Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 11, No. 4, pp. 605–618, October 2015

THE COMPLEXITY ANALYSIS OF THE SHORTEST PATH IMPROVEMENT PROBLEM UNDER THE HAMMING DISTANCE[*]

Binwu Zhang, Xiucui Guan[†], Qin Wang, Chunyuan He
and Samson Hansen Sackey

Abstract: In this paper, we prove the inapproximability of the shortest path improvement problems under the sum-type Hamming distance. We first show that achieving an algorithm within a worst-case ratio of $O(\log |V|)$ is NP-hard, where V is the set of nodes. Then we propose a greedy-type heuristic algorithm to solve the problem. Numerical experiments show the effectiveness of the algorithm.

Key words: *shortest path improvement problem, Hamming distance, inapproximability, heuristic algorithm*

Mathematics Subject Classification: *68Q25, 90B10*

1 Introduction

The shortest path problems focus on finding the path with minimum distance, time, or cost from an origin to a destination through a connected network. It is a classical and important problem in the area of combinatorial optimization, and we can find numerous applications and generalizations in communication networks and transportation networks.

Recently there are many papers discussing inverse shortest path problems and the shortest path improvement problems [1, 5, 15, 14, 13, 10]. In such problems, an edge weighted graph is given with a set of source-terminal pairs, we need to modify the lengths of edges by a minimum cost under some norm. In an inverse shortest path problem, the aim is to make a set of given paths become the shortest source-terminal paths, while in a shortest path improvement problem, the aim is to make the modified distances of the shortest paths upper-bounded by given values.

Burton and Toint [1] considered the inverse shortest path problem under l_2 norm, and they transformed the problem into a quadratic programming problem using the Goldfarb-Idnani method. Zhang et al. [15] formulated the inverse shortest path problem under l_1 norm as a special linear programming problem and proposed a column generation method. Zhang and Lin [14] showed that the shortest path improvement problem under l_1 norm is **NP**-complete and proposed polynomial-time algorithms for the case of trees and the case

[*]This work is supported by The National Natural Science Foundation of China (11471073), the Project-sponsored by SRF for ROCS, SEM (BZX/11H002) and Chinese Universities Scientific Fund (2014B11414).
[†]Corresponding author.

of single source-terminal path. For detail of inverse optimization problems, the readers may refer to the survey paper by Heuberger [5].

Since the inverse minimum spanning tree problem under the sum-type Hamming distance was first studied by He et al. [4], the inverse optimization problems and network improvement problems under Hamming distance have been paid much attention (see, for example, Zhang et al. [11, 12, 13, 10], Duin and Volgenant[2], Guan and Zhang[3], Liu and Yao[7], Jiang et al. [6]).

In this paper, we consider the shortest path improvement problems under Hamming distance (denoted by **SPIH**), which can be described as follows.

Let $G = (V, E, w, l, c)$ be a connected undirected network, where V is the vertex set, $E = \{e_1, e_2, \cdots, e_m\}$ is the edge set. Let $w_i \geq 0$ be the length of edge e_i, l_i be the lower bound on the modified length of e_i, and c_i be the cost incurred by modifying length of e_i, where $i = 1, 2, \ldots, m$. Let $\{(f_k, t_k),\ k = 1, 2, \cdots, r\}$ be the set of source-terminal pairs of vertices. Let d_k be the upper bound of the shortest distance connecting the source-terminal pair (f_k, t_k). Denote by $d_w(k)$ the shortest distance from f_k to t_k under the length vector w. The **SPIH** problem is to reduce the edge length vector to w^* with $l \leq w^* \leq w$ such that the modified shortest distance of the pair (f_k, t_k) is upper bounded by d_k, and the total edge modification cost is minimized under sum-type Hamming distance, which can be formulated as a mathematical model below.

$$\begin{array}{lll} \min & \sum\limits_{i=1}^{m} c_i H(w_i^*, w_i) & \\ s.t & d_{w^*}(k) \leq d_k, & k = 1, 2, \ldots, r, \\ & l_i \leq w_i^* \leq w_i, & i = 1, 2, \ldots, m, \end{array} \tag{1.1}$$

where the Hamming distance $H(w_i^*, w_i)$ is defined as:

$$H(w_i^*, w_i) = \begin{cases} 1, & \text{if } w_i^* \neq w_i, \\ 0, & \text{if } w_i^* = w_i. \end{cases}$$

Zhang et al. showed in [13] that the **SPIH** problem on general network is strongly *NP*-hard, they also showed that even if the network is a chain network, it is still *NP*-hard. Zhang et al. [10] considered the shortest path improvement problems on arborescent network under unit Hamming distance. Firstly it was formulated as a 0-1 integer programming model. Secondly, some strongly polynomial-time algorithms were designed for the problems on some special arborescent networks. Thirdly, a heuristic algorithm was proposed for the problem on general graphs [10]. However, it is still unknown whether there is an algorithm within constant-bounded approximation rate for the SPIH on general network. So it is meaningful to consider the complexity of approximation for the SPIH.

The shortest path improvement problems under Hamming distance have practical background. For example, a large earthquake happened in some towns. The relief goods need to be quickly handed out to the people in the stricken area. However, many roads were badly damaged and we have to repair the roads as soon as possible. Consider the network $G = (V, E, w, l, c)$ of towns, where $v \in V$ denotes a town, $e = (u, v) \in E$ denotes a road connecting u to v. Let w_i be the travelling time through the road e_i if e_i is not repaired and l_i denotes the travelling time through the road e_i after e_i is repaired. Note that some points of the road e_i are damaged (rather than every point of the road is destroyed), so the repair time for the road e_i may be a fixed amount c_i. Our objective is to design a repair scheme so that the relief goods are transported from some supplier town f_k to some stricken town t_k $(k = 1, 2, \ldots, r)$within the stipulated time and the total repair time is minimized. This is just the shortest path improvement problem under Hamming distance [10].

The outline of the paper is as follows. In Section 2, we show that achieving an algorithm with a worst-case ratio of $O(\log |V|)$ is NP-hard for the **SPIH**. In Section 3, we present a heuristic algorithm to solve the **SPIH**. In Section 4, numerical experiments are given to show the effectiveness of the algorithm. Conclusion and further research are given in Section 5.

$\boxed{2}$ Inapproximability of the SPIH Problem

In this section, we first introduce an L-reduction from an instance of the Set-Cover problem to an instance of **SPIH** problem, then show that approximately solving the **SPIH** problem within a worst-case ratio of $O(\log |V|)$ is NP-hard.

$\boxed{2.1}$ An L-reduction from the Set-Cover problem to the SPIH problem

First we introduce some definitions and a preliminary lemma.

Definition 2.1. [8] An NP-hard minimization problem is approximable with a worst-case ratio $\delta > 1$ if there exists a polynomial-time algorithm which produces for every instance a solution of objective value at most δ times of the optimal value.

The Set-Cover problem, which is used in this paper, can be described as follows: Let $T = \{t_1, t_2, \ldots, t_n\}$ be a finite set, and $S_1, S_2, \ldots, S_m$ be a collection of its subsets satisfying $\bigcup_{j=1}^{m} S_j = T$, find the minimum cardinality subset $\{S_{j_1}, S_{j_2}, \ldots, S_{j_q}\}$ in $\{S_1, S_2, \ldots, S_m\}$ that covers T, that is, $\bigcup_{\tau=1}^{q} S_{j_\tau} = T$ and q is minimized.

Lemma 2.2 ([9])**.** It is NP-hard to approximate the Set-Cover problem within a worst-case ratio of $O(\log n)$, if $m = O(p(n))$, where $p(n)$ is a polynomial function of n.

Now we introduce an L-reduction from an instance of the Set-Cover problem to an instance of **SPIH** problem.

Let I be an instance of the Set-Cover problem, that is, a collection of m subsets $\{S_1, S_2, \ldots, S_m\}$ of $T = \{t_1, t_2, \ldots, t_n\}$, which satisfies $\bigcup_{j=1}^{m} S_j = T$. We construct an undirected network $G = (V, E, w, l, c)$ and an instance $\tau(I)$ of **SPIH** problem in the following three steps.

(1) For each element $t_i \in T$, introduce an *element vertex* t_i in V. For each subset S_j of instance I, introduce a *subset vertex* S_j in V. In addition, introduce $n+1$ vertices $S, f_1, f_2, \ldots, f_n$ in V. Thus $V = F \cup \{S\} \cup S^* \cup T$, where $F = \{f_1, f_2, \ldots, f_n\}$ and $S^* = \{S_1, S_2, \ldots, S_m\}$.

(2) For each vertex $f_i \in F$, introduce an edge $(f_i, S) \in E$ linking f_i and vertex S. For each subset vertex $S_j \in S^*$, $j = 1, 2, \ldots, m$, introduce an edge $(S, S_j) \in E$ linking S_j and vertex S, and introduce an edge $(S_j, t_i) \in E$ linking S_j and the element vertex t_i if and only if t_i is an element of S_j in instance I. Thus $E = E_1 \cup E_2 \cup E_3$, where $E_1 = \{(f_i, S) : \ f_i \in F\}$, $E_2 = \{(S, S_j) : \ S_j \in S^*\}$, and $E_3 = \bigcup\limits_{j=1}^{n} \bigcup\limits_{i=1}^{m} \{(S_j, t_i) : \ t_i \textit{ is an element of } S_j \textit{ in instance } I\}$.

(3) For each edge $e \in E$, let its length be $w(e) = 1$, the lower bound of the reduced length be $l(e) = 0$, and the cost be $c(e) = 1$ if the edge is modified. For each source-terminal pair (f_k, t_k), $k = 1, 2, \cdots, n$, let $d_k = 2$, where d_k is the upper bound of the shortest distance connecting the source-terminal pair (f_k, t_k).

Notice that the constructed graph $G = (V, E, w, l, c)$ is a tetrapartite graph and the number of edges in any path from f_k to t_k is odd.

We give an example to illustrate the construction of the instance $\tau(I)$. Let $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$, $S_1 = \{t_1, t_3, t_4\}$, $S_2 = \{t_1, t_2, t_4, t_6\}$, $S_3 = \{t_2, t_3, t_5\}$, $S_4 = \{t_1, t_3, t_5\}$, and $S_5 = \{t_4, t_6\}$. Then the corresponding network G is shown in Figure 1.

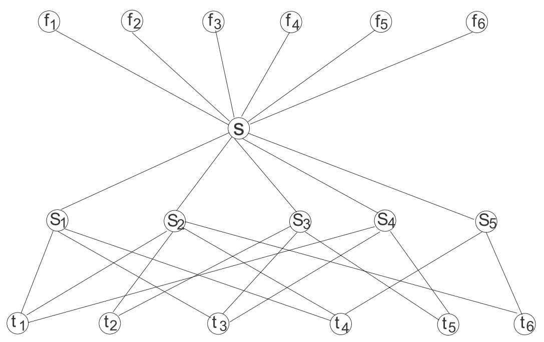

Figure 1: An example of instance $\tau(I)$

$\boxed{2.2}$ Proof of Inapproximability of the SPIH problem

In this subsection, we first propose an algorithm to construct a length vector w'' satisfying some properties, based on which we can show that the **SPIH** problem is equivalent to the Set-Cover problem under L-reduction introduced in Subsection 2.1. Then we prove the Inapproximability of the **SPIH** problem.

For convenience, in this section we use the following symbols:
$(u, v_1, v_2, \ldots, v_k, v)$: the path between vertices u and v passing through vertices v_1, v_2, $\ldots$, v_k in sequence;
$w(u, v)$: the length of edge (u, v) under the length vector w;
$P(i)$: a path between vertices f_i and t_i;
$d_w(P)$: the length of path P under the length vector w;
$P_w(i)$: the shortest path between vertices f_i and t_i under w;
$d_w(i)$: the length of the shortest path $P_w(i)$;
$|P_w(i)|$: the number of edges in path $P_w(i)$;
$P_w^{u,v}(i)$: the sub-path between vertices u and v along the path $P_w(i)$.
$d_w^{u,v}(i)$: the length of sub-path $P_w^{u,v}(i)$.

For a feasible solution w' of instance $\tau(I)$, we denote by $\beta_{w'}(\tau(I)) = \sum_{e\in E} H(w'(e), w(e))$ as the cost (objective) function. Let $\beta(I)$ and $\beta(\tau(I))$ be the optimal values of instance I and instance $\tau(I)$, respectively.

We first give some definitions.

Definition 2.3. For the vertex S and set $T' \subset T$, let $t_h = \arg\max\left\{d_{w'}^{S,t_i}(i) : t_i \in T'\right\}$, then we call t_h the **farthest element vertex** in T' to vertex S under w'.

Definition 2.4. For the element vertex t_h, if the set vertex $S_{j_h} \in S^*$ satisfies that $|P_{w'}^{S_{j_h},t_h}(h)| = \min\left\{|P_{w'}^{S_j,t_h}(h)| : S_j \in S^*\right\}$, then we call S_{j_h} the **nearest set vertex** to vertex t_h under w'.

Definition 2.5. For the element vertex t_h, if the element vertex $t_{i_h} \in T'\backslash\{t_h\}$ satisfies that $|P_{w'}^{S_{j_h},t_h}(h)| = \min\left\{|P_{w'}^{S_j,t_h}(h)| : S_j \in S^*\right\}$, $|P_{w'}^{t_{i_h},t_h}(h)| = \min\left\{|P_{w'}^{t_i,t_h}(h)| : t_i \in T'\backslash\{t_h\}\right\}$, then we call t_{i_h} the **nearest element vertex** to vertex t_h under w'.

Now we propose an algorithm to construct a length vector w'', followed by proving that w'' is a feasible solution of instance $\tau(I)$ and $\beta_{w''}(\tau(I)) \leq \beta_{w'}(\tau(I))$.

Algorithm A
Input: $T = \{t_1, t_2, \ldots, t_n\}$, any feasible solution w'.
Step 1: Put $w^0 \leftarrow w'$ and $T' \leftarrow T$.
Step 2: Find the farthest element vertex t_h in T' to vertex S under w^0. Get $P_{w^0}(h)$, which is one of the shortest paths between vertices f_h and t_h under w^0.
Step 3: If $|P_{w^0}(h)| \geq 5$, find the nearest subset vertex S_{j_h} to t_h in the path $P_{w^0}(h)$, and find the nearest element vertex t_{i_h} to t_h in the path $P_{w^0}(h)$. Set $E_1^h = \{(f_h, S), (S, S_{j_h}), (t_{i_h}, S_{j_h}), (S_{j_h}, t_h)\}$. For each edge (u, v) in E, define

$$\bar{w}(u,v) = \begin{cases} 0, & \text{if } (u,v) = (S, S_{j_h}), \\ 1, & \text{if } (u,v) \in E_1^h \backslash (S, S_{j_h}), \\ w^0(u,v), & \text{if } (u,v) \in E \backslash E_1^h. \end{cases} \tag{2.1}$$

Step 4: If $|P_{w^0}(h)| = 3$, find the only subset vertex S_{j_h} between S to t_h in the path $P_{w^0}(h)$. Set $E_2^h = \{(f_h, S), (S, S_{j_h}), (S_{j_h}, t_h)\}$. For each edge (u, v) in E, define

$$\bar{w}(u,v) = \begin{cases} 0, & \text{if } (u,v) = (S, S_{j_h}), \\ 1, & \text{if } (u,v) \in E_2^h \backslash (S, S_{j_h}), \\ w^0(u,v), & \text{if } (u,v) \in E \backslash E_2^h. \end{cases} \tag{2.2}$$

Step 5: $T' \leftarrow T' \setminus \{t_h\}$. If $T' = \emptyset$, then $w'' \leftarrow \bar{w}$, **go to Step 6**, else $w^0 \leftarrow \bar{w}$, **go to Step 2**.
Step 6: Output the length vector w''.

We designate computations starting from Step 2 until switching back to the next Step 2 in Algorithm A as **one iteration**. Next we present two lemmas to analyze the properties of Algorithm A.

Lemma 2.6. In **Algorithm A**, once the lengths of edges in E_1^h or E_2^h are assigned to 0 or 1 by formulae (2.1) or (2.2), they will not be changed in the subsequent iterations.

Proof. We only show that the lemma holds in the case of E_1^h, since it is similar to prove it in the case of E_2^h. According to Algorithm A, every $t_i \in T$ can be taken as the farthest element vertex t_h at some iteration in Algorithm A. Note that the edge (f_h, S) is only appeared in $P_{w^0}(h)$, so once $\bar{w}(f_h, S) = 1$, it cannot be changed in other paths. Once $\bar{w}(S, S_{j_h}) = 0$, no changes on (S, S_{j_h}) will be made in the subsequent iterations. Once $\bar{w}(t_{i_h}, S_{j_h}) = 1$ or $\bar{w}(S_{j_h}, t_h) = 1$, the $\bar{w}$ value will never be designed to 0 since only edges like (S, S_{j_h}) can be changed to 0. □

Lemma 2.7. For each element vertex t_i, there exists a subset vertex S_{j_i} satisfying $w''(S, S_{j_i}) = 0$, and $w''(S_{j_i}, t_i) = 1$;

Proof. According to Algorithm A, every $t_i \in T$ can be taken as the farthest element vertex t_h at some iteration in Algorithm A. For each $t_i \in T$, there is a (nearest) subset vertex S_{j_i} satisfying $\bar{w}(S, S_{j_i}) = 0$, and $\bar{w}(S_{j_i}, t_i) = 1$. By Lemma 2.6, the values $\bar{w}(S, S_{j_i})$ and $\bar{w}(S_{j_i}, t_i)$ will not be modified in the subsequent iterations. Finally, when $T' = \emptyset$, we have $w''(S, S_{j_i}) = 0$ and $w''(S_{j_i}, t_i) = 1$ for each element vertex t_i. □

Now we use the induction method to show that w'' is a feasible solution of instance $\tau(I)$ and $\beta_{w''}(\tau(I)) \le \beta_{w'}(\tau(I))$. To make it clearer, we use three lemmas to prove it.

Lemma 2.8. *$\bar{w}$ generated in the first iteration of* **Algorithm A** *is a feasible solution of $\tau(I)$.*

Proof. Before running Step 5 of the first iteration, we have $w^0 = w'$, $T' = T$, t_h is the farthest element vertex. We need to consider two cases: $|P_{w^0}(h)| \ge 5$ and $|P_{w^0}(h)| = 3$. However, we only consider the first case since the proofs are similar in two cases.

Next we prove, in the case of $|P_{w^0}(h)| \ge 5$, the feasibility of $\bar{w}$, that is, the distance $d_{\bar{w}}(i)$ between f_i and t_i is not more than 2 for each i $(1 \le i \le n)$.

Assume that

$$P_{w^0}(h) = (f_h, S, \ldots, t_{i_h}, S_{j_h}, t_h). \tag{2.3}$$

It is clear that $d_{\bar{w}}(h) \le \bar{w}(f_h, S) + \bar{w}(S, S_{j_h}) + \bar{w}(S_{j_h}, t_h) = 2$ and $d_{\bar{w}}(i_h) \le \bar{w}(f_{i_h}, S) + \bar{w}(S, S_{j_h}) + \bar{w}(S_{j_h}, t_{i_h}) \le 2$.

Next we show that $d_{\bar{w}}(i) \le 2$ for each element vertex $t_i \in \{t_1, t_2, \ldots, t_n\} \backslash \{t_h, t_{i_h}\}$. Assume that $P_{w^0}(i)$ is the shortest path between vertices f_i and t_i under w^0 with the minimum number of edges. Obviously $P_{w^0}(i)$ is acyclic. We have $d_{w^0}(i) \le 2$, since w^0 is a feasible solution of $\tau(I)$, Next we prove that for all possible subcases of path $P_{w^0}(i)$, we can always construct a path $P(i)$ connecting f_i to t_i, such that $d_{\bar{w}}(P(i)) \le 2$, which leads to $d_{\bar{w}}(i) \le 2$.

Case 1.1 $E_1^h \cap P_{w^0}(i) = \emptyset$. In this case we can take $P(i)$ as path $P_{w^0}(i)$. In fact, by formula (2.1), we know that $d_{\bar{w}}(P(i)) = d_{\bar{w}}(P_{w^0}(i)) = d_{w^0}(P_{w^0}(i)) \le 2$.

Case 1.2 $|E_1^h \cap P_{w^0}(i)| = 1$. There are three subcases based on which edge is the common one.

Case 1.2.1 $E_1^h \cap P_{w^0}(i) = (S, S_{j_h})$. Then we still take $P(i)$ as $P_{w^0}(i)$, and thus

$$\begin{aligned} d_{\bar{w}}(P(i)) &= d_{\bar{w}}(P_{w^0}(i)) = d_{\bar{w}}(P_{w^0}(i) \backslash (S, S_{j_h})) + \bar{w}(S, S_{j_h}) \\ &= d_{w^0}(P_{w^0}(i) \backslash (S, S_{j_h})) \le d_{w^0}(P_{w^0}(i)) \le 2. \end{aligned}$$

Case 1.2.2 $E_1^h \cap P_{w^0}(i) = (S_{j_h}, t_{i_h})$.

(a) If $P_{w^0}(i)$ is in the form of $(f_i, S, \ldots, \underline{S_{j_h}, t_{i_h}}, \ldots, t_i)$, construct path $P(i) = (f_i, S) \cup P_{w^0}^{S, t_{i_h}}(h) \cup P_{w^0}^{t_{i_h}, t_i}(i)$. Note that neither of these two parts of $P(i)$ includes the edge (S_{j_h}, t_{i_h}). Because $P_{w^0}^{S, t_{i_h}}(h)$ and $P_{w^0}^{S, t_{i_h}}(i)$ are the shortest paths between S and t_{i_h} along the paths $P_{w^0}(h)$ and $P_{w^0}(i)$ under w^0, respectively, we have $d_{w^0}(P_{w^0}^{S, t_{i_h}}(h)) = d_{w^0}(P_{w^0}^{S, t_{i_h}}(i))$. Hence we can get

$$\begin{aligned} d_{\bar{w}}(P(i)) &= \bar{w}(f_i, S) + d_{\bar{w}}(P_{w^0}^{S, t_{i_h}}(h)) + d_{\bar{w}}(P_{w^0}^{t_{i_h}, t_i}(i)) \\ &= w^0(f_i, S) + d_{w^0}(P_{w^0}^{S, t_{i_h}}(i)) + d_{w^0}(P_{w^0}^{t_{i_h}, t_i}(i)) \\ &= d_{w^0}(P_{w^0}(i)) \le 2. \end{aligned}$$

(b) If $P_{w^0}(i)$ is in the form of $(f_i, S, \ldots, \underline{t_{i_h}, S_{j_h}}, \ldots, t_i)$, construct the path $P(i) = (f_i, S) \cup (S, S_{j_h}) \cup P_{w^0}^{S_{j_h}, t_i}(i)$. we can get that

$$\begin{aligned} d_{\bar{w}}(P(i)) &= \bar{w}(f_i, S) + \bar{w}(S, S_{j_h}) + d_{\bar{w}}(P_{w^0}^{S_{j_h}, t_i}(i)) \\ &= w^0(f_i, S) + d_{w^0}(P_{w^0}^{S_{j_h}, t_i}(i)) \\ &\le d_{w^0}(P_{w^0}(i)) \le 2. \end{aligned}$$

Case 1.2.3 $E_1^h \cap P_{w^0}(i) = (S_{j_h}, t_h)$.

(a) If $P_{w^0}(i)$ is in the form of $(f_i, S, \ldots, \underline{t_h, S_{j_h}}, \ldots, t_i)$, construct the path $P(i) = (f_i, S) \cup (S, S_{j_h}) \cup P_{w^0}^{S_{j_h}, t_i}(i)$. In this situation, by the same argument as (b) of Case 1.2.2, we can show that $d_{\bar{w}}(P(i)) \leq 2$.

(b) If $P_{w^0}(i)$ is in the form of $(f_i, S, \ldots, \underline{S_{j_h}, t_h}, S_{j_\tau}, \ldots, t_i)$, construct the path $P(i) = (f_i, S) \cup (S, S_{j_\tau}) \cup P_{w^0}^{S_{j_\tau}, t_i}(i)$. Since t_h is the farthest element vertex in T' to vertex S under w^0, we have $d_{w^0}(P_{w^0}^{S_{j_\tau}, t_i}(i)) = 0$. So, we can easily get that

$$d_{\bar{w}}(P(i)) = w^0(f_i, S) + w^0(S, S_{j_\tau}) + d_{w^0}(P_{w^0}^{S_{j_\tau}, t_i}(i)) \leq 2.$$

Case 1.3 $|E_1^h \cap P_{w^0}(i)| = 2$. Similarly, we consider all three possible subcases.

Case 1.3.1 $E_1^h \cap P_{w^0}(i) = \{(S, S_{j_h}), (S_{j_h}, t_{i_h})\}$. As $P_{w^0}(i)$ is acyclic, it must take the form

$$P_{w^0}(i) = (f_i, \underline{S, S_{j_h}, t_{i_h}}, \ldots, t_i). \tag{2.4}$$

We construct the path $P(i) = (f_i, S) \cup P_{w^0}^{S, t_{i_h}}(h) \cup P_{w^0}^{t_{i_h}, t_i}(i)$. Note that by (2.3) and (2.4), $P(i) \bigcap E_1^h = \emptyset$. By the same argument as (a) of Case 1.2.2, we can get $d_{\bar{w}}(P(i)) \leq 2$.

Case 1.3.2 $E_1^h \cap P_{w^0}(i) = \{(S, S_{j_h}), (S_{j_h}, t_h)\}$. By the same reason as in Case 1.3.1, $P_{w^0}(i)$ must take the form $(f_i, \underline{S, S_{j_h}, t_h}, \ldots, t_i)$. We take $P(i)$ as the path $P_{w^0}(i)$. Note that $d_{w^0}(P_{w^0}^{t_h, t_i}(i)) = 0$. We can get that

$$\begin{aligned} d_{\bar{w}}(P(i)) &= \bar{w}(f_i, S) + \bar{w}(S, S_{j_h}) + \bar{w}(S_{j_h}, t_h) + d_{\bar{w}}(P_{w^0}^{t_h, t_i}(i)) \\ &= w^0(f_i, S) + 1 + d_{w^0}(P_{w^0}^{t_h, t_i}(i)) \\ &= w^0(f_i, S) + 1 \leq 2. \end{aligned}$$

Case 1.3.3 $E_1^h \cap P_{w^0}(i) = \{(S_{j_h}, t_{i_h}), (S_{j_h}, t_h)\}$. As $P_{w^0}(i)$ is acyclic, it must take either the form $P_{w^0}^1(i) = (f_i, S, \ldots, \underline{t_{i_h}, S_{j_h}, t_h}, \ldots, t_i)$ or $P_{w^0}^2(i) = (f_i, S, \ldots, \underline{t_h, S_{j_h}, t_{i_h}}, \ldots, t_i)$. We construct the path

$$P(i) = \begin{cases} (f_i, S) \cup (S, S_{j_h}) \cup (S_{j_h}, t_h) \cup P_{w^0}^{t_h, t_i}(i), & \text{if } P_{w^0}(i) = P_{w^0}^1(i), \\ (f_i, S) \cup (S, S_{j_h}) \cup (S_{j_h}, t_{i_h}) \cup P_{w^0}^{t_{i_h}, t_i}(i), & \text{if } P_{w^0}(i) = P_{w^0}^2(i). \end{cases}$$

By the same argument as in Case 1.2.3, we can get $d_{\bar{w}}(P(i)) \leq 2$.

Case 1.4 $E_1^h \subseteq P_{w^0}(i)$. In this case, the path $P_{w^0}(i)$ must contain a cycle, contradicting the definition of $P_{w^0}(i)$. Thus this subcase does not hold.

Based on the above discussion, we can conclude that the $\bar{w}$ generated in the first iteration is a feasible solution of $\tau(I)$. □

Lemma 2.9. The length vector w'' outputted by **Algorithm A** is a feasible solution of $\tau(I)$.

Proof. We use the induction method to show this result. It follows from Lemma 2.8 that the conclusion holds in the first iteration.

Now assume that the current iteration is not the first one. Let $(\hat{w}, \hat{T})$ and $(\tilde{w}, \tilde{T})$ be the $(\bar{w}, \bar{T})$ obtained in the previous iteration and the current iteration, respectively. Suppose $\hat{w}$ is a feasible solution of $\tau(I)$. We show that $\tilde{w}$ is also a feasible solution of $\tau(I)$, that is, $d_{\tilde{w}}(i) \leq 2$ for each $1 \leq i \leq n$. There are two possible cases:

(1) $t_i \in T \setminus \hat{T}$. Such t_i must be chosen as the farthest element vertex t_h in one of the previous iterations. By Algorithm A and Lemma 2.6, once the length of an edge is changed, it cannot be modified in the subsequent iterations. Thus for $t_i \in T \setminus \hat{T}$, there exists a

subset vertex S_{j_i} such that $\tilde{w}(S, S_{j_i}) = \hat{w}(S, S_{j_i}) = 0$ and $\tilde{w}(S_{j_i}, t_i) = \hat{w}(S_{j_i}, t_i) = 1$. So, $d_{\tilde{w}}(i) \leq \tilde{w}(f_i, S) + \tilde{w}(S, S_{j_i}) + \tilde{w}(S_{j_i}, t_i) \leq 2$.

(2) $t_i \in \hat{T}$. By the same argument as that in the first iteration, we can show that $d_{\tilde{w}}(i) \leq 2$.

Thus $\tilde{w}$ is a feasible solution of $\tau(I)$. Note that w'' is $\bar{w}$ generated in the last iteration. So, w'' is a feasible solution of $\tau(I)$ by the induction method. $\square$

Theorem 2.10. Let w' and w'' be the length vectors inputted and outputted by **Algorithm A**, respectively. Then

$$\beta_{w''}(\tau(I)) \leq \beta_{w'}(\tau(I)). \tag{2.5}$$

Proof. Similar to proof of Lemma 2.9, we use the induction method.

In the first iteration, we have $w^0 = w'$, $T' = T$, t_h is the farthest element vertex.

Case 1: $|P_{w^0}(h)| \geq 5$. Because w^0 is a feasible solution of instance $\tau(I)$, we have

$$d_{w^0}(h) = w^0(f_h, S) + d_{w^0}^{S, t_{i_h}}(h) + w^0(t_{i_h}, S_{j_h}) + w^0(S_{j_h}, t_h) \leq 2.$$

Hence at least one of the three edges $(f_h, S), (t_{i_h}, S_{j_h})$ and (S_{j_h}, t_h) has been shortened. Then we have

$$\begin{aligned} \beta_{w^0}(\tau(I)) &= \sum_{e \notin E_1^h} H(w^0(e), w(e)) \\ &\quad + \sum_{e \in E_1^h \setminus \{(S, S_{j_h})\}} H(w^0(e), w(e)) + H(w^0(S, S_{j_h}), w(S, S_{j_h})) \\ &\geq \sum_{e \notin E_1^h} H(w^0(e), w(e)) + 1 \end{aligned} \tag{2.6}$$

Furthermore, we can know that

$$\begin{aligned} \beta_{\bar{w}}(\tau(I)) &= \sum_{e \notin E_1^h} H(\bar{w}(e), w(e)) + \sum_{e \in E_1^h \setminus (S, S_{j_h})} H(\bar{w}(e), w(e)) + H(\bar{w}(S, S_{j_h}), w(S, S_{j_h})) \\ &= \sum_{e \notin E_1^h} H(w^0(e), w(e)) + \sum_{e \in E_1^h \setminus (S, S_{j_h})} H(w(e), w(e)) + H(0, 1) \\ &= \sum_{e \notin E_1^h} H(w^0(e), w(e)) + 0 + 1 \end{aligned} \tag{2.7}$$

It follows from (2.6) and (2.7) that $\beta_{\bar{w}}(\tau(I)) \leq \beta_{w^0}(\tau(I))$.

Case 2: $|P_{w^0}(h)| = 3$. Because w^0 is a feasible solution of instance $\tau(I)$, we have

$$d_{w^0}(h) = w^0(f_h, S) + w^0(S, S_{j_h}) + w^0(S_{j_h}, t_h) \leq 2.$$

Hence at least one of the three edges $(f_h, S), (S, S_{j_h})$ and (S_{j_h}, t_h) has been shortened. Then we have

$$\begin{aligned} \beta_{w^0}(\tau(I)) &= \sum_{e \notin E_2^h} H(w^0(e), w(e)) + \sum_{e \in E_2^h} H(w^0(e), w(e)) \\ &\geq \sum_{e \notin E_2^h} H(w^0(e), w(e)) + 1 \end{aligned} \tag{2.8}$$

Furthermore, we can know that

$$\begin{aligned}\beta_{\bar{w}}(\tau(I)) &= \sum_{e\notin E_2^h} H(\bar{w}(e),w(e)) + \sum_{e\in E_2^h\setminus(S,S_{j_h})} H(\bar{w}(e),w(e)) + H(\bar{w}(S,S_{j_h}),w(S,S_{j_h}))\\ &= \sum_{e\notin E_2^h} H(w^0(e),w(e)) + \sum_{e\in E_2^h\setminus(S,S_{j_h})} H(w(e),w(e)) + H(0,1)\\ &= \sum_{e\notin E_2^h} H(w^0(e),w(e)) + 0 + 1 \qquad (2.9)\end{aligned}$$

It follows from (2.8) and (2.9) that $\beta_{\bar{w}}(\tau(I)) \le \beta_{w^0}(\tau(I))$.

Combining the results in two cases, we can conclude that $\beta_{\bar{w}}(\tau(I)) \le \beta_{w^0}(\tau(I))$.

Now assume that the current iteration is not the first one. Let $\hat{w}$ and $\tilde{w}$ be the $\bar{w}$ obtained in the previous iteration and the current iteration, respectively. It follows from Lemma 2.9, $\hat{w}$ is a feasible solution of $\tau(I)$. Similar to the first iteration, $\hat{w}$ is the w^0 inputted in the current iteration and $\tilde{w}$ is the $\bar{w}$ obtained in the current iteration. Hence, formulae (2.6)-(2.9) still hold and we have $\beta_{\tilde{w}}(\tau(I)) \le \beta_{\hat{w}}(\tau(I))$.

Therefore, by the induction method, we can conclude that $\beta_{w''}(\tau(I)) \le \beta_{w'}(\tau(I))$. □

Theorem 2.11. For each feasible solution w' of $\tau(I)$, we can construct a feasible solution SC of instance I in polynomial time such that $|SC| \le \beta_{w'}(\tau(I))$, and therefore, $\beta(I) \le \beta(\tau(I))$.

Proof. Let w'' be outputted by Algorithm A with input vector w' and the set T. We now define a set SC by w'': $SC = \{S_j :\ w''(S,S_j) = 0\}$. We next prove that SC is a set-cover of T. By Lemma 2.7, for each element vertex t_i, $1 \le i \le n$, there exists a subset vertex S_{j_i} such that $w''(S,S_{j_i}) = 0$ and $w''(S_{j_i},t_i) = 1$. So, $S_{j_i} \in SC$. Moreover, the subset vertex S_{j_i} is adjacent to the element vertex t_i in G, and hence by the construction of network G, $t_i \in S_{j_i}$, which means that SC is a set-cover. For each $S_j \in SC$, $w''(S,S_j) = 0$. But $w(S,S_j) = 1$, hence $H(w''(S,S_j),w(S,S_j)) = 1$. So,

$$\beta_{w''}(\tau(I)) \ge \sum_{j=1}^{m} H(w''(S,S_j),w(S,S_j)) \ge \sum_{S_j\in SC} H(w''(S,S_j),w(S,S_j)) = |SC|.$$

Due to (2.5) and $|SC| \ge \beta(I)$, we know that $\beta_{w'}(\tau(I)) \ge \beta(I)$ for every feasible solution of instance $\tau(I)$, which means $\beta(\tau(I)) \ge \beta(I)$.

It remains to prove the time complexity of the construction of SC, which is the same as that of Algorithm A. Obviously, there are $|T|$ iterations in Algorithm A. And in each iteration, the main computation is to find the farthest element vertex, which needs $O(|T||V|^2)$ operations. Hence the total running time is $O(|T|^2|V|^2)$ in the worst-case. □

Theorem 2.12. For any set-cover SC of T, we can define a feasible solution w' of $\tau(I)$ in polynomial time such that $|SC| = \beta_{w'}(\tau(I))$, and therefore, $\beta(\tau(I)) \le \beta(I)$.

Proof. Let $SC = \{S_{j_1}, S_{j_2}, \ldots, S_{j_q}\}$ be an arbitrary set-cover of T. We define w' on the network $G(V,E,w,l,c)$ constructed as follows:

$$w'(u,v) = \begin{cases} 0, & \text{if } u = S,\ v \in SC,\\ 1, & \text{otherwise.}\end{cases}$$

We now prove that w' is a feasible solution of instance $\tau(I)$. For each element vertex t_i, $1 \le i \le n$, there exists $S_{j_i} \in SC$, such that $t_i \in S_{j_i}$. Thus the subset vertex S_{j_i} is adjacent to the element vertex t_i in G, and $d_{w'}(i) \le w'(f_i, S) + w'(S, S_{j_i}) + w'(S_{j_i}, t_i) = 2$. Therefore, w' is a feasible solution of $\tau(I)$. And we know that $\beta(\tau(I)) \le \beta_{w'}(\tau(I)) = |SC|$ for every set-cover SC, which means that $\beta(\tau(I)) \le \beta(I)$. Obviously, the construction of w' is in polynomial time. □

We are now ready to present the main result of this section.

Theorem 2.13. Even if $c(e) = 1$ and $l(e) = 0$ for each edge e of the network, approximately solving the **SPIH** problem within a worst-case ratio of $O(\log |V|)$ is NP-hard.

Proof. By the L-reduction defined in Subsection 2.1 and Theorems 2.11 and 2.12, we have $\beta(\tau(I)) = \beta(I)$. Thus, the **SPIH** problem is equivalent to the Set-Cover problem under the L-reduction. Also, from Theorem 2.11, for each feasible solution w' of the instance $\tau(I)$ in the **SPIH** problem, there is a set cover SC of the instance I such that $|SC| \le \beta_{w'}(\tau(I))$. So, we can obtain

$$\frac{|SC| - \beta(I)}{\beta(I)} = \frac{|SC| - \beta(\tau(I))}{\beta(\tau(I))} \le \frac{\beta_{w'}(\tau(I)) - \beta(\tau(I))}{\beta(\tau(I))}.$$

Moreover, if the **SPIH** problem has an approximation algorithm with a performance ratio of $\delta \log n$, so does the Set-Cover problem. As the number of vertices of G is $|V| = m+2n+1$, by the assumption $m = O(p(n))$, we know that $\log |V| = O(\log n)$. Therefore, the theorem follows from Lemma 2.2. □

$\boxed{3}$ A Heuristic Algorithm of the SPIH Problem

In this section, we design a greedy-type heuristic algorithm to solve the **SPIH** problem (1.1).

we can use the next lemma to check the feasibility of **SPIH** problem.

Lemma 3.1. The **SPIH** problem (1.1) is feasible if and only if $d_l(f_k, t_k) \le d_k$, for each $k = 1, 2, \ldots, r$.

Now we design a heuristic algorithm to solve the **SPIH** problem. The main idea is as follows.

Suppose the **SPIH** problem is feasible. Initialize $l' = l$. Define $K = \{k : 1 \le k \le r, d_w(k) > d_k\}$. First compute a shortest path P_k from f_k to t_k under l' for each $k \in K$. Let $X = \{e_i : e_i \in P_k, k \in K\}$ and $X' = \{e_i : e_i \in X, w_i - l'_i > 0\}$. Then find a specific edge e_j, and update $l'_j = w_j$ and $X' = X' - \{e_j\}$. Next we check if the **SPIH** problem is feasible or not under the current vector l'. If not, modify l'_j to the original value l_j. Repeat the above process until $X' = \emptyset$. Note that there is an edge deleted from X' in each iteration and the cardinality of the original set X' is at most m, and hence the algorithm can be done in m iterations.

Algorithm B (A greedy-type heuristic algorithm)

Input: A network $G(V, E, w, l, c)$ with a set $\{(f_k, t_k) : k = 1, 2, \ldots, r\}$ of source-terminal pairs, three edge length vectors w, l, c and a set $\{d_k : k = 1, 2, \ldots, r\}$ of values.

Step 1: For each $k = 1, 2, \ldots, r$, compute $d_l(k)$ and $d_w(k)$. If there exists a shortest path P_k from f_k to t_k under l such that $d_l(P_k) > d_k$, then output that the **SPIH** problem has no feasible solution, stop. Otherwise, let $l' = l$ and $K = \{k : 1 \le k \le r, d_w(k) > d_k\}$. For each $k \in K$, compute a shortest path P_k from f_k to t_k under l'.

Step 2: Let $X = \{e_i : e_i \in P_k,\ k \in K\}$. For each edge $e_i \in X$, let $Q_i = \{P_k : e_i \in P_k\}$. Let $X' = \{e_i : e_i \in X, w_i - l'_i > 0\}$.

Step 3: While $X' \neq \emptyset$, do

Find the edge e_j such that $\frac{(w_j - l'_j)|Q_j|}{c_j} = \min\left\{\frac{(w_i-l'_i)|Q_i|}{c_i} : e_i \in X'\right\}$, and let $l'_j = w_j$ and $X' = X' \backslash \{e_j\}$.

Compute a shortest path P_k from f_k to t_k under l' in the set Q_j of paths.

If there exists a $P_k \in Q_j$ satisfying $d_{l'}(P_k) > d_k$, then let $l'_j = l_j$, otherwise, let $X = \{e_i : e_i \in P_k,\ k \in K\}$ and $Q_i = \{P_k : e_i \in P_k\}$ for each edge $e_i \in X$.

Step 4: Let $E^* = \{e_i : e_i \in X, l'_i = l_i, w_i \neq l_i\}$, output an approximation solution w^* and its objective value $\sum\limits_{e_i \in E^*} c_i$ of the **SPIH** problem, where $w^*_i = \begin{cases} l_i, & \text{if } e_i \in E^*, \\ w_i, & \text{otherwise.} \end{cases}$

Now we analyze the time complexity of Algorithm B. For each $k = 1, 2, \ldots, r$, there are $O(|V|^2)$ operations to compute a shortest path P_k; and hence there are $O(r|V|^2)$ operations in Step 1. Step 2 requires $O(|V|m)$ operations. Step 3 requires $O(r^2|V|^3)$ operations. Furthermore, there are at most m iterations in Algorithm B. Hence Algorithm B runs in $O(r^2|V|^3 m)$ operations in the worst-case and it is a strongly polynomial-time approximation algorithm.

$\boxed{4}$ Computational Experiments of Algorithm B

In this section, we first give an example to explain the detailed computational process of Algorithm B, and then present its computational experiments.

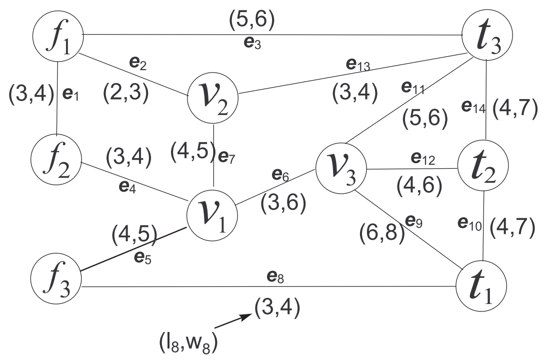

Figure 2: An example of the **SPIH** problem.

Given a network $G = (V, E)$ shown in Fig.2, where $V = \{f_1, f_2, f_3, v_1, v_2, v_3, t_1, t_2, t_3\}$, $E = \{e_i | i = 1, 2, \ldots, 14\}$. Let $w = (4, 3, 6, 4, 5, 6, 5, 4, 8, 7, 6, 6, 4, 7)$, $l = (3, 2, 5, 3, 4, 3, 4, 3, 6, 4, 5, 4, 3, 4)$, the set of source-terminal pairs of vertices be $\{(f_k, t_k),\ k = 1, 2, 3\}$, and $d = (13, 14, 11)$. Let $c = (3.1, 2.1, 5.02, 1.1, 2.1, 1.01, 1.02, 3.02, 2.03, 4.1, 4.2, 2.1, 3.05, 5.1)$.

When calling Algorithm B for the instance given by Fig.2, we obtain an approximation solution $w^* = (4, 3, 5, 4, 5, 6, 5, 3, 8, 4, 6, 6, 4, 4)$ and its objective value is 17.24, where $w^*_i = \begin{cases} l_i, & \text{if } i = 3, 8, 10, 14, \\ w_i, & \text{otherwise.} \end{cases}$. Please see Table 1 for the details of iterations in Algorithm B.

The heuristic algorithm B is coded in Matlab 7.0 and run on a PC with intel core i7-3770, 3.4 GHz under Windows 7. We have tested the heuristic algorithm on ten classes of network configurations which differ from the number n of nodes, varying from 25 to 200, and the number r of source-terminal pairs, being 5, 15, 30 or 50. There are 20, 50 or 100

Table 1 Iteration process of Algorithm B.

IN	P_k	$l'(P_k)$	d_k	e_j	LW	IN	P_k	$l'(P_k)$	d_k	e_j	LW
1	$f_1 f_2 v_1 f_3 t_1$	13	13	e_1	Yes	6	$f_1 t_3 t_2 t_1$	13	13	e_4	Yes
	$f_2 v_1 v_3 t_2$	10	14				$f_2 v_1 v_3 t_2$	10	14		
	$f_3 v_1 v_2 t_3$	11	11				$f_3 v_1 v_3 t_3$	12	11		
2	$f_1 v_2 v_1 f_3 t_1$	13	13	e_{13}	Yes	7	$f_1 t_3 t_2 t_1$	13	13	e_{12}	Yes
	$f_2 v_1 v_3 t_2$	10	14				$f_2 v_1 v_3 t_2$	11	14		
	$f_3 v_1 v_2 t_3$	11	11				$f_3 t_1 t_2 t_3$	11	11		
3	$f_1 v_2 v_1 f_3 t_1$	13	13	e_2	Yes	8	$f_1 t_3 t_2 t_1$	13	13	e_{10}	No
	$f_2 v_1 v_3 t_2$	10	14				$f_2 f_1 t_3 t_2$	13	14		
	$f_3 t_1 t_2 t_3$	11	13				$f_3 t_1 t_2 t_3$	11	11		
4	$f_1 t_3 t_2 t_1$	13	13	e_3	No	9	$f_1 v_2 v_1 f_3 t_1$	14	13	e_{14}	No
	$f_2 v_1 v_3 t_2$	10	14				$f_2 f_1 t_3 t_2$	13	13		
	$f_3 t_1 t_2 t_3$	11	11				$f_3 t_1 t_2 t_3$	12	11		
5	$f_1 f_2 v_1 f_3 t_1$	14	13	e_8	No						
	$f_2 v_1 v_3 t_2$	10	14								
	$f_3 t_1 t_2 t_3$	11	11								

IN: The number of iterations; e_j: Edge obtained in Step 3;

$l'(P_k)$: The distance of the shortest path P_k under l'; LW: Can l'_j be changed to w_i?

random instances generated for each class of network configuration. In all instances of the configurations, the cost range of c_i is 0.5-1.5 for each edge e_i; the length range of w_i is 0-27 for each edge e_i; and the length range of lower bound l_i is 0-15. In order to avoid solving a big number of non-feasible instances, we assume that d_k is between $d_l(f_k, t_k)$ and $d_w(f_k, t_k)$ for each source-terminal pair (f_k, t_k).

Table 2 Computational results of Algorithm B and that of implicit enumeration algorithm.

NI	n	m	r	VC	VC1	TC	TC1
20	25	35	5	7.4	6.2	0.56	623.23
20	30	40	5	7.2	6.3	0.91	3783.3
5	35	45	5	8.3	7.5	1.44	18054.8
5	35	50	5	8.6	7.7	1.54	33156.2
100	35	50	15	10.6	-	1.89	-
100	45	70	15	11.9	-	4.38	-
100	70	100	15	14.6	-	16.7	-
50	100	150	15	18.6	-	49.8	-
50	150	200	15	28.4	-	191.6	-
50	150	200	50	37.1	-	249.3	-
50	200	250	30	33.2	-	606.7	-

NI: The number of instances; n: The number of nodes;
m: The number of edges; r: The number of source-terminal pairs;
VC: Average approximation objective value by using Algorithm B;
VC1: Average objective value by using implicit enumeration algorithm;
TC: Average running time in CPU-seconds by using Algorithm B;
TC1: Average running time in CPU-seconds by using implicit enumeration algorithm.

Computational results are shown in Table 2. It displays the average approximation objective values and the average CPU-time in seconds of the algorithm B using 20, 50 or 100 instances in each class of network configuration.

Table 2 shows that the average running time of Algorithm B is far less than that of implicit enumeration algorithm. If $m > 50$, then the running time of implicit enumeration algorithm is unbearable.

5 Conclusions

In this paper we studied the shortest path improvement problem under sum-type Hamming distance. We show that the **SPIH** problem is NP-hard to be approximated within a worst-case ratio of $O(\log |V|)$. We also proposed a greedy-type heuristic algorithm to solve the problem and numerical experiments show that the algorithm is effective.

It remains open if there are polynomial-time algorithms that can approximate the **SPIH** within a ratio $O(|V|^q)$ for some $q > 0$. It is also meaningful to consider some special cases in which the **SPIH** is solvable in polynomial time.

References

[1] D. Burton and P. Toint, On an instance of the inverse shortest path problem, *Mathematical Programming* 53 (1992) 45–61.

[2] C. Duin and A. Volgenant, Some inverse optimization problems under the Hamming distance, *European Journal of Operational Research*, 170 (2006) 887–899.

[3] X.C. Guan and J.Z. Zhang, Inverse Bottleneck Optimization Problems under Weighted Hamming Distance, *Lecture Notes in Computer Science* 4041 (2006) 220–230.

[4] Y. He, B.W. Zhang and E.Y. Yao, Weighted inverse minimum spanning tree problems under Hamming distance, *Journal of Combinatorial Optimization* 9 (2005) 91–100.

[5] C. Heuberger, Inverse optimization: a survey on problems, methods, and results, *Journal of Combinatorial Optimization* 8 (2004) 329–361.

[6] Y.W. Jiang, L.C. Liu, B. Wu and E.Y. Yao, Inverse minimum cost flow problems under the weighted Hamming distance, *European Journal of Operational Research* 207 (2010) 50–54.

[7] L.C. Liu and E. Y. Yao, Inverse min-max spanning tree problem under the weighted sum-type Hamming distance, *Theoretical Computer Science* 396 (2008) 28–34.

[8] C.H. Papadimitriou, Computational complexity, *Addison-Wesley*, Reading, MA, 1994.

[9] R. Raz and S. Safra, A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP, *Proc. 29th Ann. ACM Symp. on Theory of Computing,* ACM, 1997, pp. 475–484.

[10] B.W. Zhang, X.C. Guan, C.Y. He, S.G. Wang, Algorithms for the Shortest Path Improvement Problems under Unit Hamming Distance, *Journal of Applied Mathematics*, http://dx.doi.org/10.1155/2013/847317,2013.

[11] B.W. Zhang, J.Z. Zhang and Y. He, The center location improvement under Hamming distance, *Journal of Combinatorial Optimizaton* 9 (2005) 187–198.

[12] B.W. Zhang, J.Z. Zhang and Y. He, Constrained inverse minimum spanning tree problems under bottleneck-type Hamming distance, *Journal of Global Optimization* 34 (2006) 467–474.

[13] B. W. Zhang, J. Z. Zhang and L. Q. Qi, The shortest path improvement problem under Hamming distance, *Journal of Combinatorial optimization* 12 (2006) 351–361.

[14] J.Z. Zhang and Y.X. Lin, Computation of the reverse shortest-path problem, *Journal of Global Optimization* 25 (2003) 243–261.

[15] J.Z. Zhang, Z.F. Ma and C. Yang, A column generation method for inverse shortest path problems, *ZOR-Mathematical Methods of Operations Research* 41 (1995) 347–358.

Manuscript received 20 July 2014
revised 15 July 2015
accepted for publication 15 July 2015

Binwu Zhang
Department of Mathematics and Physics, Hohai University
Changzhou Campus, Changzhou 213022, China
E-mail address: bwzhang71@163.com

Xiucui Guan
Department of Mathematics, Southeast University
Nanjing 210096, China
E-mail address: xcguan@163.com

Qin Wang
Department of Mathematics, China Jiliang University
Hangzhou 310018, China
E-mail address: luckyqin@163.com

Chunyuan He
Department of Mathematics and Physics, Hohai University
Changzhou Campus, Changzhou 213022, China

Samson Hansen Sackey
College of Internet of Things, Hohai University
Changzhou Campus, Changzhou 213022, China
E-mail address: Samsonsackey@yeah.net