Yugoslav Journal of Operations Research
27 (2017), Number 1, 46–60
DOI: 10.2298/YJOR150624009T

EFFICIENT ALGORITHMS FOR THE REVERSE SHORTEST PATH PROBLEM ON TREES UNDER THE HAMMING DISTANCE

Javad TAYYEBI*
Department of Industrial Engineering, Birjand University of Technology, Birjand, Iran
javadtayyebi@birjandut.ac.ir, javadtayyebi@birjand.ac.ir

Massoud AMAN
Department of Mathematics, Faculty of Mathematical Sciences and Statistic, University of Birjand, Birjand, Iran
mamann@birjand.ac.ir

Received: June 2015 / Accepted: May 2016

Abstract: Given a network $G(V, A, \mathbf{c})$ and a collection of origin-destination pairs with prescribed values, the reverse shortest path problem is to modify the arc length vector $\mathbf{c}$ as little as possible under some bound constraints such that the shortest distance between each origin-destination pair is upper bounded by the corresponding prescribed value. It is known that the reverse shortest path problem is NP-hard even on trees when the arc length modifications are measured by the weighted sum-type Hamming distance. In this paper, we consider two special cases of this problem which are polynomially solvable. The first is the case with uniform lengths. It is shown that this case transforms to a minimum cost flow problem on an auxiliary network. An efficient algorithm is also proposed for solving this case under the unit sum-type Hamming distance. The second case considered is the problem without bound constraints. It is shown that this case is reduced to a minimum cut problem on a tree-like network. Therefore, both cases studied can be solved in strongly polynomial time.

Keywords: Inverse problems, Shortest path problem, Hamming distance, Combinatorial algorithms.

MSC: 90C27, 90C35, 05C85.

* Corresponding author

1. INTRODUCTION

Suppose that $G(V, A, \mathbf{c})$ is a directed and connected network where V is the set of nodes, A is the set of arcs and $\mathbf{c}$ is the nonnegative arc length vector. Let $S \subseteq V \times V$ be a collection of origin-destination pairs (s_i, t_i), $i = 1, \ldots, k$. The well-known shortest path problem is to find the shortest paths between each $(s_i, t_i) \in S$ on G. This problem has a wide variety of applications in practice such as transportation systems, computer networks. It is also theoretically interested because there exists several efficient algorithms to solve the problem and furthermore, the problem arises frequently as a subproblem when solving many combinatorial and network optimization problems [1].

For any optimization problem, one can introduce some inverse problems. We refer readers to the survey paper [11] for more details on inverse optimization problems. Two general classes of the inverse shortest path problems are studied in the literature: the inverse shortest path (ISP) problem and the reverse shortest path (RSP) problem. The ISP problem is to adjust the arc lengths minimally such that each given path from s_i to t_i becomes a shortest path. The RSP problem consists of modifying the arc lengths as little as possible so that the shortest distance between each pair (s_i, t_i) is upper bounded by a prescribed value $d_i \geq 0$. The RSP problem is also called as "the shortest path improvement problem" and "the inverse shortest path lengths problem" by some authors [7, 16]. The inverse and reverse shortest path problems have attracted great attention due to its broad applications in practice such as the traffic modeling, the seismic tomography, and the design of computer networks [3, 4, 6, 10].

Zhang et. al [19] showed that the ISP problem is equivalent to solving a minimum-weight circulation problem when the modifications are measured by the l_1 norm. In [18], a column generation scheme is developed to solve the ISP problem under the l_1 norm. Ahuja and Orlin [2] showed that the ISP problem under the l_1 norm can be solved by solving a new shortest path problem. For the l_∞ norm, they showed that the problem reduces to a minimum mean cycle problem. In [15], it is shown that all feasible solutions of the ISP problem form a polyhedral cone and the relationship between this problem and the minimum cut problem is discussed. Duin and Volgenant [8] proposed an efficient algorithm based on the binary search technique to solve the ISP problem under the bottleneck-type Hamming distance. In [12, 13], we extended their method to solve the inverse minimum cost flow problem and the inverse linear programming problem.

The concept of the inverse optimization problems was introduced by Burton and Toint [3, 4]. They studied the RSP problem under the l_2 norm and proposed a method based on nonlinear optimization techniques to solve the problem. In [5], it is shown that the RSP problem under the l_2 norm is NP-hard. A similar result is obtained for the l_1 norm in [7, 20] and also, efficient algorithms are proposed in some special cases. Fekete et. al [9] studied the complexity of obtaining a feasible solution to the RSP problem. They showed that it is intractable even in very restricted cases.

The RSP problem under the weighted sum-type Hamming distance is formulated

as follows [16]:

$$\begin{aligned} \min \quad & z = \sum_{e_j \in A} w_j H(c_j, \hat{c}_j) \\ s.t. \quad & d_{\hat{\mathbf{c}}}(s_i, t_i) \leq d_i \quad \forall (s_i, t_i) \in S, \\ & 0 \leq \hat{c}_j \leq c_j \quad \forall e_j \in A, \end{aligned} \tag{1}$$

where $d_{\hat{\mathbf{c}}}(s_i, t_i)$ is the length of the shortest path from s_i to t_i with respect to the length vector $\hat{\mathbf{c}}$; $H(c_j, \hat{c}_j)$ is the Hamming distance between c_j and $\hat{c}_j$ (i.e., $H(c_j, \hat{c}_j) = 0$ if $\hat{c}_j = c_j$ and $H(c_j, \hat{c}_j) = 1$, otherwise); $w_j \geq 0$ is a penalty for modifying the length of each arc e_j and $d_i \geq 0$ is the prescribed distance for each origin-destination pair (s_i, t_i).
Two cases of the problem (1) are proved to be NP-hard in [16]:

- The RSP problem with a single origin in which $c_j = w_j = 1$ for every $e_j \in A$: We call this problem as the RSP (URSP) problem with uniform data.
- The RSP problem on trees with a single origin: This problem is referred to as the RSP (TRSP) problem on trees.

The second result is interested because the RSP problem on trees under the l_1 norm is polynomially solvable [20]. This is a reason why the behaviour of the RSP problem under the Hamming distances is different from that of the problem under norms. An alternative reason is that the Hamming distances unlike to norms are nonconvex and discontinuous. Therefore, the known approaches for norms cannot be applied for solving the RSP problem under the Hamming distances.
As the RSP problem on trees under the sum-type Hamming distance is NP-hard, this aspect is worthwhile to further study special cases which the problem is polynomially solvable. In [17], the authors proposed polynomial-time algorithms to solve the problem under the unit Hamming distance on chain networks and star-tree networks. Then, they designed an efficient algorithm for solving the TRSP problem with a very special constraint on paths.
In this article, we first consider the TRSP problem when $c_j = 1$ for each $e_j \in A$. It is shown that the problem transforms to an instance of the minimum cost flow problem due to the special form of its coefficient matrix. As a special case, we propose a simple algorithm to solve the problem that is in the intersection of the ULRSP and TRSP problems. We also consider the TRSP problem without nonnegativity constraints and show that the problem can be reduced to solving a minimum cut problem on a tree-like network. Figure 1 provides a diagram to compare the complexities of these problems.

The rest of this article is organized as follows: Section 2 focuses on the TRSP problem with uniform lengths. Section 3 considers the TRSP problem with uniform penalties and uniform lengths. Section 4 discusses the RSP problem without nonnegativity restrictions. Section 5 presents some computational experiments of our proposed algorithms. Finally, some concluding remarks are given in Section 6.

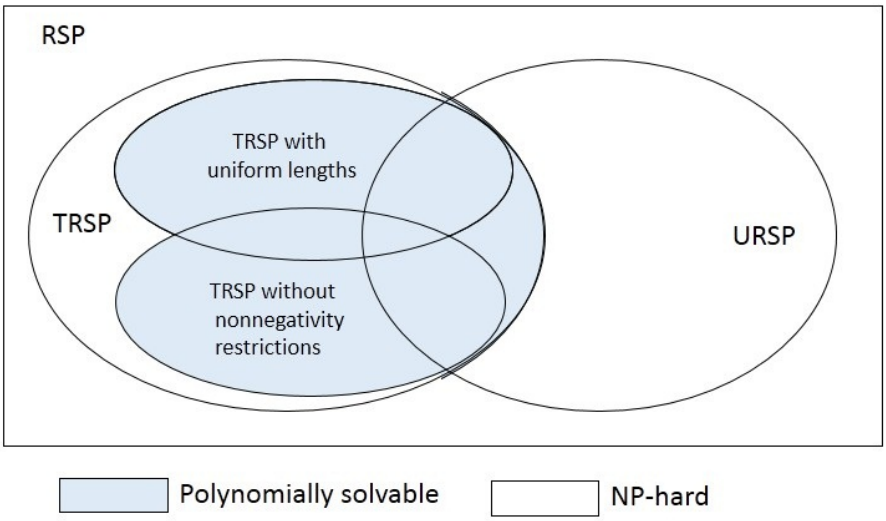

Figure 1: Complexity of the RSP problem

2. REVERSE SHORTEST PATH PROBLEM ON TREES WITH UNIFORM LENGTHS

Suppose that $T(V, A)$ is a tree where V is the set of n nodes, $A = \{e_1, e_2, \ldots, e_{n-1}\}$ is the set of arcs. In this section, we consider the TRSP problem when all arc lengths are equal to 1. The problem contains a single origin s and multiple destinations t_i, $i = 1, 2, \ldots, k$. We assume that T is a directed out-tree rooted at node s, i.e., the unique path in the tree from node s to every other node is a directed path. This problem can be formulated as follows:

$$\min \quad z = \sum_{e_j \in A} w_j H(1, \hat{c}_j) \tag{2a}$$

$$s.t. \quad \sum_{e_j \in P_{t_i}} \hat{c}_j \leq d_i \quad \forall i = 1, \ldots, k, \tag{2b}$$

$$0 \leq \hat{c}_j \leq 1 \quad \forall e_j \in A, \tag{2c}$$

where P_{t_i} is the unique path from s to each destination node t_i and the other parameters are defined as in the problem (1). Note that the problem (2) is feasible because $\hat{\mathbf{c}} = \mathbf{0}$ is a feasible solution to the problem.
In T, we denote the head node of each $e_j \in A$ as v_{e_j}. For every arc $e_j \in A$, we denote the subtree of T rooted at v_{e_j} as T_{e_j} and the unique path from s to v_{e_j} by P_{e_j}.

Lemma 2.1. *The problem (2) has a zero-one optimal solution.*

Proof. Since the set of objective values of the problem is finite, the problem has at least one optimal solution $\hat{c}^*$. Define the solution $\hat{\mathbf{c}}^{**}$ as follows:

$$\hat{c}_j^{**} = \begin{cases} 1 & \hat{c}_j^* = 1, \\ 0 & \hat{c}_j^* \neq 1, \end{cases} \quad \forall e_j \in A.$$

We show that $\hat{\mathbf{c}}^{**}$ is also optimal to the problem (2). It is obvious that the objective values of both the solutions are the same because $\hat{c}_j^* \neq 1$ if and only if $\hat{c}_j^{**} \neq 1$ for each $e_j \in A$. On the other hand, $\hat{\mathbf{c}}^{**} \leq \hat{\mathbf{c}}^*$ based on the definition of $\hat{\mathbf{c}}^{**}$. This guarantees that $\hat{\mathbf{c}}^{**}$ is also feasible to the problem because

$$\sum_{e_j \in P_{t_i}} \hat{c}_j^{**} \leq \sum_{e_j \in P_{t_i}} \hat{c}_j^* \leq d_i \quad \forall i = 1, 2, \ldots, k.$$

Therefore, the problem has the zero-one optimal solution $\hat{\mathbf{c}}^{**}$. □

By using Lemma 2.1, we can restrict our attention to zero-one solutions. Therefore, the problem is converted into the following problem:

$$\begin{aligned} \min \quad & z = \sum_{e_j \in A} w_j(1 - \hat{c}_j) = \sum_{e_j \in A} w_j - \sum_{e_j \in A} w_j \hat{c}_j \\ s.t. \quad & \sum_{e_j \in P_{t_i}} \hat{c}_j \leq d_i \quad \forall i = 1, \ldots, k, \\ & \hat{c}_j = 0 \text{ or } 1 \quad \forall e_j \in A, \end{aligned}$$

In matrix form, this problem is rewritten as

$$\min \quad z = \mathbf{w}\hat{\mathbf{c}} \tag{3a}$$

$$s.t. \quad B\hat{\mathbf{c}} \leq \mathbf{d}, \tag{3b}$$

$$\hat{\mathbf{c}} \in \{0, 1\}^{n-1}, \tag{3c}$$

where $\mathbf{w} = [-w_1, -w_2, \ldots, -w_{n-1}]$, $\mathbf{d} = [d_1, d_2, \ldots, d_k]^T$, $\hat{\mathbf{c}} = [\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_{n-1}]^T$ and $B = (\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{n-1})$ is a $k \times (n-1)$ matrix whose jth column is defined as follows:

$$b_{ij} = \begin{cases} 1 & e_j \in P_{t_i}, \\ 0 & e_j \notin P_{t_i}, \end{cases} \quad \forall i = 1, 2, \ldots, k.$$

As an immediate consequence, we can assume that the prescribed values d_i are integral because the left-hand side of each constraint (3b) is an integer between 0 and $[d_i]$ for each zero-one feasible solution $\hat{\mathbf{c}}$.

We next show that 1's of each column of the coefficient matrix B are consecutive if its rows are arranged in a special fashion. This result guarantees that the problem can transform to an instance of the minimum cost flow problem on an auxiliary network.

Since each row of B corresponds to a destination node, it is sufficient to sort the destination nodes. The Depth-First-Search (DFS) algorithm is used to traverse nodes of T starting from the origin node s. Assume that $D = \{t_1, t_2, \ldots, t_k\}$ is the set of destinations sorted in the order of their traversal by the DFS algorithm.

Theorem 2.2. *1's of each column of the matrix B are consecutive whenever the constraints (3b) are arranged in the order of D.*

Proof. For each column $\mathbf{b_j}$, $b_{ij} = 1$ if $e_j \in P_{t_i}$. Equivalently, $b_{ij} = 1$ if $t_i \in T_{e_j}$. Since nodes of T_{e_j} are visited consecutively by the DFS algorithm, it follows that 1's of $\mathbf{b}_j$ are consecutive. □

Based on Theorem 2.2, we can transform the problem (3) into a minimum cost flow problem due to the special form of the coefficient matrix [1]. Let the constraints (3b) be arranged in the order of D. We first introduce a slack variable s_i for each constraint (3b) to bring it into an equality form and also, add a redundant equality constraint $\mathbf{0}.\hat{\mathbf{c}} = 0$. Therefore, the problem has $k+1$ equality constraints. We next subtract the ith constraint from the $(i+1)$th constraint for each $i = 1, 2, \ldots, k$. Suppose that B' and $\mathbf{d}'$ are the new coefficient matrix and the new right hand vector, respectively. Theorem 2.2 implies that B' is a matrix whose each column has exactly one +1 and exactly one −1 (i.e., B' is an incidence matrix). On the other hand, it can easily seen that $\sum_{i=1}^{k+1} d'_i = 0$. Therefore, the problem (3) is converted into the following equivalent problem:

$$\begin{aligned} \min \quad & z = \mathbf{w}\hat{\mathbf{c}} \\ s.t. \quad & B' \begin{bmatrix} \hat{\mathbf{c}} \\ \mathbf{s} \end{bmatrix} = \mathbf{d}', \\ & \hat{c}_j = 0 \text{ or } 1 \quad \forall e_j \in A, \\ & s_i \geq 0 \quad \forall i \in \{1, 2, \ldots, k\}, \end{aligned} \tag{4}$$

Note that the relaxation $0 \leq \hat{\mathbf{c}} \leq 1$ converts the problem into a minimum cost flow problem because B' is an incidence matrix. Since each minimum cost flow problem with integral data has at least an optimal integer solution and most minimum cost flow algorithms obtain such the optimal solutions [1], we can use this relaxation. Therefore, the problem (3) reduces to an instance of the minimum cost flow problem on the auxiliary network $G'(V', A', \mathbf{w}, \mathbf{d}')$ which is defined as follows:

- The node set is $V' = \{t_1, t_2, \ldots, t_k, t_{k+1}\}$ where t_{k+1} corresponds to the redundant equality constraint $\mathbf{0}.\hat{\mathbf{c}} = 0$.
- The arc set is $A' = A \cup \{s_1, \ldots, s_k\}$. Each arc $e_j \in A$ emanates from t_u and terminates at t_v where u is the row corresponding to the first 1 of $\mathbf{b}_j$ and v is the row corresponding to the first 0 of $\mathbf{b}_j$ after u. Each arc s_i emanates from t_i and terminates at t_{i+1}.
- The cost of each arc e_j is $-w_j$ and the cost of each arc s_i is zero.
- The upper bound of each arc e_j is 1 and each arc s_i has an infinite upper bound.
- Node t_i has a supply or demand equal to $d'_i = d_i - d_{i-1}$ for each $i \in \{2, 3, \ldots, k+1\}$ and the supply of t_1 is $d_1 \geq 0$.

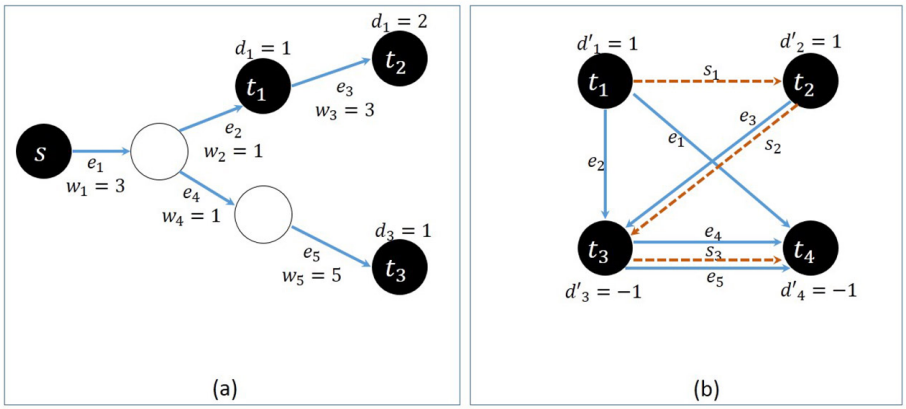

Figure 2: (a) A network $G(V, A, \mathbf{w})$; (b) the auxiliary network $G'(V', A', \mathbf{w}, \mathbf{d}')$

We illustrate this transformation using an example.

Example 2.3. *Consider an instance of the problem (2) defined on the network shown in Figure 2.a. The coefficient matrix B is as follows:*

$$B = \begin{array}{c|ccccc} & \hat{c}_1 & \hat{c}_2 & \hat{c}_3 & \hat{c}_4 & \hat{c}_5 \\ t_1 & 1 & 1 & 0 & 0 & 0 \\ t_2 & 1 & 1 & 1 & 0 & 0 \\ t_3 & 1 & 0 & 0 & 1 & 1 \end{array}.$$

Note that rows of B are sorted by the DFS algorithm and hence, Theorem 2.2 holds. The new coefficient matrix is

$$B' = \begin{array}{c|cccccccc} & \hat{c}_1 & \hat{c}_2 & \hat{c}_3 & \hat{c}_4 & \hat{c}_5 & s_1 & s_2 & s_3 \\ t_1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ t_2 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 \\ t_3 & 0 & -1 & -1 & 1 & 1 & 0 & -1 & 1 \\ t_4 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & -1 \end{array}.$$

The auxiliary network G' is shown in Figure 2.b. By solving the minimum cost flow problem defined on G', we obtain the optimal solution

$$\hat{c}_2 = \hat{c}_3 = \hat{c}_5 = 1, \hat{c}_1 = \hat{c}_4 = s_1 = s_2 = s_3 = 0, z = -1 - 3 - 5 = -9.$$

Therefore, the optimal solution of the problem (2) is to change only the cost of e_1 and e_4 from 1 to 0. Its objective value is $4 = \sum_{j \in e_j} w_j - \sum w_j \hat{c}_j = 13 - 9$.

The auxiliary network contains $k + 1$ nodes and $n + k - 1$ arcs. One can use the enhanced capacity scaling algorithm to solve the associated minimum cost flow problem in $O(n^2 \log k + nk \log^2 k)$ time [1]. Therefore, the problem (2) can be solved in the same time.

Remark 2.4. *In the problem (2), we have assume that $c_j = 1$ for each $e_j \in A$. It is easy to see that this transformation is valid for the problem with uniform lengths, and not necessarily unit lengths.*

3. REVERSE SHORTEST PATH PROBLEM ON TREES WITH UNIFORM PENALTIES AND UNIFORM LENGTHS

In this section, we consider the TRSP problem when $w_j = c_j = 1$ for every $e_j \in A$. The TRSP problem with uniform penalties and uniform lengths is simply convertible to this special case. Note that this problem is a special type of the problem (2) with unit penalties and consequently, it can be solved efficiently by the transformation stated in Section 2. However, we here present a simple algorithm to solve the problem with a better complexity. Similar to the problem (2), we can formulate the TRSP problem with unit penalties as well as unit lengths as follows:

$$\begin{aligned} \min \quad & z = \sum_{e_j \in A} H(1, \hat{c}_j) \\ s.t. \quad & \sum_{e_j \in P_{t_i}} \hat{c}_j \le d_i \quad \forall i = 1, \dots, k, \\ & 0 \le \hat{c}_j \le 1 \quad \forall e_j \in A, \end{aligned} \tag{5}$$

where the parameters are defined as in the problems (1) and (2).
Based on Lemma 2.1, we restrict our attention to solutions $\hat{\mathbf{c}}$ with $\hat{c}_j = 0$ or 1 for every $e_j \in A$. Any such solution is determined by an arc set $S \subseteq A$ as follows:

$$\hat{c}_j = \begin{cases} 0 & e_j \in S, \\ 1 & e_j \in A \backslash S, \end{cases} \qquad \forall e_j \in A. \tag{6}$$

A solution $\hat{\mathbf{c}}$ corresponding to S has the objective value equal to $|S|$. Our proposed algorithm begins with $S = \emptyset$ and checks whether or not the corresponding solution is feasible. If the solution is feasible, the algorithm terminates and otherwise, it goes to the next iteration by adding one arc e_q to S. This process is repeated until a feasible solution is obtained.
For a given set S, if the value

$$d'_i = \sum_{e_j \in P_{t_i}} \hat{c}_j - d_i = \Big(\sum_{e_j \in P_{t_i} \cap S} 0 + \sum_{e_j \in P_{t_i} \backslash S} 1 \Big) - d_i = |P_{t_i} \backslash S| - d_i$$

is nonpositive for each $i = 1, 2, \dots, k$, then the solution $\hat{\mathbf{c}}$ corresponding to S is feasible to the problem (5). In other words, the solution corresponding to S is feasible if S contains at least $|P_{t_i}| - d_i$ elements of P_{t_i} for each $i = 1, 2, \dots, k$. Therefore, the problem (5) is reduced to the following problem:

$$\begin{aligned} & \min_{S \subseteq A} |S| \\ & \text{s.t. } d'_i \le 0, \qquad \forall i = 1, 2, \dots, k. \end{aligned} \tag{7}$$

For obtaining the optimal solution S to the problem (7), we put arcs closer to the origin s into S because such arcs belong to more sets P_{t_i}. For this purpose, we use the Breadth-First-Search algorithm to traverse arcs starting from the origin node s (see Algorithm 1).
The set S obtained by Algorithm 1 has obviously the following property: if e_j

Input: A tree $T(V, A)$, an origin node s and a collection of destination nodes t_i , $i = 1, 2, \ldots, k$, with nonnegative prescribed distances d_i.
Initialization: Apply the BFS algorithm to traverse the tree starting from the origin s. Suppose that $A = \{e_1, e_2, \ldots, e_{n-1}\}$ is the sorted set of arcs in the order of their appearance.
$S = \emptyset$.
$j = 1$.
For j=1 to n-1 **do**
    For i=1 to k **do**
        If $e_j \in P_{t_i}$ and $d'_i > 0$, **then** $S = S \cup \{e_j\}$.
    For l=1 to k **do**
        If $e_j \in P_{t_l}$, **then** $d'(l) = d'(l) - 1$.
Output The solution corresponding to S defined by (6) is optimal to the problem (5) with the optimal objective value $|S|$.

Algorithm 1: Reverse problem with uniform penalties and uniform lengths

belongs to S, then each arc on the unique path from the origin s to e_j also belongs to S. The following lemma guarantees the existence of an optimal solution with this property.

Lemma 3.1. *There exists an optimal solution S^* to the problem (7) so that if $e_j \in S^*$, then $e_l \in S^*$ for each $e_l \in P_{e_j}$.*

Proof. The problem (7) is feasible because $S = A$ is a feasible solution to the problem. Since the problem (7) has a finite number of feasible solutions, it follows that the problem has at least an optimal solution. Suppose that S^* is an optimal solution to the problem. If S^* does not satisfy the desired property, then $e_j \in S^*$ and $e_l \notin S^*$ for some $e_j \in A$ and some $e_l \in P_{e_j}$. Define $\bar{S} = (S^* \backslash \{e_j\}) \cup \{e_l\}$. The relation $e_j \in T_{e_l}$ together with the feasibility of S^* guarantee that $\bar{S}$ is feasible to the problem (7). Therefore, $\bar{S}$ is also an optimal solution because $|S^*| = |\bar{S}|$. By repeating this process, we can obtain an optimal solution $\bar{S}$ satisfying the desired property. □

Theorem 3.2. *Algorithm 1 solves the problem (5) in $O(nk)$ time.*

Proof. Suppose that $\hat{\mathbf{c}}$ is the solution corresponding to the set S obtained by the algorithm. For proving the correctness of Algorithm 1, we show that $\hat{\mathbf{c}}$ is feasible and optimal to the problem (5). Suppose that $\hat{\mathbf{c}}$ is not feasible by contradiction. Then, there exists some index $i_0 \in \{1, 2, \ldots, k\}$ such that $d'_{i_0} > 0$. Consequently,

$|S \cap P_{t_{i_0}}| < |P_{t_{i_0}}|$. This implies that there exists at least one arc e_{j_0} belonging to $P_{t_{i_0}} \backslash S$. This contradicts the process of line 7 of Algorithm 1 because $e_{j_0} \notin S$ while $e_{j_0} \in P_{t_{i_0}}$ with $d'_{i_0} > 0$.
Assume that S^* is an optimal solution of the problem (7) satisfying the conditions of Lemma 3.1. Suppose that S is the set obtained by Algorithm 1 and $|S| > |S^*|$ by contradiction. Then, $S \backslash S^* \neq \emptyset$. Select one arc $e_{j_0} \in S \backslash S^*$ with the property that $S \cap T_{e_{j_0}} = \emptyset$. Based on Lemma 3.1, $S^* \cap T_{e_{j_0}} = \emptyset$. Hence,

$$d'_i = |P_{t_i} \backslash S| - d_i < |P_{t_i} \backslash S^*| - d_i \leq 0 \quad \forall t_i \in T_{e_{j_0}}. \tag{8}$$

Consider the iteration that e_{j_0} is added to S. In this iteration, $d'_{i_0} = |P_{t_{i_0}} \backslash S| - d_{i_0} > 0$ for some $t_{i_0} \in T_{e_{j_0}}$. Adding e_{j_0} to S decreases d'_{i_0} by one unit. In the next iterations, d'_{i_0} remains unchanged because $S \cap T_{e_{j_0}} = \emptyset$. Therefore, d'_{i_0} is nonnegative which contradicts (8).
Now, we analyze the complexity of Algorithm 1. The number of iterations is $n-1$ because the algorithm examines one arc in each iteration. We show that line 6 of Algorithm 1 can be done in $O(k)$ time. This implies that the complexity of Algorithm 1 is $O(kn)$. For each arc $e_j \in A$, suppose that the accessible list $L(e_j)$ is the set of destination nodes t_i so that $e_j \in P_{t_i}$. The algorithm maintains each accessible list $L(e_j)$ as a linked list. In each iteration, an accessible list is traversed to check the existence of an index $i \in \{1, 2, \ldots, k\}$ with $e_j \in P_{t_i}$ and $d'_i > 0$. When arc e_j is added to S, d'_i is decreased by 1 for each $t_i \in L(e_j)$. Both these operations require at most $O(k)$ time. This completes the proof. □

Example 3.3. *Consider the problem (5) defined on the network shown in Figure 2.a. Algorithm 1 gives us the optimal solution corresponding to* $S = \{e_1, e_4\}$.

4. REVERSE PROBLEM WITHOUT NONNEGATIVITY RESTRICTIONS

In this section, we consider the TRSP problem without nonnegativity restrictions, and show that the problem is transformable to a minimum cut problem on a tree-like network. Consequently, it can be solved in strongly polynomial time.
For a given tree $T(V, A)$ with length vector $\mathbf{c}$, the problem (1) on trees without nonnegativity restrictions is stated formally as follows:

$$\begin{aligned} \min \quad & z = \sum_{e_j \in A} w_j H(c_j, \hat{c}_j) \\ s.t. \quad & \sum_{e_j \in P_{t_i}} \hat{c}_j \leq d_i \quad \forall i = 1, \ldots, k, \\ & \hat{c}_j \leq c_j \quad \forall e_j \in A, \end{aligned} \tag{9}$$

where the parameters are defined as in the problems (1) and (2).

Remark 4.1. *In the previous sections, we have assumed that the arc lengths and the prescribed distances are nonnegative. Here, we drop this assumption.*

Suppose that V' is the subset of $\{1, 2, \ldots, k\}$ so that $i \in V'$ if and only if $d_i'' = \sum_{e_j \in P_{t_i}} c_j - d_i$ is positive. We construct the tree-like network $\bar{G}(\bar{V}, \bar{A}, \bar{\mathbf{w}})$ in the following manner.

- The node set $\bar{V}$ is $V \cup \{t\}$.
- $\bar{G}$ contains the same arcs of T plus additional arcs (t_i, t), called artificial arcs, for each $i \in V'$, i.e., $\bar{A} = A \cup \{(t_i, t) : i \in V'\}$.
- Each arc $e_j \in A$ has a capacity $\bar{w}_j$ equal to w_j and a fixed capacity $W = \sum_{e_j \in A} w_j$ is associated with each artificial arc.

Suppose that C^* is a minimum $s - t$ cut of $\bar{G}(\bar{V}, \bar{A}, \bar{\mathbf{w}})$. Obviously, C^* has not any artificial arc. Consider the solution $\hat{\mathbf{c}}^*$ defined by

$$\hat{c}_j^* = \begin{cases} c_j & e_j \notin C^*, \\ c_j - \max_{t_i \in T_{e_j}} \{d_i''\} & e_j \in C^*, \end{cases} \qquad \forall e_j \in A. \tag{10}$$

We show that $\hat{\mathbf{c}}^*$ is an optimal solution to the problem (9). First, note that there exists at least one destination $t_i \in T_{e_j}$ with $d_i'' > 0$ for each $e_j \in C^*$. If not, we can remove e_j from C^* and however, C^* remains an $s - t$ cut which contradicts to the definition of a cut. This implies that the objective value of $\hat{\mathbf{c}}^*$ is equal to the capacity of C^*, i.e., $\sum_{e_j \in A} w_j H(c_j, \hat{c}_j^*) = \sum_{e_j \in C^*} w_j$. The fact that there exists an arc e_{j_i} belonging to C^* for each $i \in V'$ guarantees that $\hat{\mathbf{c}}^*$ is feasible because

$$\begin{aligned} \sum_{e_j \in P_{t_i}} \hat{c}_j^* &\leq \sum_{e_j \in P_{t_i}} c_j - \max_{t_l \in T_{e_{j_i}}} \{d_l''\} \\ &\leq \sum_{e_j \in P_{t_i}} c_j - d_i'' = d_i \qquad \forall i \in V'. \end{aligned}$$

Suppose that $\hat{\mathbf{c}}^*$ is not optimal by contradiction. Then, there exists a feasible solution $\hat{\mathbf{c}}^0$ so that $\sum_{e_j \in A} w_j H(c_j, \hat{c}_j^0) < \sum_{e_j \in C^*} w_j$. Let C^0 be the set of modified arcs of $\hat{\mathbf{c}}^0$, i.e., $C^0 = \{e_j \in A : \hat{c}_j^0 \neq c_j\}$. It is easy to see that C^0 separates s and t. Therefore, C^0 contains an $s - t$ cut whose capacity is less than that of the cut C^*. This leads to a contradiction. We have thus established the following result.

Lemma 4.2. *If C^* is a minimum $s - t$ cut of $\bar{G}(\bar{V}, \bar{A}, \bar{\mathbf{w}})$, then $\hat{\mathbf{c}}^*$ defined by (10) is an optimal solution to the problem (9).*

Now we are ready to state the proposed algorithm (see Algorithm 2) for solving the problem (9).

Theorem 4.3. *Algorithm 2 solves the problem (9) in $O(n)$ time.*

Proof. The correctness of Algorithm 2 is immediate by Lemma 4.2. We analyze its complexity. Obviously, the bottleneck operation is to find a minimum cut on the tree-like network $\bar{G}$. This can be done in linear time by solving the corresponding maximum flow problem (refer to Theorem 2 of [14]). Therefore, Algorithm 2 can solve the problem (9) in linear time. □

Input: A tree $T(V, A)$ with length vector $\mathbf{c}$ and penalty vector $\mathbf{w}$, an origin node s and a collection of destination nodes t_i , $i = 1, 2, \ldots, k$, with prescribed distances d_i.
For $i \in \{1, 2, \ldots, t\}$ **do** $d_i'' = \sum_{e_j \in P_{t_i}} c_j - d_i$.
flag= true.
For $i \in \{1, 2, \ldots, t\}$ **do**
    If $d_i'' > 0$ **then** flag=false.
If flag==true **then** $\hat{\mathbf{c}}^* = \mathbf{c}$ and stop **else** $V' = \{i : d_i'' > 0\}$.
Construct the tree-like network $\bar{G}(\bar{V}, \bar{A}, \bar{\mathbf{w}})$ in the following manner:
    $\bar{V} = V \cup \{t\}$.
    $\bar{A} = A \cup \{(t_i, t) : i \in V'\}$.
    If $e_j \in A$ **then** $\bar{w}_j = w_j$ **else** $\bar{w}_j = \sum_{e_j \in A} w_j$.
Find a minimum $s - t$ cut C^* of $\bar{G}(\bar{V}, \bar{A}, \bar{\mathbf{w}})$.
Obtain $\hat{\mathbf{c}}^*$ by (10) using C^*.
Output: $\hat{\mathbf{c}}^*$ is an optimal solution to the problem (9).

Algorithm 2: Reverse problem without nonnegativity restrictions

5. COMPUTATIONAL EXPERIMENTS

In this section, we have conducted a computational study to observe the performance of Algorithms 1 and 2. The following computational tools were used to develop algorithms: Python 2.7.5, Matplotlib 1.3.1 and NetworkX 1.8.1. All computational experiments were conducted on a 32-bit Windows 7 with Processor Intel(R) Core(TM) i5 − 3210M CPU @2.50GHz and 4 GB of RAM.

For computational experiments, we first generate a random tree with n nodes. We suppose that trees are rooted in an origin node s and their arcs are orientated such that a unique path exists from s to each other nodes. In generated instances, we use random data generated using a uniform random distribution as follows:

$$c_{ij} \sim U(1, n) \quad \forall (i, j) \in A,$$
$$w_{ij} \sim U(1, n) \quad \forall (i, j) \in A,$$
$$d_i \sim U(1, n) \quad \forall i \in V.$$

We count the average number of modified arcs and calculate average CPU time on a series of instances. We have tested the algorithms on five classes of networks which differ from the number of nodes, varying from 10 to 1000. There are 100 random instances generated for each class of networks. Tables 1 and 2 present average performance statistics of Algorithms 1 and 2, respectively. The running times of Algorithms 1 and 2 grow linearly as the size of instance increases. It is remarkable that the number of modified arcs in the reverse problems with uniform data increases faster than that in the unconstrained reverse problems. This is a reasonable result because in the unconstrained problem, arc costs can

Table 1: Average performance statistics of Algorithm 1

Number of nodes	Average running time (second)	Average number of modified arcs
10	0.0065	1.88
50	0.0083	8.92
100	0.0139	15.28
500	0.0413	40.65
1000	0.1004	73.6

Table 2: Average performance statistics of Algorithm 2

Number of nodes	Average running time (second)	Average number of modified arcs
10	0.0050	1.87
50	0.0090	2.1
100	0.0182	2.22
500	0.1133	2.28
1000	0.2102	2.30

be decreased in any arbitrary amount and consequently, the lesser number of modifications can satisfy the constraints (2b).

As a special case of trees, we perform experiments on star-tree networks G where any two origin-destination paths of G have no common arcs [17]. Throughout experiments, we assume that the origin node s is a leaf of G, i.e., one node with degree 1. Table 3 presents experimental results. As seen in Table 3, Algorithm 2 often needs to modify capacity of one arc for solving the problem. This arc is the one with least penalty among all the arcs on the path from s to a node whose degree is greater than 2.

6. CONCLUSION

It is known that the reverse shortest path problem on trees under the sum-type Hamming distance is NP-hard [16]. In this article, we studied the special cases of the problem which are polynomially solvable. First, we considered the case with uniform lengths and showed that this problem can be transformed to a minimum

Table 3: Average performance statistics of Algorithms 1 and 2 for star-tree networks

Number of nodes	Average running time (second)		Average number of modified arcs	
	Algorithm 1	Algorithm 2	Algorithm 1	Algorithm 2
10	0.0020	0.0031	1.72	1.12
50	0.0035	0.0035	8.17	1.01
100	0.0063	0.0073	11.45	1
500	0.0822	0.0971	32.12	1
1000	0.1024	0.1542	68.21	1

cost flow problem. We also proposed an efficient algorithm for the special case with unform lengths as well as uniform penalties. Finally, we considered the problem without nonnegativity restrictions and showed that the problem can be reduced to a minimum cut problem on a tree-like network and consequently, it can be solved in linear time.

Due to NP-hardness of the reverse shortest path problem under the sum-type Hamming distance, it is meaningful to design heuristic and approximation algorithms for obtaining satisfying solutions and to consider other cases where the problem is polynomially solvable such as the problem with multiple origins and multiple destinations and the problem on general networks.

Acknowledgments: The authors wish to thank the anonymous referees whose valuable comments allowed us to improve the paper.

REFERENCES

[1] Ahuja, R. K., Magnanti, T. L. and Orlin, J. B., *Network Flows*, *Prentice-Hall, Englewood Cliffs*, 1993.

[2] Ahuja, R. K. and Orlin, J. B., "Inverse optimization", *Operation Research*, 49 (2001) 771–783.

[3] Burton, D. and Toint, Ph. L., "On an instance of the inverse shortest paths problem", *Mathematical Programming*, 53 (1992) 45–61.

[4] Burton, D. and Toint, Ph. L., On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Mathematical Programming*, 63 (1994) 1–22.

[5] Burton, D., Pulleyblank, W.R. and Toint, Ph.L., "The inverse shortest path problem with upper bounds on shortest path costs", In: Pardalos, P., Hearn, D.W., Hager, W.H. (Editors), *Lecture Notes in Economics and Mathematical Systems, Springer*, 450 (1997) 156–171.

[6] Call, M., *Inverse Shortest Path Routing Problems in the Design of IP Networks*, Department of Mathematics Linkping University, Sweden, 2010.

[7] Cui, T. and Hochbaum, D. S., "Complexity of some inverse shortest path lengths problems", *Networks*, 56 (2010) 20–29.

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

[9] Fekete, S.P., Hochstattler, W.H., Kromberg, S. and Moll, C., "The complexity of an inverse shortest paths problem", Contemporary trends in discrete mathematics: From DIMACS and DIMATIA to the future, Graham, R.L., Kratochvil, J., Nesetrol, J., Roberts, F.S. (Editors). American Mathematical Society, DIMATIA-DIMACS Conference, Stirin Castle, Czech Republic, 1997, 49–113.

[10] Gódor, I., Harmatos, J. and Jüttner, A., "Inverse shortest path algorithms in protected UMTS access networks", *Computer Communications*, 28 (2005) 765–772.

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

[12] Tayyebi, J. and Aman, M., "On inverse linear programming problems under the bottleneck-type weighted Hamming distance", *Discrete Applied Mathematics*, (2015) DOI: 10.1016/j.dam.2015.12.017.

[13] Tayyebi, J. and Aman, M., "Note on inverse minimum cost flow problems under the weighted hamming distance", *European Journal of Operational Research*, 234 (2014) 916–920.

[14] Vygen, J., "On dual minimum cost flow algorithms", *Math. Meth. Oper. Res.*, 56 (2002) 101–126.

[15] Xu, S. and Zhang, J., "An Inverse Problem of the Weighted Shortest Path Problem", *Japan Journal of Industrial Applied Mathematics*, 12 (1995) 47–59.

[16] Zhang, B. W., Zhang, J. Z. and Qi, L. Q., "The shortest path improvement problems under Hamming distance", *Journal of Combinatorial Optimization*, 12 (4) (2006) 351–361.

[17] Zhang, B., Guan, X., He, C. and Wang, S., "Algorithms for the Shortest Path Improvement Problems under Unit Hamming Distance", *Journal of Applied Mathematics*, Article ID 847317 (2013) 8 pages.

[18] Zhang, J. Z., Ma, Z. and Yang, C., "A column generation method for inverse shortest path problems", *Zeitschrift fr Operations Research*, 41 (3) (1995) 347–358.
[19] Zhang, J. Z. and Ma, Z., "A network flow method for solving some inverse combinatorial optimization problems", *Optimization: A Journal of Mathematical Programming and Operations Research*, 37 (1) (1996) 59–72.
[20] Zhang, J. Z. and Lin, Y. X., "Computation of the reverse shortest path problem", *Journal of Global Optimization*, 25 (2003) 243–261.