Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 847317, 8 pages
http://dx.doi.org/10.1155/2013/847317

Research Article

Algorithms for the Shortest Path Improvement Problems under Unit Hamming Distance

Bingwu Zhang,[1] Xiucui Guan,[2] Chunyuan He,[1] and Shuguo Wang[1]

[1] *Department of Mathematics and Physics, Hohai University, Changzhou 213022, China*
[2] *Department of Mathematics, Southeast University, Nanjing 210096, China*

Correspondence should be addressed to Bingwu Zhang; bwzhang71@163.com

Received 1 July 2013; Accepted 19 August 2013

Academic Editor: Frank Werner

In a shortest path improvement problem under unit Hamming distance (denoted by SPIUH), an edge weighted graph with a set of source-terminal pairs is given; we need to modify the lengths of edges by a minimum cost under unit Hamming distance such that the modified distances of the shortest paths are upper bounded by given values. The SPIUH problem on arborescent network is formulated as a 0-1 integer programming model. Some strongly polynomial time algorithms are designed for the problems on some special arborescent networks. Firstly, two greedy algorithms are proposed for problems on chain networks and special star-tree networks, respectively. Secondly, a strongly polynomial time algorithm is presented for the problem with a single source and constrained paths. Finally, a heuristic algorithm and its computational experiments are given for the SPIUH problem on general graphs.

1. Introduction

Due to theoretical and practical importance, the inverse shortest path problems and the shortest path improvement problems have been extensively studied in recent years. In such problems, an edge weighted graph with a set of source-terminal pairs is given; we need to modify the lengths of edges by a minimum cost under a given 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] solved the inverse shortest path problem under l_2 norm by the nonlinear programming technique. Zhang et al. [2] proposed a column generation method for the inverse shortest path problem under l_1 norm. Zhang and Lin [3] 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 of single source-terminal. Guan and Zhang [4, 5] studied network improvement problems for a tree-path system or a multicut by upgrading nodes in a directed tree. For detail of inverse optimization problems, the readers may refer to the survey paper by Heuberger [6].

Since the inverse optimization problems under Hamming distance were first studied by He et al. [7], it has been investigated by many researchers (see, e.g., Zhang et al. [8–10], Duin and Volgenant [11], Guan and Zhang [12], Liu and Yao [13], and Jiang et al. [14]). Zhang et al. [8] showed that the shortest path improvement problem under Hamming distance (denoted by SPIH) is strongly NP-hard on general graphs and is still NP-hard even if the network is a chain network. So it is meaningful to design polynomial time algorithms for some special cases of the problems and to propose approximation and/or heuristic algorithms for the SPIH and the shortest path improvement problem under unit Hamming distance (denoted by SPIUH) on general graphs. That is the objective of this paper.

Let $G = (V, E)$ be a connected undirected network, where V is the vertex set, $E = \{e_1, e_2, \ldots, e_m\}$ is the edge set. Let $w_i \geq 0$ be the length of edge e_i and l_i the lower bound on the modified length of e_i, where $i = 1, 2, \ldots, m$. Let $\{(s_k, t_k),\ k = 1, 2, \ldots, r\}$ be the set of source-terminal pairs of vertices. Denote by $d_w(u, v)$ the shortest distance connecting u to v

under the length vector w. Let d_k be the upper bound of the shortest distance connecting the source-terminal pair (s_k, t_k). The SPIUH problem is to find a new edge length vector w^* satisfying $l \le w^* \le w$ such that the modified shortest distance $d_{w^*}(s_k, t_k)$ is upper bounded by d_k, and the total edge modification cost is minimized under unit Hamming distance, which can be formulated as a mathematical model shown below:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} H\left(w_i^*, w_i\right) \\ \text{s.t.} \quad & d_{w^*}\left(s_k, t_k\right) \le d_k, \quad k = 1, 2, \ldots, r, \\ & l_i \le w_i^* \le w_i, \quad i = 1, 2, \ldots, m, \end{aligned} \tag{1}$$

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

$$H\left(w_i^*, w_i\right) = \begin{cases} 1, & \text{if } w_i^* \ne w_i, \\ 0, & \text{if } w_i^* = w_i. \end{cases} \tag{2}$$

Zhang et al. show in [8] that SPIH problem can be transformed into a 0-1 knapsack problem in the special case when the given network is a chain network and $r = 1$.

Lemma 1 (see [8]). *The SPIH problem is NP-hard even if the network is a chain network.*

The shortest path improvement problems under Hamming distance have practical background. For example, a large earthquake happened in some towns. The relief supplies 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 $G = (V, E)$ of towns, where $v \in V$ denotes a town, $e = (u, v) \in E$ denotes a road connecting u to v. Let w_i and l_i be the travelling time through the road e_i before/after e_i is repaired. Note that some points of the road e_i are damaged (rather than every point of the road being destroyed), so the repair time for the road e_i may be a fixed amount c_i instead of $c_i(w_i - l_i)$. We need to transport the relief goods from some supplier town s_k to some stricken town t_k within the stipulated time. Our objective is to design a repair scheme satisfying the above transportation requirement such that the total repair time of repaired roads is minimized, which is just the shortest path improvement problem under Hamming distance.

The rest of this paper is organized as follows. A 0-1 integer programming model of the SPIUH problem on arborescent networks is constructed in Section 2. Some polynomial time algorithms are proposed for some special cases of the SPIUH problem on arborescent networks in Section 3. A heuristic algorithm and its computational experiment of the SPIUH problem on general graphs are given in Section 4. Conclusion and further research are given in Section 5.

For convenience, we denote by P_k a path from s_k to t_k and by $w(P_k) = \sum_{e_i \in P_k} w_i$ the length of P_k under w. In Sections 2 and 3, we use the following useful notations. For the unique path P_k from s_k to t_k in G, if $e_i, e_j \in P_k$ and e_i appears before e_j in the path P_k, then we denote it by $e_j <_{P_k} e_i$.

2. The SPIUH Problems on Arborescent Networks

In this section, we construct a 0-1 integer programming model for the SPIUH problem on arborescent networks (denoted by SPIUH-AN).

The SPIUH-AN problem can be reformulated by the definition of P_k as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} H\left(w_i^*, w_i\right) \\ \text{s.t.} \quad & w^*\left(P_k\right) \le d_k, \quad k = 1, 2, \ldots, r, \\ & l_i \le w_i^* \le w_i, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{3}$$

We can check the feasibility of problem (3) based on the following lemma.

Lemma 2. *Problem* (3) *is feasible if and only if* $l(P_k) \le d_k$ *for each* $k = 1, 2, \ldots, r$.

Proof

Sufficiency. Suppose that $l(P_k) \le d_k$ for each $k = 1, 2, \ldots, r$. Denote by $\Gamma = \bigcup_{k=1}^{r} P_k$ the union of given paths. Define a vector $\overline{w} : E \to R^m$ as follows:

$$\overline{w}_j = \begin{cases} l_j, & \text{if } e_j \in \Gamma, \\ w_j, & \text{otherwise.} \end{cases} \tag{4}$$

It is easy to see that $l_i \le \overline{w}_i \le w_i$ for each $i = 1, 2, \ldots, m$. Furthermore, for each $k = 1, 2, \ldots, r$, we have

$$\overline{w}\left(P_k\right) = l\left(P_k\right) \le d_k, \tag{5}$$

and thus $\overline{w}$ is a feasible solution of problem (3).

Necessity. Suppose that $\overline{w}$ is a feasible solution of problem (3). Note that P_k is the unique path from s_k to t_k and $l \le \overline{w}$; then we have

$$l\left(P_k\right) \le \overline{w}\left(P_k\right) \le d_k, \quad k = 1, 2, \ldots, r. \tag{6}$$

□

Obviously, we have the following property on optimal solutions of (3).

Lemma 3. *If* w^* *is an optimal solution of problem* (3)*, then* $\overline{w}$ *defined below is also an optimal solution of problem* (3)*:*

$$\overline{w}_i = \begin{cases} l_i, & \text{if } w_i^* \ne w_i, \\ w_i, & \text{if } w_i^* = w_i. \end{cases} \tag{7}$$

By Lemma 3, the constraints $w^*(P_k) \le d_k$ $(k = 1, 2, \ldots, r)$ in problem (3) are equivalent to

$$\begin{aligned} & \sum_{e_i \in P_k} w_i \left(1 - H\left(w_i^*, w_i\right)\right) \\ & \quad + \sum_{e_i \in P_k} l_i H\left(w_i^*, w_i\right) \le d_k \quad (k = 1, 2, \ldots, r); \end{aligned} \tag{8}$$

that is,

$$\sum_{e_i \in P_k} (w_i - l_i)\left(1 - H\left(w_i^*, w_i\right)\right) \le d_k - \sum_{e_i \in P_k} l_i = d_k - l\left(P_k\right) \quad (k = 1, 2, \ldots, r). \tag{9}$$

For each edge e_i, if we define a 0-1 variable x_i as follows:

$$x_i = \begin{cases} 1, & \text{if the length of edge } e_i \text{ is not reduced,} \\ 0, & \text{if the length of edge } e_i \text{ is reduced,} \end{cases} \tag{10}$$

and let $w_i^0 = w_i - l_i$, $d_k^0 = d_k - l(P_k)$, then problem (3) is equivalent to solving the problem below:

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} (1 - x_i) \\ \text{s.t.} \quad & \sum_{e_i \in P_k} w_i^0 x_i \le d_k^0, \quad k = 1, 2, \ldots, r, \\ & x_i = 0, 1, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{11}$$

As a conclusion, we have the following theorem.

Theorem 4. *The SPIUH-AN problem is equivalent to solving the 0-1 integer programming problem* (11)*. An optimal solution w^* of the SPIUH-AN problem can be given by* (12)*, and the optimal objective value is $\sum_{i=1}^{m}(1 - x_i^*)$, where x^* is an optimal solution of problem* (11)*:*

$$w_i^* = \begin{cases} l_i, & \text{if } x_i^* = 0, \\ w_i, & \text{if } x_i^* = 1. \end{cases} \tag{12}$$

3. Some Special Cases of the SPIUH-AN Problem

In this section, we consider some special cases of SPIUH problem on arborescent networks. In Section 3.1, we design greedy algorithms for the SPIUH problems on chain networks and special star-tree networks. In Section 3.2, we present a strongly polynomial time algorithm for the SPIUH problem with a single source and a special constraint on paths.

3.1. The SPIUH Problem on Special Star-Tree Networks. In this subsection, we first study the SPIUH problem on a chain network, where G is a chain network and (s_1, t_1) is the only source-terminal pair. We design a greedy algorithm for the SPIUH problem on a chain network. The main idea is to reduce the length of the edge with the current largest value w_i^0 to l_i in each iteration.

Algorithm 5 (a greedy algorithm). Consider the following.

Input: A chain G with a source-terminal pair (s_1, t_1), two edge length vectors l, w and a value d_1.

Step 1: If $l(E) > d_1$, then output that the instance is infeasible, stop.

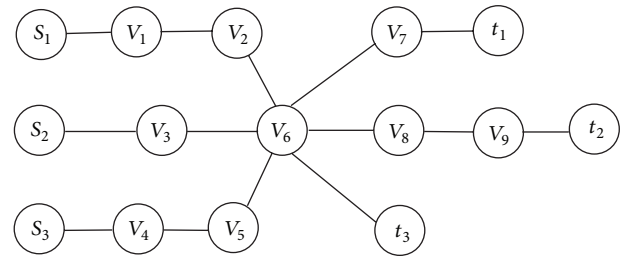

Figure 1: An example of a star tree for the SPIUH-ST problem.

Step 2: Let $w^0 = w - l$. Rearrange the values of w^0 in a non-increasing order, that is, $w_{j_1}^0 \ge w_{j_2}^0 \ge \cdots \ge w_{j_m}^0$.

Step 3: Let $p = 0$, and $W = w(P_1)$.

Step 4: While $W > d_1$ do

Put $p = p + 1$, and $W = W - w_{j_p}^0$.

Step 5: Output an optimal solution w^* defined below:

$$w_{j_q}^* = \begin{cases} l_{j_q}, & \text{if } 1 \le q \le p, \\ w_{j_q}, & \text{otherwise.} \end{cases} \tag{13}$$

It is easy to see that the main computation is to sort the values of w^0, and hence the time complexity of Algorithm 5 is $O(m \log m)$.

Next, we consider the SPIUH problem on a special star-tree network G, where any two source-terminal paths of G have no common edges. Such a problem is denoted by SPIUH-ST. See Figure 1 for example; there are three source-terminal paths (s_1, t_1), (s_2, t_2), and (s_3, t_3).

Obviously, the set of source-terminal paths is the union of edge-disjoint chains in the SPIUH-ST problem. Now we extend the greedy algorithm for SPIUH problem on a chain network to the SPIUH-ST problem.

Algorithm 6. Consider the following.

Input: A star-tree network G with a set $\{(s_k, t_k) \mid k = 1, 2, \ldots, r\}$ of source-terminal pairs, two edge length vectors l, w and a set $\{d_k \mid k = 1, 2, \ldots, r\}$ of values.

Step 1: Let $w^0 = w - l$ and $S = \emptyset$. Rearrange the values of w^0 in a non-increasing order, that is, $w_{j_1}^0 \ge w_{j_2}^0 \ge \cdots \ge w_{j_m}^0$.

Step 2: For $k = 1$ to r do

If $l(P_k) > d_k$, then output that the instance is infeasible, stop.

Else run Steps 3, 4, 5 in Algorithm 5 to solve the problem on the chain network P_k. Denote the optimal solution on P_k by $\{\overline{w}_i \mid e_i \in P_k\}$, and let $S = S \cup \{e_i \mid e_i \in P_k, \overline{w}_i = l_i, \text{ and } w_i^0 \ne 0\}$.

Step 3: Output the optimal solution w^* and the optimal objective value $|S|$ of the SPIUH-ST problem, where

$$w_i^* = \begin{cases} l_i, & \text{if } e_i \in S, \\ w_i, & \text{otherwise.} \end{cases} \tag{14}$$

Now we analyze the time complexity of Algorithm 6. Sorting the values of w^0 in Step 1 can be done in $O(m \log m)$ operations. In Step 2, there are r iterations and there are $O(|P_k|)$ operations in each iteration. Let $D = \max_{1 \le k \le r} |P_k|$. Then Algorithm 6 can be done in

$$\begin{aligned} O(m \log m + rD) &= O(m \log m + rm) \\ &= O(m(r + \log m)) \end{aligned} \tag{15}$$

operations. We sum up the above results in the following theorem.

Theorem 7. *The shortest path improvement problems on star-tree networks under unit Hamming distance can be solved by Algorithm 6 in $O(m(r + \log m))$ time, where r is the number of source-terminal pairs.*

Remark 8. There is a special case of the SPIUH-ST problem in which G is a star-tree network with a central vertex s_c such that there are no common edges on the paths from s_c to all leaves of the tree, and the central vertex s_c is the common source of all the source-terminal pairs; that is, $s_k = s_c$ for each $k = 1, 2, \ldots, r$. Hence, we only need to call Algorithm 5 in r times to solve the problem, and its time complexity is also $O(m(r + \log m))$.

3.2. The SPIUH Problem with a Single Source and Constrained Paths. In this subsection, we consider the SPIUH problem with a single source and a special constraint on paths, where $s_k = s$ for each $k = 1, 2, \ldots, r$, and the path P_k satisfies the following constraint:

$$\begin{aligned} &\text{if } e_i <_{P_k} e_q \text{ and } |Q_i| < |Q_q|, \\ &\text{then we have } w_i^0 \le w_q^0, \end{aligned} \tag{16}$$

where $Q_i = \{t_k \mid e_i \in P_k, k = 1, 2, \ldots, r\}$ is the set of terminals t_k of path P_k which passes through e_i. We call such a network an arborescent network with a single source and a special constraint on paths, which is denoted by SSCP. Such a problem is denoted by SPIUH-SSCP. The SPIUH problem with a single source and without the constraint condition (16) is denoted by SPIUH-SS.

For example, given is a network $G = (V, E)$ in Figure 2, where $V = \{s, v_1, v_2, t_1, t_2, t_3\}$, $E = \{e_i \mid i = 1, 2, \ldots, 5\}$, and $c_i = 1$ for each $i = 1, 2, \ldots, 5$. If $w = (2, 3, 2, 1, 2)$, $l = (0, 0, 0, 0, 0)$, and $d = (3, 3, 3)$ (see Figure 2(a)), then it is an instance of SPIUH-SSCP problem. However, if $w = (1, 1, 3, 3, 3)$, $l = (0, 0, 0, 0, 0)$, and $d = (3, 3, 3)$ (see Figure 2(b)), then we have $e_3 <_{P_1} e_1$ and $|Q_3| = 1 < |Q_1| = 3$, but $w_1^0 = 1 < w_3^0 = 3$. Hence it is an instance of SPIUH-SS problem, but not an instance of SPIUH-SSCP problem. Next we give a polynomial time algorithm to solve the SPIUH-SSCP problem.

Algorithm 9. Consider the following.

Input: An arborescent network G with a single source s and constrained paths P_k from s to t_k ($k = 1, 2, \ldots, r$), two edge length vectors l, w and a set $\{d_k \mid k = 1, 2, \ldots, r\}$ of values.

Step 1: Let $w^0 = w - l$ and $S = \emptyset$.

Step 2: For $k = 1$ to r do

If $l(P_k) > d_k$, then output that the instance is infeasible, stop.

Else let $d_k = d_k - w^0(P_k \cap S)$ and let $w_i' = l_i$ if $e_i \in P_k \cap S$ and $w_i' = w_i$ if $e_i \in P_k \setminus S$. Call Algorithm 5 to solve the problem on the chain network P_k with respect to the length vector w' and d_k. In Step 2 of Algorithm 5, we let $w^0 = w' - l$, then for all $e_i \in P_k$, rearrange the values $w_i^0 = w_i' - l_i$ in a non-increasing order, and if there are two values are equal (let's say w_p^0 and w_q^0), and e_p appears before e_q in the direct path P_k from s to t_k, then w_p^0 will always appear before w_q^0 in the non-increasing order. (This sorting method is called *stable sorting method*).

Denote by $\overline{w}$ the optimal solution to the problem on P_k. Put $S = S \cup \{e_j \mid e_j \in P_k, \overline{w}_j = l_j, \text{ and } w_i' \ne l_j\}$.

Step 3: Output the optimal solution w^* and optimal objective value $|S|$ of the SPIUH-SSCP problem, where

$$w_j^* = \begin{cases} l_j, & \text{if } e_j \in S, \\ w_j, & \text{otherwise.} \end{cases} \tag{17}$$

Lemma 10. *If the SPIUH-SSCP problem is feasible, then there is an optimal solution $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ of the SPIUH-SSCP problem satisfying the following.*

(1) *If there exists path P_k, $e_i \in P_k$ and $e_q \in P_k$, such that $e_i <_{P_k} e_q$, $0 < w_i^0 \le w_q^0$, and $\alpha_i = l_i$, then we have $\alpha_q = l_q$.*

(2) *If there exists path P_k, $e_i \in P_k$, and $e_q \in P_k$ such that $e_q <_{P_k} e_i$, $0 < w_i^0 < w_q^0$, and $\alpha_i = l_i$, then we have $\alpha_q = l_q$.*

Proof. Suppose $\beta = (\beta_1, \beta_2, \ldots, \beta_m)$ is an optimal solution of the SPIUH-SSCP problem. By Lemma 3, $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_m)$ defined below is also an optimal solution of the SPIUH-SSCP problem:

$$\gamma_j = \begin{cases} l_j, & \text{if } \beta_j \ne w_j, \\ w_j, & \text{if } \beta_j = w_j. \end{cases} \tag{18}$$

(1) If there exists a path P_k, $e_i \in P_k$ and $e_q \in P_k$, such that $e_i <_{P_k} e_q$, $0 < w_i^0 \le w_q^0$, $\gamma_i = l_i$, and $\gamma_q = w_q$, because

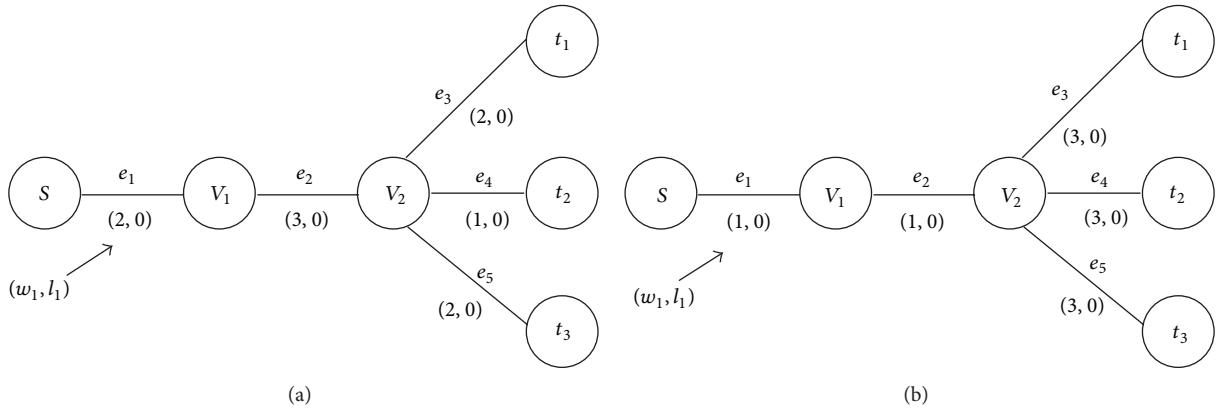

Figure 2: Examples of arborescent networks with a single source.

$|Q_q| \geq |Q_i|$ and reducing the weight of e_q first is better, or at least not worse, than reducing that of e_i, then we can get that α is also an optimal solution of the SPIUH-SSCP problem, where

$$\alpha_j = \begin{cases} w_i, & \text{if } j = i, \\ l_q, & \text{if } j = q, \\ \gamma_j, & \text{otherwise.} \end{cases} \tag{19}$$

(2) If there exists a path P_k, $e_i \in P_k$ and $e_q \in P_k$, such that $e_q <_{P_k} e_i$, $0 < w_i^0 < w_q^0$, $\gamma_i = l_i$, and $\gamma_q = w_q$, because by (16) we know that $|Q_i| = |Q_q|$ and reducing the weight of e_q first is better, or at least not worse, than reducing that of e_i, then we can get that α is also an optimal solution of the SPIUH-SSCP problem, where

$$\alpha_j = \begin{cases} w_i, & \text{if } j = i, \\ l_q, & \text{if } j = q, \\ \gamma_j, & \text{otherwise.} \end{cases} \tag{20}$$

Thus repeating the above operations, we can get an optimal solution α of the SPIUH-SSCP problem that satisfies the property in Lemma 10. □

Theorem 11. *If the SPIUH-SSCP problem is feasible, then Algorithm 9 outputs an optimal solution w^* of the SPIUH-SSCP problem in $O(m(r + \log m))$ time.*

Proof. If the SPIUH-SSCP problem is feasible, then it is not difficult to show that the solution w^* obtained by Algorithm 6 is a feasible solution of the SPIUH-SSCP problem.

Next we show that w^* obtained by Algorithm 9 satisfies properties (1) and (2) in Lemma 10. Note that we run Step 2 of Algorithm 5 for the path P_k by using stable sorting method in the kth iteration of for-loop in Algorithm 9 for any $0 \leq k \leq r$. When e_i is added to S in the kth iteration of for-loop in Algorithm 9, we consider the following two cases.

(1) If there is an e_q satisfying $e_i <_{P_k} e_q$ and $0 < w_i^0 \leq w_q^0$, then by the greedy property of Algorithm 9 (i.e., if $e_i <_{P_k} e_q$ and $0 < w_i^0 \leq w_q^0$, then e_q is added to S before e_i is added to S), we have $e_q \in S$. Hence, for any $w_i^* = l_i$ (i.e., $e_i \in S$), if $e_i <_{P_k} e_q$, $0 < w_i^0 \leq w_q^0$, then we have $w_q^* = l_q$ (i.e., $e_q \in S$).

(2) If there is an $e_q \in P_k$ such that $e_q <_{P_k} e_i$, $0 < w_i^0 < w_q^0$, then by the greedy property of Algorithm 9 (i.e., if $e_q <_{P_k} e_i$, $0 < w_i^0 < w_q^0$, then e_q is added to S before e_i is added to S), we have $e_q \in S$. Hence, we have $w_q^* = l_q$.

Hence, w^* obtained by Algorithm 9 is a feasible solution of the SPIUH-SSCP problem that satisfies properties (1) and (2) in Lemma 10.

Suppose that α is the optimal solution of the SPIUH-SSCP problem which satisfies the property (1) and (2) in Lemma 10. Now we show that for each $e_q \in E$, if $w_q^0 > 0$ and $w_q^* = l_q$, then $\alpha_q = l_q$. Suppose $w_q^* = l_q$ (i.e., $e_q \in S$); we consider the following three cases.

Case 1. If there exist a path P_k and an edge $e_i \in P_k$ such that $e_i <_{P_k} e_q$, $0 < w_i^0 \leq w_q^0$, and $\alpha_i = l_i$, then because the optimal solution α of the SPIUH-SSCP problem satisfies property (1) in Lemma 10, we can get that $\alpha_q = l_q$.

Case 2. If there exist a path P_k and an edge $e_i \in P_k$ such that $e_q <_{P_k} e_i$, $0 < w_i^0 < w_q^0$, and $\alpha_i = l_i$, then because the optimal solution α of the SPIUH-SSCP problem satisfies property (2) in Lemma 10, we can get that $\alpha_q = l_q$.

Case 3. If for each P_k and for each $e_i \in P_k$ such that (1) $e_i <_{P_k} e_q$, $0 < w_i^0 \leq w_q^0$ or (2) $e_q <_{P_k} e_i$, $0 < w_i^0 < w_q^0$, we have $\alpha_i = w_i$. Note that $w_q^* = l_q$ (i.e., $e_q \in S$); suppose e_q is added to S in the kth iteration of for-loop in Algorithm 9; let $\Omega = \{e_i \mid e_i \in P_k, 0 < w_q^0 < w_i^0\} \cup \{e_i \mid e_i \in P_k, e_q <_{P_k} e_i, 0 < w_q^0 = w_i^0\}$, $\Theta = P_k \setminus \Omega = \{e_q\} \cup \Gamma$, where $\Gamma = \{e_i \mid e_q <_{P_k} e_i, 0 \leq w_i^0 < w_q^0\} \cup \{e_i \mid e_i <_{P_k} e_q, 0 \leq w_i^0 \leq w_q^0\}$; then $\sum_{e_i \in \Omega} l_i + \sum_{e_i \in \Theta} w_i > d_k$ (otherwise, $e_q \notin S$). Note that we have the assumption that $\alpha_i = w_i$ for each $e_i \in \Gamma$, if $\alpha_q = w_q$; then

$$\alpha(P_k) = \sum_{e_i \in \Omega} \alpha_i + \sum_{e_i \in \Theta} \alpha_i$$

$$= \sum_{e_i \in \Omega} \alpha_i + \alpha_q + \sum_{e_i \in \Gamma} \alpha_i$$
$$= \sum_{e_i \in \Omega} \alpha_i + w_q + \sum_{e_i \in \Gamma} w_i$$
$$\geq \sum_{e_i \in \Omega} l_i + \sum_{e_i \in \Theta} w_i > d_k. \tag{21}$$

This contradicts that α is an optimal solution of the SPIUH-SSCP problem, so $\alpha_q = l_q$.

Hence, for each $e_q \in E$, if $w_q^0 > 0$ and $w_q^* = l_q$, then $\alpha_q = l_q$. That is, the objective value of w^* is not greater than the objective value of α. Note that α is an optimal solution of the SPIUH-SSCP problem, so w^* is an optimal solution of the SPIUH-SSCP problem. □

The time complexity analysis is similar to that of Theorem 7.

Example 12. As shown in Figure 2(a), let $w = (2, 3, 2, 1, 2)$, $l = (0, 0, 0, 0, 0)$, and $d = (3, 3, 3)$. The optimal solution of the SPIUH-SSCP problem generated by Algorithm 9 is $w^* = (0, 0, 2, 1, 2)$, and the optimal objective value is 2.

Remark 13. Note that Algorithm 9 cannot always obtain an optimal solution of the SPIUH-SS problem. For example, as shown in Figure 2(b), let $w = (1, 1, 3, 3, 3)$, $l = (0, 0, 0, 0, 0)$, and $d = (3, 3, 3)$. In this case, it is an instance of SPIUH-SS problem, but not an instance of SPIUH-SSCP problem. The solution generated by Algorithm 9 is $w^* = (1, 1, 0, 0, 0)$, whose objective value is 3. But the optimal solution is $\alpha = (0, 0, 3, 3, 3)$, and the optimal objective value is 2.

So it is meaningful to design efficient algorithms for the SPIUH-SS problem. We guess that the SPIUH-SS problem has strongly polynomial time algorithms.

4. A Heuristic Algorithm for the General SPIUH Problem

In this section, we design a heuristic algorithm to solve the general SPIUH problem (1) and give computational experiments to test effectiveness of the algorithm.

Lemma 14 (see [8]). *The SPIUH problem is strongly NP-hard even if $l_i = 0$ for each edge of the network.*

Similar to Lemma 2, we can use the next lemma to check the feasibility of SPIUH problem.

Lemma 15. *The SPIUH problem* (1) *is feasible if and only if $d_l(s_k, t_k) \leq d_k$, for each $k = 1, 2, \ldots, r$.*

We design a heuristic algorithm to solve the SPIUH problem. The main idea is as follows.

Suppose the SPIUH problem is feasible. Initialize $l' = l$. Define $K = \{k \mid 1 \leq k \leq r, d_w(s_k, t_k) > d_k\}$. First compute a shortest path P_k from s_k to t_k under l' for each $k \in K$. Let $X = \{e_i \mid e_i \in P_k,\ k \in K\}$ and $X' = \{e_i \mid e_i \in X, w_i - l_i' > 0\}$. Then find a specific edge $e_j \in X'$, and update $l_i' = w_j$ and $X' = X' - \{e_j\}$. Next we check if the SPIUH problem is feasible or not under the current weight 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 16. Consider the following.

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

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

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

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

> Find the edge e_j such that $j = \text{argmin}\{(w_i - l_i')|Q_i| \mid e_i \in X'\}$, and let $l_i' = w_j$ and $X' = X' \setminus \{e_j\}$.
>
> Compute a shortest path P_k from s_k to t_k under l' in the set Q_j of paths. Call the new path as P_k', if $l'(P_k') > d_k$, then $l_i' = l_j$ (and ignore P_k'); otherwise, let $P_k = P_k'$ (i.e., delete the previous P_k), $X = \{e_i \mid e_i \in P_k,\ k \in K\}$ and $Q_i = \{P_k \mid e_i \in P_k\}$ for each edge $e_i \in X$.

Step 4: Let $S = \{e_i \mid e_i \in X, l_i' = l_i, w_i \neq l_i\}$, output an approximation solution w^* and its objective value $|S|$ of the SPIUH problem, where

$$w_i^* = \begin{cases} l_i, & \text{if } e_i \in S, \\ w_i, & \text{otherwise.} \end{cases} \tag{22}$$

Now we analyze the time complexity of Algorithm 16. For each $k = 1, 2, \ldots, r$, there are $O(n^2)$ operations to compute a shortest path P_k; and hence there are $O(rn^2)$ operations in Step 1, where $n = |V|$. Step 2 requires $O(nm)$ operations. Step 3 requires $O(r^2n^3)$ operations. Furthermore, there are at most m iterations in Algorithm 16. Hence Algorithm 16 runs in $O(r^2n^3m)$ operations in the worst case, and it is a strongly polynomial time heuristic algorithm.

Next we first give an example to explain detailed computation process of the heuristic Algorithm 16, and then we present its computational experiments.

Table 1: Iteration process of Algorithm 16.

IN	P_k	$l'(P_k)$	d_k	e_j	LW	IN	P_k	$l'(P_k)$	d_k	e_j	LW
1	$s_1s_2v_1s_3t_1$	13	13								
	$s_2v_1v_3t_2$	10	14	e_1	Yes	6	$s_3t_1t_2t_3$	12	11	e_8	No
	$s_3v_1v_2t_3$	11	11								
2	$s_1v_2v_1s_3t_1$	13	13				$s_1t_3t_2t_1$	13	13		
	$s_2v_1v_3t_2$	10	14	e_2	Yes	7	$s_2s_1t_3t_2$	13	14	e_{12}	Yes
	$s_3v_1v_2t_3$	11	11				$s_3t_1t_2t_3$	11	11		
3	$s_1t_3t_2t_1$	14	13	e_3	No	8	$s_1t_3t_2t_1$	15	13	e_{10}	No
4	$s_1t_3t_2t_1$	13	13								
	$s_2v_1v_3t_2$	11	14	e_4	Yes	9	$s_1t_3t_2t_1$	15	13	e_{14}	No
	$s_3v_1v_2t_3$	11	11								
5	$s_1t_3t_2t_1$	13	13								
	$s_2v_1v_3t_2$	11	14	e_5	Yes						
	$s_3v_1v_2t_3$	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?

Table 2: Computational results of the Algorithm 16 and that of implicit enumeration algorithm.

NI	n	m	r	VC	VC1	TC	TC1
100	25	35	5	7.5	5.2	0.29	1068.61
100	30	40	5	7.4	5.3	0.53	4687.8
100	35	45	5	7.2	5.7	0.78	8512.14
50	35	50	5	7.6	5.6	0.9	11360.04
50	40	50	10	8.7	—	1.29	—
50	45	70	10	10.2	—	2.6	—
50	100	130	15	18.4	—	49.4	—
50	100	150	15	18.1	—	283.4	—
50	150	200	15	39.9	—	296	—
50	200	250	30	52.4	—	789.8	—

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 16; VC1: average objective value by using implicit enumeration algorithm; TC: average running time in CPU-seconds by using Algorithm 16; TC1: average running time in CPU-seconds by using implicit enumeration algorithm.

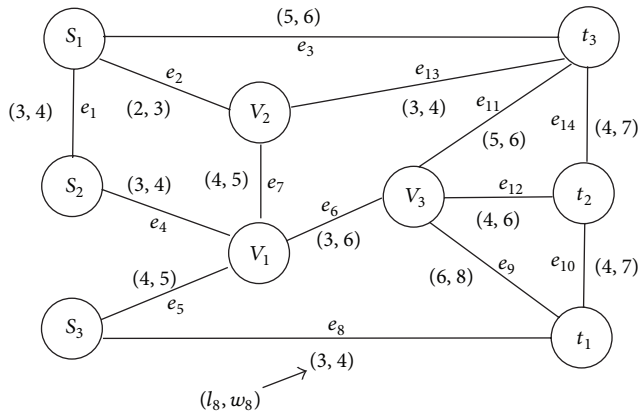

Figure 3: An example of the SPIUH problem.

Given is a network $G = (V, E)$ shown in Figure 3, where $V = \{s_1, s_2, s_3, v_1, v_2, v_3, t_1, t_2, t_3\}$, $E = \{e_i \mid i = 1, 2, \ldots, 14\}$. 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 is $\{(s_k, t_k),\ k = 1, 2, 3\}$, and $d = (13, 14, 11)$.

When calling Algorithm 16 for the instance given by Figure 3, 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 $|S| = 4$, where

$$w_i^* = \begin{cases} l_i, & \text{if } e_i \in S, \\ w_i, & \text{otherwise.} \end{cases} \tag{23}$$

See Table 1 for the details of iterations in running Algorithm 16.

Note that when Algorithm 16 is applied to solve the SPIUH-SSCP problem, similar to the proof of Theorem 11, it is not difficult to show that Algorithm 16 can obtain an optimal solution of the SPIUH-SSCP problem.

The heuristic Algorithm 16 is coded in Matlab 7.0 and run on a PC Pentium D, 2.8 GHz, under Windows XP. We have tested the heuristic algorithm on ten classes of network configurations which differ from the number n of nodes, varying from 30 to 100, and the number r of source-terminal pairs, being 5 or 15. There are 50 or 100 random

instances generated for each class of network configuration. In all instances of the configurations, the length range of w_i is 0–37 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 nonfeasible instances, we assume that d_k is between $d_l(s_k, t_k)$ and $d_w(s_k, t_k)$ for each source-terminal pair (s_k, t_k).

Computational results are shown in Table 2. It displays the average approximation objective values and the average CPU-time in seconds of Algorithm 16 and implicit enumeration algorithm by using 50 or 100 instances in each class of network configuration. Table 2 shows that the average running time of Algorithm 16 is far less than that of implicit enumeration algorithm, and if $m \geq 50$ and optimal objective value is greater than 5, then the running time of implicit enumeration algorithm is unbearable.

5. Conclusion and Further Research

The shortest path improvement problems under unit Hamming distance are studied. Firstly, the problem on arborescent networks is formulated as a 0-1 integer programming model. Secondly, two greedy algorithms are proposed for problems on chain networks and special star-tree networks. Thirdly, a strongly polynomial time algorithm is designed for the problem on arborescent networks with a single source and a special constraint on paths. Finally, a heuristic algorithm and corresponding computational experiments are presented for the SPIUH problem on general graphs.

For further research, it is meaningful to design exact algorithms or approximate algorithms for the shortest path improvement problems on arborescent networks with a single source under unit Hamming distance and to propose approximate or heuristic algorithms for the shortest path improvement problems under general Hamming distance.

Acknowledgments

The authors would like to thank the anonymous referees for their valuable comments and suggestions. This research is supported by NSFC (10801031), the project sponsored by SRF for ROCS, SEM (BZX/11H002) and Chinese Universities Scientific Fund (2012B14114, 2013B10014).

References

[1] D. Burton and Ph. L. Toint, "On an instance of the inverse shortest paths problem," *Mathematical Programming*, vol. 53, no. 1, pp. 45–61, 1992.

[2] J. Z. Zhang, Z. F. Ma, and C. Yang, "A column generation method for inverse shortest path problems," *Zeitschrift für Operations Research. Mathematical Methods of Operations Research*, vol. 41, no. 3, pp. 347–358, 1995.

[3] J. Z. Zhang and Y. X. Lin, "Computation of the reverse shortest-path problem," *Journal of Global Optimization*, vol. 25, no. 3, pp. 243–261, 2003.

[4] X. C. Guan and J. Z. Zhang, "A class of node based bottleneck improvement problems," *European Journal of Operational Research*, vol. 174, no. 3, pp. 1540–1552, 2006.

[5] X. C. Guan and J. Z. Zhang, "Improving multicut in directed trees by upgrading nodes," *European Journal of Operational Research*, vol. 183, no. 3, pp. 971–980, 2007.

[6] C. Heuberger, "Inverse combinatorial optimization: a survey on problems, methods, and results," *Journal of Combinatorial Optimization*, vol. 8, no. 3, pp. 329–361, 2004.

[7] Y. He, B. W. Zhang, and E. Y. Yao, "Weighted inverse minimum spanning tree problems under Hamming distance," *Journal of Combinatorial Optimization*, vol. 9, no. 1, pp. 91–100, 2005.

[8] B. W. Zhang, J. Z. Zhang, and L. Q. Qi, "The shortest path improvement problems under Hamming distance," *Journal of Combinatorial Optimization*, vol. 12, no. 4, pp. 351–361, 2006.

[9] B. W. Zhang, J. Z. Zhang, and Y. He, "The center location improvement problem under the Hamming distance," *Journal of Combinatorial Optimization*, vol. 9, no. 2, pp. 187–198, 2005.

[10] B. W. Zhang, J. Z. Zhang, and Y. He, "Constrained inverse minimum spanning tree problems under the bottleneck-type Hamming distance," *Journal of Global Optimization*, vol. 34, no. 3, pp. 467–474, 2006.

[11] C. W. Duin and A. Volgenant, "Some inverse optimization problems under the Hamming distance," *European Journal of Operational Research*, vol. 170, no. 3, pp. 887–899, 2006.

[12] X. C. Guan and J. Z. Zhang, "Inverse bottleneck optimization problems on networks," in *Algorithmic Aspects in Information and Management*, vol. 4041 of *Lecture Notes in Computer Science*, pp. 220–230, 2006.

[13] L. C. Liu and E. Y. Yao, "Inverse min-max spanning tree problem under the weighted sum-type Hamming distance," *Theoretical Computer Science*, vol. 396, no. 1–3, pp. 28–34, 2008.

[14] 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*, vol. 207, no. 1, pp. 50–54, 2010.