SPT_G(t)$ rooted at t.*

Proof We already know by Lemma 3.3 that the shortest path $SP_G(x^*, t)$ must pass through the *most vital real time edge* e^*. On the other hand, $SPT_G(t)$ is a spanning tree of G rooted at t with unique shortest path from t to v for each $v \in V \backslash \{t\}$. So, we have $e^* \in SPT_G(t)$. □

The following corollary is a special case in Theorem 3.1 for the ARP problem.

Corollary 3.1 *Let $P_G^*(s, t)$ be the anti-risk path, if $e_{P^*}^* = (x^*, y^*)$ is the most vital real time edge with respect to the path, then we have $e_{P^*}^* \in SPT_G(t)$.*

We have the following lemma that characterizes an edge which is not on the shortest path tree rooted at t.

Lemma 3.4 *For an arbitrary path $P_G(s, t) \in \mathcal{P}$, if an edge $e_i = (v_{i-1}, v_i) \in P_G(s, t)$ is not on the shortest path tree $SPT_G(t)$, then the length of real time detour path with removal of e_i is shorter than that of the original path $P_G(s, t)$, that is, $D_{G-e_i}(s, t|P) < d_G(s, t|P)$.*

Proof Since $e_i = (v_{i-1}, v_i)$ is not on the $SPT_G(t)$ rooted at t, the edge e_i is not on the shortest path from v_{i-1} to t, where v_{i-1} is closer to s on $P_G(s, t)$. Thus, when the edge e_i is removed, the detour path $SP_{G-e_i}(v_{i-1}, t)$ is the shortest path $SP_G(v_{i-1}, t)$. Then, we have: $d_{G-e_i}(v_{i-1}, t|SP) = d_G(v_{i-1}, t|SP) < d_G(v_{i-1}, t|P)$. Hence, $D_{G-e_i}(s, t|P) < d_G(s, t|P)$, completing the proof. □

3.2 Description of the algorithm

In this section, we will propose the *Delete Edge on Shortest Path Tree* (*DESPT*) algorithm for the ARP problem in a metric space G and subsequently establish its correctness and running time. By Theorem 3.1, the main idea of algorithm is to sequentially delete the edge which is the *most vital real time edge* with respect to the updated shortest path in the remaining graph, and then to find the minimum risk among the sequentially shortest paths. Hence, the *anti-risk path* is the corresponding updated shortest path with minimum risk. The algorithm *DESPT* is formally described in the following.

Algorithm *DESPT*

Step 1. Produce the two shortest path trees rooted at s and t in G respectively. Record $d_G(s, x|SP)$ for every node $x \in V/\{s\}$ and $d_G(y, t|SP)$ for every node $y \in V/\{t\}$ (see Fredman and Tarjan 1987).

Step 2. Compute $d_{G-e_i}(v_{i-1}, t|SP)$ for every edge $e_i = (v_{i-1}, v_i) \in SPT_G(t)$, where v_i is closer to t (see Nardelli et al. 1998).

Step 3. Set $k = 1, \tau = 0, G_1 = G, R_0 = +\infty, R_1 = 0$ and $DE_1 = \emptyset$.

Step 4. For $e_i = (v_{i-1}, v_i) \in SP_{G_k}(s, t)$, if $e_i \notin SPT_G(t)$, then set $D_{G_k-e_i}(s, t|SP) = 0$; If $e_i \in SPT_G(t)$, then let $D_{G_k-e_i}(s, t|SP) = d_{G_k}(s, v_{i-1}|SP) + d_{G-e_i}(v_{i-1}, t|SP)$.

Step 5. Let $D_{G_k-e^*_{G_k}}(s,t|SP) = \max_{e_i \in SP_{G_k}(s,t)}\{D_{G_k-e_i}(s,t|SP)\}$ and $R(SP_{G_k}(s,t)) = D_{G_k-e^*_{G_k}}(s,t|SP)$.

Step 6. If $R(SP_{G_k}(s,t)) < R_{k-1}$, then let $R_k = R(SP_{G_k}(s,t))$ and $\tau = k$; otherwise, let $R_k = R_{k-1}$.

Step 7. Delete $e^*_{G_k} = (x^*_{G_k}, y^*_{G_k})$, then let $G_{k+1} = G_k - e^*_{G_k}$ and $DE_{k+1} = DE_k + e^*_{G_k}$.

Step 8. Set $k = k+1$. If $k = n$, then turn to Step 11.

Step 9. Compute $SP_{G_k}(s,t)$ and $d_{G_k}(s,t|SP)$ in G_k.

Step 10. If $d_{G_k}(s,t|SP) < R_{k-1}$, then turn to Step 4; Otherwise, turn to Step 11.

Step 11. Output τ and $SP_{G_\tau}(s,t)$.

We will prove $SP_{G_\tau}(s,t)$ is the *anti-risk* path in G by using the algorithm *DESPT* with the following lemma and theorem. The lemma is easily established, and we omit the proof.

Lemma 3.5 *Let G_k be $G_{k-1} - e^*_{G_{k-1}}$ in the algorithm DESPT, then the length of shortest path $SP_{G_k}(s,t)$ in G_k is not longer than that of $SP_{G_{k-1}}(s,t)$ in G_{k-1}.*

Theorem 3.2 *The algorithm DESPT correctly computes the anti-risk path for an undirected graph G which is in a metric space.*

Proof We first prove that the minimum between $D_{G_k-e^*_{G_k}}(s,t|SP)$ and R_{k-1} in Step 6 is the lower bound of the risk of a path in G which passes through some edges in DE_{k+1} in Step 7. Consider the following three steps.

(1) For each edge $e_i = (v_{i-1}, v_i)$ in G_k, since the detour path with removal of e_i is the shortest path from v_{i-1} to t which does not make use of e_i in G, the length of the detour path with removal of e_i is $d_{G-e_i}(v_{i-1}, t|SP)$. Then $D_{G_k-e_i}(s,t|SP)$ in step 4 is the length of real time detour path with removal of e_i with respect to $SP_{G_k}(s,t)$ in G.

(2) We know $SP_{G_k}(s, v_{i-1})$ has the minimum length among those from s to v_{i-1} in G_k. Hence, $D_{G_k-e^*_{G_k}}(s,t|SP)$ is the lower bound of the risk of a path which passes through the edge $e^*_{G_k}$ in G_k.

(3) Consider the edges in DE_{k+1} in Step 7. The risk of a path which passes through $e^*_{G_k}$ and $e^*_{G_{k+1}}$ is not less than $D_{G_k-e^*_{G_k}}(s,t|SP)$ in G_k. And in Step 6, R_k is updated by the minimum between $D_{G_k-e^*_{G_k}}(s,t|SP)$ and R_{k-1}. Hence, R_k is the lower bound of the risk of a path which passes through some edges in DE_{k+1} in Step 7.

We next prove that the algorithm *DESPT* is correctly halted. Consider two cases:

Case 1. If $d_{G_k}(s,t|SP) \geq R_{k-1}$ in G_k, then by Lemma 3.2 and Lemma 3.5, there doesn't exist a path such that its risk is less than R_{k-1} in G_k. Hence, $SP_{G_\tau}(s,t)$ in Step 11 is the *anti-risk path* with the risk R_{k-1}.

Case 2. If $k = n$ in Step 8, then $n-1$ edges on the shortest path tree $SPT_G(t)$ have been deleted and G_k will be an unconnected graph. Then there doesn't exist a path from s to t. Hence, $SP_{G_\tau}(s,t)$ in Step 11 is the *anti-risk path* with the risk R_{k-1}.

This concludes the proof of Theorem 3.2. □

For the algorithm *DESPT*, the following theorem holds.

Theorem 3.3 *For two given nodes s to t in G which is in a metric space, the ARP problem can be solved in $O(mn + n^2 \log n)$ time, where n and m denote the number of vertices and edges in G respectively.*

Proof First, the shortest path trees rooted at s and t respectively can be obtained in $O(m + n \log n)$ time in Step 1 (see Fredman and Tarjan 1987). In Step 2, we can compute the detour path with respect to each edge on the shortest path tree in $O(mn)$ time by applying Nardelli's algorithm (see Nardelli et al. 1998). Step 4 consumes $O(n)$ time to compute the real time detour path for every edge of the shortest path. In Step 5, the comparison costs $O(n)$ time. All the operations in Step 6, Step 7 and Step 8 totally consume $O(1)$ time to update R_k and delete $e^*_{G_k}$ from G_k. In Step 9, the shortest path from s to t in G_k can be obtained in same time complexity as that of Step 1. There are at most $O(n)$ loops from Step 4 to Step 10, and thus we need compute in a total time of $O(mn + n^2 \log n)$. Thus, the above theorem follows. □

4 Solving the ARP problem in a general graph

In Sect. 3, we give an algorithm to solve the ARP problem which is in a metric space. The length of an edge is the minimum length among all paths between two ends of that edge in a metric space. However, Lemma 3.1 is not established in a general space. In some cases, it is possible that there is a shortest path between two ends of an edge whose length is shorter than the edge length. Hence, we will first give the definition of a long-edge in a general space.

Definition 5 A *long-edge* $Le_i = (v_{i-1}, v_i)$ is defined as an edge whose length is strictly longer than that of the shortest path between v_{i-1} and v_i.

We will present some properties of a long-edge, and then give the efficient algorithm to solve the ARP problem.

4.1 The properties of a long-edge

Lemma 4.1 *Let $Le_i = (v_{i-1}, v_i)$ be a long-edge. Then, we have $Le_i \notin SPT(t)$.*

Proof Suppose, for the sake of contradiction, that $Le_i \in SPT(t)$, the shortest distance between v_{i-1} and v_i is the length of the edge by the properties of the shortest path tree. However, there is a path whose length is shorter than the edge length by the definition of long-edge. Hence, it is a contradiction. □

Lemma 4.2 *For an arbitrary path $P_G(s,t) \in \mathcal{P}$, if the following equality holds: $R(P) = \max\{d_G(s,t|P), D_{G-e^*}(s,t|P)\} = d_G(s,t|P)$, then there is at least one long-edge on the path.*

Proof Suppose, for the sake of contradiction, that there is no one long-edge on the path $P_G(s,t)$, we consider the last edge $e_j = (v_{j-1}, t)$ on the path. According to the definition of real time detour path, we have $D_{G-e_j}(s,t|P) = d_G(s, v_{j-1}|P) + d_{G-e_j}(v_{j-1}, t|SP)$. Since $e_j = (v_{j-1}, t)$ is not a long-edge, it is established that $d_{G-e_j}(v_{j-1}, t|SP)$ is larger than $d(v_{j-1}, t)$. Hence, the following inequality holds: $d_G(s,t|P) < D_{G-e_j}(s,t|P) \leq R(P)$, which is in contradiction with the equality: $R(P) = d_G(s,t|P)$. □

Lemma 4.3 *Let $Le_i = (v_{i-1}, v_i)$ be a long-edge of a path $P_G(s,t)$, if $Le_i = (v_{i-1}, v_i)$ does not join with t, then $Le_i = (v_{i-1}, v_i)$ is not the most vital real time edge with respect to $P_G(s,t)$.*

Proof Let $e_j = (v_j, t)$ be the last edge of $P_G(s,t)$, there are two cases.

Case 1. e_j is a long-edge.

We consider a path $P'_{G-Le_i}(v_{i-1}, t)$ from v_{i-1} to t in $G - Le_i$ which is the shortest path from v_{i-1} to v_i joining the path from v_i to v_j along $P_G(s,t)$, and joining the shortest path from v_j to t. Then, the length of $P'_{G-Le_i}(v_{i-1}, t)$ is not shorter than that of detour path $SP_{G-Le_i}(v_{i-1}, t)$. On the other hand, since both Le_i and e_j are the long-edges, the length of real time detour path $P_{G-e_j}(s,t)$ is longer than that of $P_G(s, v_{i-1})$ joining $P'_{G-Le_i}(v_{i-1}, t)$. Hence, the following inequality holds:

$D_{G-Le_i}(s,t|P) < D_{G-e_j}(s,t|P)$.

Therefore, the long-edge Le_i is not the *most vital real time edge* with respect to $P_G(s,t)$.

Case 2. e_j is not a long-edge.

We consider a path $P''_{G-Le_i}(v_{i-1}, t)$ from v_{i-1} to t in $G - Le_i$ which is the shortest path from v_{i-1} to v_i joining the path from v_i to t along $P_G(s,t)$. Then, the length of $P''_{G-Le_i}(v_{i-1}, t)$ is not shorter than that of detour path $SP_{G-Le_i}(v_{i-1}, t)$. However, the length of Le_i is longer than that of the shortest path from v_{i-1} to v_i. Hence, $d_G(s,t|P)$ is larger than $D_{G-Le_i}(s,t|P)$. On the other hand, since the length of e_j is shorter than the shortest path from v_j to t, $D_{G-e_j}(s,t|P)$ is larger than $d_G(s,t|P)$. Then the following inequality holds:

$D_{G-Le_i}(s,t|P) < d_G(s,t|P) < D_{G-e_j}(s,t|P)$.

Therefore, the long-edge Le_i is not the *most vital real time edge* with respect to $P_G(s,t)$.

This concludes the proof of Lemma 4.3. □

From the above proof of Lemma 4.3, the next corollary is established.

Corollary 4.1 *If a long-edge $Le_i = (v_{i-1}, v_i)$ is the most vital real time edge of a path, then Le_i joins with t.*

Lemma 4.4 *If $Le_j = (v_{j-1}, t) \in P_G(s,t)$ is the most vital real time edge with respect to $P_G(s,t)$, then the risk of $P_G(s,t)$ satisfies the following equality*: $R(P) = \max\{d_G(s,t|P), D_{G-Le_j}(s,t|P)\} = d_G(s,t|P)$.

Proof First, by the definition of the *most vital real time edge*, we have $D_{G-e_i}(s, t|P) \leq D_{G-Le_j}(s,t|P)$ for each edge $e_i \in P_G(s,t)$. Secondly, by the definition of

the long-edge, the inequality holds: $d_{G-Le_j}(v_{j-1}, t|SP) < d(v_{j-1}, t)$. Hence, we have $D_{G-Le_j}(s, t|P) < d_G(s, t|P)$ and then $R(P) = d_G(s, t|P)$. The proof is now completed. □

Lemma 4.5 *If $Le_j = (v_{j-1}, t)$ is not the most vital real time edge with respect to a path $P_G(s, t)$, then the most vital real time edge is on the shortest path tree $SPT(t)$.*

Proof Let $e^* = (x^*, y^*)$ be the *most vital real time edge* with respect to $P_G(s, t)$, then the following inequality holds:

$$D_{G-Le_j}(s, t|P) \leq D_{G-e^*}(s, t|P). \tag{4}$$

For the sake of contradiction, suppose that $e^* = (x^*, y^*)$ is not on the shortest path tree $SPT(t)$, the detour path $SP_{G-e^*}(x^*, t)$ with removal of e^* is the shortest path $SP_G(x^*, t)$ from x^* to t in G. On the other hand, there is a path $P'_G(x^*, t)$ which is the path from x^* to v_{j-1} along $P_G(s, t)$ joining the shortest path from v_{j-1} to t. Hence, the real time detour path $P_{G-Le_j}(s, t)$ is the path from s to x^* along $P_G(s, t)$ joining the path from x^* to t along $P'_G(x^*, t)$. Since the length of $SP_{G-e^*}(x^*, t)$ is shorter than that of $P'_G(x^*, t)$, $D_{G-e^*}(s, t|P)$ is less than $D_{G-Le_j}(s, t|P)$, which is in contradiction with (4), completing the proof. □

The following corollary is a special case of Lemma 4.5 for the ARP problem.

Corollary 4.2 *If $Le_j = (v_{j-1}, t)$ is not the most vital real time edge with respect to the anti-risk path $P^*_G(s, t)$, then the most vital real time edge is on the shortest path tree $SPT(t)$.*

4.2 Description of the algorithm

In this section, we will put forward *Delete Most Vital Real Time Edge* (*DMVRTE*) algorithm for the ARP problem in G, and subsequently establish its correctness and running time. The main idea of algorithm is similar to the algorithm *DESPT*, but the long-edges have to be considered in G by Corollaries 4.1 and 4.2. The algorithm *DMVRTE* is formally described in the following.

Algorithm *DMVRTE*

Step 1. Produce the two shortest path trees rooted at s and t in G respectively. Record $d_G(s, x|SP)$ for every node $x \in V/\{s\}$ and $d_G(y, t|SP)$ for every node $y \in V/\{t\}$ (see Fredman and Tarjan 1987).

Step 2. Compute $d_{G-e_i}(v_{i-1}, t|SP)$ for every edge $e_i = (v_{i-1}, v_i) \in SPT_G(t)$, where v_i is closer to t than v_{i-1} (see Nardelli et al. 1998).

Step 3. Set $k = 1, \tau = 0, G_1 = G, R_0 = +\infty, R_1 = 0$ and $DE_1 = \emptyset$.

Step 4. For $e_i = (v_{i-1}, v_i) \in SP_{G_k}(s, t)$, if $e_i \notin SPT_G(t)$, then let $D_{G_k-e_i}(s, t|SP) = d_{G_k}(s, v_{i-1}|SP) + d_G(v_{i-1}, t|SP)$; If $e_i \in SPT_G(t)$, then let $D_{G_k-e_i}(s, t|SP) = d_{G_k}(s, v_{i-1}|SP) + d_{G-e_i}(v_{i-1}, t|SP)$.

Step 5. Let $D_{G_k-e^*_{G_k}}(s,t|SP) = \max_{e_i \in SP_{G_k}(s,t)}\{D_{G_k-e_i}(s,t|SP)\}$, and let $R(SP_{G_k}(s,t)) = \max\{D_{G_k-e^*_{G_k}}(s,t|SP), d_{G_k}(s,t|SP)\}$.

Step 6. If $R(SP_{G_k}(s,t)) = d_{G_k}(s,t|SP)$, then let $R_{k-1} = R(SP_{G_k}(s,t))$ and $\tau = k-1$, and then turn to Step 12.

Step 7. If $R(SP_{G_k}(s,t)) < R_{k-1}$, then let $R_k = R(SP_{G_k}(s,t))$ and $\tau = k$; otherwise, $R_k = R_{k-1}$.

Step 8. Delete $e^*_{G_k} = (x^*_{G_k}, y^*_{G_k})$, then let $G_{k+1} = G_k - e^*_{G_k}$ and $DE_{k+1} = DE_k + e^*_{G_k}$.

Step 9. Set $k = k+1$. If $k = n+1$, then turn to Step 12.

Step 10. Compute $SP_{G_k}(s,t)$ and $d_{G_k}(s,t|SP)$ in G_k.

Step 11. If $d_{G_k}(s,t|SP) < R_{k-1}$, then turn to Step 4; Otherwise, turn to Step 12.

Step 12. Output τ and $SP_{G_\tau}(s,t)$.

We begin proving the correctness of the algorithm *DMVRTE*.

Theorem 4.3 *The algorithm DMVRTE correctly computes the anti-risk path for an undirected graph G.*

Proof We prove $SP_{G_\tau}(s,t)$ is the *anti-risk path*. Consider two cases:

Case 1. If $e^*_{G_k}$ is a long-edge, then the risk of $SP_{G_k}(s,t)$ is $d_{G_k}(s,t|SP)$ which is less than R_{k-1} in Step 11. Obviously, we know $d_{G_k}(s,t|SP)$ is not larger than $d_{G_{k+1}}(s,t|SP)$ by Lemma 3.5, where $G_{k+1} = G_k - e^*_{G_k}$, and then there doesn't exist a path such that its risk is less than $d_{G_k}(s,t|SP)$ in G_k. Hence, $SP_{G_k}(s,t)$ is the *anti-risk path*.

Case 2. If each edge in DE_{k+1} in Step 8 is not a long-edge, then similar to the proof of Theorem 3.2, we can easily conclude that the minimum between $D_{G_k-e^*_{G_k}}(s,t|SP)$ and R_{k-1} in Step 7 is the lower bound of the risk of a path in G which passes through some edges in DE_{k+1} in Step 8. If $d_{G_k}(s,t|SP) \geq R_{k-1}$ in Step 11, then there doesn't exist a path whose risk can less than R_{k-1} in G_k. Hence, $SP_{G_\tau}(s,t)$ with the risk R_{k-1} is the *anti-risk path*. On the other hand, if $k = n+1$ in Step 9, n edges have been deleted and each edge joining with t in G_k must be a long-edge, then according to Case 1, $SP_{G_\tau}(s,t)$ with the risk R_{k-1} is the *anti-risk path*.

This concludes the proof of Theorem 4.3. □

For the algorithm *DMVRTE*, the following theorem holds. The proof is similar to that of Theorem 3.3, and we omit it.

Theorem 4.4 *For two given nodes s to t in G, the ARP problem can be solved in* $O(mn + n^2 \log n)$ *time, where n and m denote the number of vertices and edges in G respectively.*

5 Conclusions

In this paper, we present an interesting problem—the Anti-risk Path problem of finding a minimum risk path from source to sink. We discuss the problem in two kinds

of networks where it is in a metric space and where is in a general space. We put forward efficient algorithms running in $O(mn + n^2 \log n)$ time for the ARP problem in the two networks respectively. Some open questions include the following. (1) Are there better algorithms for solving the ARP problem? (2) What is the computational complexity for the ARP problem if one permits negative cost edges but no negative cost cycle? (3) What is the computational complexity for the ARP problem if one considers directed rather than undirected graph?

References

Ball MO, Golden BL, Vohra RV (1989) Finding the most vital arcs in a network. Oper Res Lett 8:73–76

Bar-Noy A, Khuller S, Schieber B (1995) The complexity of finding most vital arcs and nodes. TRCS-TR-3539, Institute for Advanced Studies, University of Maryland, College Park, MD

Corley HW, Sha DY (1982) Most vital links and nodes in weighted networks. Oper Res Lett 1:157–160

Fredman ML, Tarjan RE (1987) Fibonacci heaps and their uses in improved network optimization algorithms. J ACM 34(3):596–615

Hershberger J, Suri S (2001) Vickrey prices and shortest paths: what is an edge worth? In: Proceedings of the 42nd annual IEEE symposium on foundations of computer science, pp. 252–259

Hershberger J, Suri S, Bhosle A (2003) On the difficulty of some shortest path problems. In: Proceedings of the 20th symposium on theoretical aspects of computer science, pp. 343–354

Malik K, Mittal AK, Gupta SK (1989) The k most vital arcs in the shortest path problem. Oper Res Lett 8:223–227

Nardelli E, Proietti G, Widmayer P (1998) Finding the detour-critical edge of a shortest path between two nodes. Inf Process Lett 67(1):51–54

Nardelli E, Proietti G, Widmayer P (2001) A faster computation of the most vital edge of a ashortest path between two nodes. Inf Process Lett 79(2):81–85

Tarjan RE (1975) Efficiency of a good but not linear set union algorithm. J ACM 22:215–225