Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 14, No. 2, pp. 245–260, 2018

INVERSE ECCENTRIC VERTEX PROBLEM ON TREES

Javad Tayyebi* and Massoud Aman

Abstract: Given an undirected network $G(V, A, \mathbf{c})$, the inverse eccentric vertex problem consists of modifying the length vector $\mathbf{c}$ minimally under some bound constraints such that a prescribed vertex t becomes the furthest vertex from another prescribed vertex s with respect to the modified length vector. The modifications can be measured by different distances. In this paper, we consider the inverse eccentric vertex problem on trees when the modifications are measured by the weighted l_1 and l_∞ norms and also by the weighted sum-type Hamming distance. With the weighted l_1 norm, it is shown that the problem is reduced to an instance of the minimum cost flow problem and consequently, it can be solved in strongly polynomial time. With the weighted l_∞ norm, an efficient algorithm is designed to solve the problem. With the weighted sum-type Hamming distance, it is proved that the problem is NP-hard on trees and strongly NP-hard on bipartite networks.

Key words: *eccentricity, Inverse problem, Hamming distance, Strongly NP-completeness*

Mathematics Subject Classification: *90C27, 05C85, 90C60*

1 Introduction

Suppose that $G(V, E, \mathbf{c})$ is an undirected and connected network where V is the set of n vertices, $E = \{e_1, e_2, \ldots, e_m\}$ is the edge set and $\mathbf{c}$ is the nonnegative edge length vector. We denote by $d_{\mathbf{c}}(v, w)$ the length of the shortest path from vertex v to vertex w with respect to the length vector $\mathbf{c}$. For a vertex v, the eccentric vertex of v, denoted by $ecc(v)$, is the furthest vertex from v, i.e., $ecc(v) = argmax\{d_{\mathbf{c}}(v, w) : w \in V\}$. The problem of finding the eccentric vertex of a vertex has several practical applications in formal hardware verification [21]. For more details on some notions and properties of the eccentric vertex, we refer the reader to [7, 23].

In the inverse eccentric vertex problem, the goal is to modify edge lengths as little as possible in such a way that a specified vertex t, called the destination, becomes the furthest vertex from another specified vertex s which is referred to as the origin. Therefore, the problem can be formulated as follows:

$$\begin{array}{ll} \min & z = f(\mathbf{c}, \hat{\mathbf{c}}) \\ s.t. & d_{\hat{\mathbf{c}}}(s, v) \leq d_{\hat{\mathbf{c}}}(s, t) \quad \forall v \in V, \\ & \max\{0, c_j - l_j\} \leq \hat{c}_j \leq c_j + u_j \quad \forall e_j \in E, \end{array} \tag{1.1}$$

where $\hat{\mathbf{c}}$ is the new nonnegative length vector to be determined, $l_j \geq 0$ and $u_j \geq 0$ are respectively the lower and upper bounds on length modifications of e_j and $f : \mathrm{R}^m \times \mathrm{R}^m \longrightarrow \mathrm{R}$

*Corresponding author.

is a penalty function to measure the distance between $\mathbf{c}$ and $\hat{\mathbf{c}}$.

To the best of our knowledge, the problem (1.1) has been only studied by Nguyen and Chassein [22] with the l_1 norm penalty function given by $f(\mathbf{c}, \hat{\mathbf{c}}) = \sum_{e_j \in E} |\hat{c}_j - c_j|$. The authors showed that the problem is NP-hard even on the cactus networks. Then they presented some efficient approaches just for the cases that the network is a tree or a cycle.

In this article, we consider the problem (1.1) on trees under three different distances:

(I) the weighted l_1 norm: $f(\mathbf{c}, \hat{\mathbf{c}}) = \sum_{e_j \in E} w_j |\hat{c}_j - c_j|$;

(II) the weighted l_∞ norm: $f(\mathbf{c}, \hat{\mathbf{c}}) = \max_{e_j \in E} |\hat{c}_j - c_j|$;

(III) the weighted sum-type Hamming distance: $f(\mathbf{c}, \hat{\mathbf{c}}) = \sum_{e_j \in E} w_j H(c_j, \hat{c}_j)$ where $H(c_j, \hat{c}_j)$ 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.

For the weighted l_1 norm, it is shown that the problem is transformed into a minimum cost flow problem on an auxiliary network and consequently, it can be solved in strongly polynomial time. It is remarkable that a combinatorial algorithm is designed in [22] to solve the problem (1.1) on a tree under the l_1 norm. But the algorithm can not solve the problem under the weighted l_1 norm in its current form. Hence, the main advantage of our approach is its generality. For the weighted l_∞ norm, an efficient algorithm is designed to solve the problem by using the binary search technique. For the weighted sum-type Hamming distance, it is proved that the problem is NP-hard on trees. As a further result, it is also shown that the problem (1.1) is strongly NP-hard on bipartite networks under the weighted sum-type Hamming distance and the l_1 norm.

A typical application of the problem (1.1) is in security network design discussed by Nguyen and Chassein [22]. Here, we illustrate another application on urban transportation systems. Figure 1.a shows two main city zones C and C' joined by several routes. There exists a hospital in the neighborhood of the zone C'. Due to transporting patients between the hospital and health centers of the zone C', the road network planners are interested in reducing the traffic flow along the road joining the hospital and the zone C'. This can be done by adding or removing toll stations and traffic lights, i.e., by increasing the cost of transportation along the routes which pass through the hospital and decreasing the cost of transportation along the others. This problem is equivalent to solving an inverse eccentric vertex problem defined on the network shown in Figure 1.b in which C is the origin vertex and C'_1 is the destination vertex. For obtaining a solution to the inverse eccentric vertex problem, one can add toll stations and traffic lights in the path from C to C'_1 and also, can remove some of them in the other paths. This yields the reduction of traffic flow along the road joining C' and the hospital. Since adding (or removing) toll stations and traffic lights has a fixed amount of penalty, the objective function can be stated in term of the Hamming distance instead of the l_p norms.

The rest of the paper is organized as follows. Section 2 considers the problem (1.1) on trees under the weighted l_1 norm. Section 3 presents an efficient algorithm for solving the problem (1.1) on trees under the weighted l_∞ norm. Section 4 is devoted to the complexity of the problem (1.1) under the sum-type Hamming distance. Finally, some concluding remarks are given in Section 5.

2 Literature review

The concept of inverse problems was first proposed by Tarantola in geophysical sciences [24]. Subsequently, Burton and Toint [8, 9] studied the inverse shortest path problem in which the

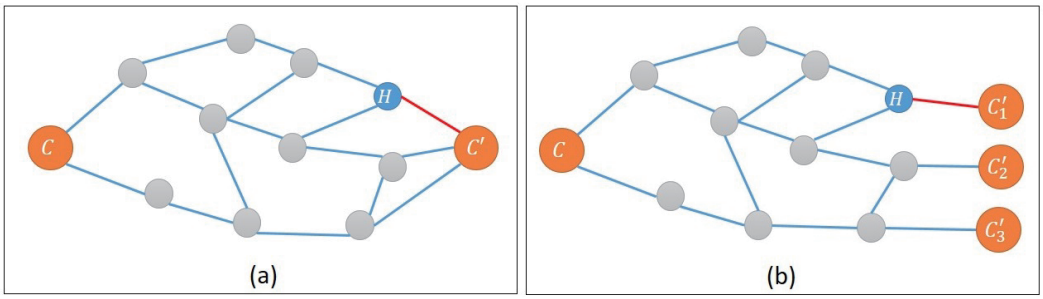

Figure 1: (a) An urban transportation system; (b) The corresponding network of the inverse eccentric vertex problem.

l_2 norm is used to measure the modifications. They also introduced two applications of the inverse shortest path problem in traffic modeling and seismic tomography. Ahuja and Orlin considered the inverse linear programming problem under the l_1 and l_∞ norms [2, 3]. They showed that this problem can be formulated as a new linear programming problem. They also considered the inverse minimum cost flow problem as a special case. In the case that the modifications of the cost vector is measured by the l_1 norm, they proved that the inverse minimum cost flow problem is reducible to a unit capacity minimum cost flow problem. For the l_∞ norm, they converted the inverse problem into a minimum cost-to-time ratio cycle problem. Afterwards, many authors considered the inverse optimization problems under the l_1 and l_∞ norms (for a survey, see [14, 12]).

The Hamming distances are also used to measure the modifications in the inverse optimization problems. Two types of the Hamming distances are applied in the literature: (I) the weighted sum-type Hamming distance (H_1); (II) the weighted bottleneck-type Hamming distance (H_∞). As the Hamming distances are nonconvex and discontinuous at every point, some inverse optimization problems under the Hamming distances have different behaviour in contrast to those under the l_1 and l_∞ norms. For example, the inverse assignment problem under the H_1 distance is NP-hard while the problem under the l_1 norm can be solved in strongly polynomial time [17, 28]. This different behaviour can be also seen in the inverse minimum cost flow problem [3, 5, 15, 25, 26], and the inverse minimum cut problem [27, 18]. However, some inverse optimization problems have a similar behaviour under the H_1 distance and the l_1 norm, e.g., the inverse maximum flow problem [11, 19] and the capacity inverse minimum cost flow problem [4, 13, 20]. This argument shows that it is worthwhile to study inverse optimization problems under various distances. In this article, we consider the inverse eccentric vertex problem on trees and show that the problem is efficiently solvable under the l_1 and l_∞ norms while it is NP-hard under the H_1 distance.

3 Inverse eccentric vertex problem under the weighted l_1 norm

In this section, we consider the problem (1.1) under the weighted l_1 norm when the network is a tree. We denote this tree by $T(V, E, \mathbf{c})$. As every two vertices v and v' of T are joined by a unique path $P_{vv'}$, the problem can be written as follows:

$$\min \quad z = \textstyle\sum_{e_j \in E} w_j |\hat{c}_j - c_j| \tag{3.1a}$$

$$\text{s.t.} \quad \textstyle\sum_{e_j \in P_{sv}} \hat{c}_j \leq \sum_{e_j \in P_{st}} \hat{c}_j \qquad \forall v \in V, \tag{3.1b}$$

$$\max\{0, c_j - l_j\} \leq \hat{c}_j \leq c_j + u_j \quad \forall e_j \in E, \tag{3.1c}$$

where $w_j \geq 0$ is the weight associated with e_j and the other parameters are defined as in the problem (1.1).

From now on suppose that the tree T is rooted at the origin s. Each vertex v, except the origin, has a unique predecessor, which is just the next vertex on the unique path in the tree from v to s. We denote the predecessor of vertex v by $pred(v)$. A vertex v' is called a successor of vertex v, denoted by $succ(v)$, if $pred(v') = v$. The descendants of a vertex v are the vertex v itself, its successors, successors of its successors, and so on. The set of descendants of v is denoted by $des(v)$. The descendants of an edge e_j are the descendants of its endpoints. The set of descendants of e_j is denoted by the similar notation $des(e_j)$.

The constraints (3.1c) guarantee the nonnegativity of $\hat{\mathbf{c}}$. One can replace the constraints (3.1c) by $-\mathbf{l}' \leq \hat{\mathbf{c}} - \mathbf{c} \leq \mathbf{u}$ where the lower bound vector $\mathbf{l}'$ is defined as $l'_j = \min\{l_j, c_j\}$ for every $e_j \in E$.

Without loss of generality, we assume that the destination vertex t is a leaf of the tree, namely, a vertex with no successors. Because if not then we have to set $\hat{c}_j = 0$ for each edge e_j whose endpoints belong to $des(t)$. Due to the nonnegativity of $\hat{\mathbf{c}}$, the destination vertex t is an eccentric vertex of s even if the constraints (3.1b) are satisfied only for the leaves of T. This shows that the problem (3.1) can be rewritten as follows:

$$\begin{aligned}
\min \quad & z = \textstyle\sum_{e_j \in E} w_j |\hat{c}_j - c_j| && (3.2a)\\
\text{s.t.} \quad & \textstyle\sum_{e_j \in P_{sv}} \hat{c}_j \leq \sum_{e_j \in P_{st}} \hat{c}_j \quad \forall v \in L, && (3.2b)\\
& -l'_j \leq \hat{c}_j - c_j \leq u_j \qquad \forall e_j \in E, && (3.2c)
\end{aligned}$$

where L is the set of leaves of T. We denote the number of leaves of T by r.

Lemma 3.1. *The problem (3.2) is feasible if and only if*

$$\sum_{e_j \in P_{st} \setminus P_{sv}} u_j + \sum_{e_j \in P_{sv} \setminus P_{st}} l_j \geq \sum_{e_j \in P_{sv} \setminus P_{st}} c_j - \sum_{e_j \in P_{st} \setminus P_{sv}} c_j \tag{3.3}$$

for each $v \in L$.

Proof. Necessity: Let $\hat{\mathbf{c}}^0$ be a feasible solution to the problem (3.2). Then for each $v \in L$,

$$\sum_{e_j \in P_{sv} \setminus P_{st}} \hat{c}^0_j \leq \sum_{e_j \in P_{st} \setminus P_{sv}} \hat{c}^0_j$$

and based on the bound constraints,

$$\sum_{e_j \in P_{sv} \setminus P_{st}} (c_j - l_j) \leq \sum_{e_j \in P_{st} \setminus P_{sv}} (c_j + u_j),$$

which completes the proof of the necessity.

Sufficiency: Define $\hat{\mathbf{c}}^0$ as follows:

$$\hat{c}^0_j = \begin{cases} c_j + u_j, & e_j \in P_{st}, \\ c_j - l_j, & \text{otherwise}, \end{cases} \qquad \forall e_j \in E.$$

It is easy to see that $\hat{\mathbf{c}}^0$ is a feasible solution to the problem (3.2) if the inequality (3.3) holds for any $v \in L$. □

For obtaining a feasible solution of the problem (3.2), it is natural to increase the initial length c_j for some $e_j \in P_{st}$ and to decrease c_j for some $e_j \notin P_{st}$. Let us present formally this simple observation.

Lemma 3.2. *If the problem (3.2) is feasible, then it contains an optimal solution* $\hat{\mathbf{c}}^*$ *satisfying the following conditions:*

(a) $\hat{c}_j^* \geq c_j, \qquad \forall e_j \in P_{st},$

(b) $\hat{c}_j^* \leq c_j, \qquad \forall e_j \notin P_{st}.$

Proof. The proof is straightforward. □

By using Lemma 3.2, we can focus only on solutions satisfying the conditions (a) and (b). For every $e_j \in E$, we set $x_j = |\hat{c}_j - c_j|$. Based on the conditions *(a)* and *(b)*,

- $x_j = \hat{c}_j - c_j \quad \forall e_j \in P_{st}$;
- $x_j = c_j - \hat{c}_j \quad \forall e_j \in E \backslash P_{st}$.

Hence, the problem (3.2) is converted into

$$\begin{aligned}
\min \quad & z = \sum_{e_j \in E} w_j x_j \\
\text{s.t.} \quad & \sum_{e_j \in P_{sv} \backslash P_{st}} x_j + \sum_{e_j \in P_{st} \backslash P_{sv}} x_j \geq \sum_{e_j \in P_{sv} \backslash P_{st}} c_j - \sum_{e_j \in P_{st} \backslash P_{sv}} c_j \quad \forall v \in L, \\
& 0 \leq x_j \leq u_j \quad \forall e_j \in P_{st}, \\
& 0 \leq x_j \leq l'_j \quad \forall e_j \in E \backslash P_{st},
\end{aligned}$$

In the matrix form, we represent the problem as follows:

$$\begin{aligned}
\min \quad & z = \mathbf{w}^T \mathbf{x} && (3.4a) \\
s.t. \quad & A\mathbf{x} \geq \mathbf{b}, && (3.4b) \\
& 0 \leq \mathbf{x} \leq \mathbf{u}'. && (3.4c)
\end{aligned}$$

Here,

$$\begin{aligned}
&\mathbf{x} = [x_1, x_2, \ldots, x_{n-1}]^T \in \mathrm{R}^{n-1}, \\
&\mathbf{w} = [w_1, w_2, \ldots, w_{n-1}]^T \in \mathrm{R}^{n-1}, \\
&\mathbf{u}' = [u'_1 \ldots, u'_{n-1}]^T \in \mathrm{R}^{n-1}, \\
&\mathbf{b} = [b_1, b_2, \ldots, b_r]^T \in \mathrm{R}^r, \\
&A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{n-1}] \in \mathrm{R}^{r \times (n-1)},
\end{aligned}$$

where

$$u'_j = \begin{cases} u_j & e_j \in P_{st}, \\ l'_j & e_j \in E \backslash P_{st}, \end{cases} \qquad \forall e_j \in E,$$

$$b_v = \sum_{e_j \in P_{sv} \backslash P_{st}} c_j - \sum_{e_j \in P_{st} \backslash P_{sv}} c_j \quad \forall v \in L,$$

and the coefficient matrix A is as follows:

$$a_{vj} = \begin{cases} 1 & e_j \in (P_{sv} \backslash P_{st}) \cup (P_{st} \backslash P_{sv}), \\ 0 & \text{otherwise}, \end{cases} \qquad \forall v \in L, \quad \forall e_j \in E. \qquad (3.5)$$

It is worth noting that an element a_{vj} of A is zero if either $e_j \in P_{st} \cap P_{sv}$ or $e_j \notin P_{st} \cup P_{sv}$. On the other hand, for an edge $e_j \in P_{st}$, if an element a_{vj} is zero then $e_j \in P_{st} \cap P_{sv}$. While for an edge $e_j \notin P_{st}$, if an element a_{vj} is zero then $e_j \notin P_{st} \cup P_{sv}$.
Now, we show that the 1's of each column of A are consecutive whenever its rows are arranged in a special fashion. For arranging the rows of A, we use a slight variation of the well-known Depth-First-Search (DFS) algorithm as described below.

Algorithm 3.3.

Input: A tree $T(V, E)$ with the origin vertex s as well as the destination vertex t.

Initialization: Mark all vertices as unvisited. Set $L' = \emptyset$. Put all the vertices of P_{st} into a stack S in the order of their appearance by starting at s.

Step 1: If all vertices are visited then stop. Otherwise, go to Step 2.

Step 2: Let i be the top element of S . If there exists an unvisited vertex $j \in succ(i)$, then go to Step 3. Otherwise, go to Step 4.

Step 3: Choose an unvisited successor j of i and add it to the top of S. Go to Step 1.

Step 4: Take i out from the top of the stack S. Mark i as visited. If i is a leaf, then add i to L'. Go to Step 1.

In Algorithm 3.3, S is a stack and L' is a sorted list of L. Since each row of A is associated with a leaf of T, we suppose that the rows of A are arranged in the order of L'. To simplify the notations, we denote the leaf vertex after (before) v in L' by $v + 1$ $(v - 1)$.

Remark 3.4. For each edge e_j, the leaves belonging to $des(e_j)$ are consecutive in L' because the algorithm visits a vertex after all of its descendants have been visited.

Lemma 3.5. *All elements of the first row of A are zero.*

Proof. The first row of A corresponds to the destination vertex t because t is the first leaf added to L' (see Algorithm 3.3). If $a_{tj} = 1$ for some $e_j \in E$, then $e_j \in P_{st} \backslash P_{st} = \emptyset$ based on (3.5) which is a contradiction. □

Theorem 3.6. *The one entries of each column of A are consecutive.*

Proof. Our proof is divided into two parts. In the first part, we prove the result for columns $\mathbf{a}_j$ corresponding to $e_j \notin P_{st}$. In second part, we prove that 0's of $\mathbf{a}_j$, $e_j \in P_{st}$, are consecutive. This together with Lemma 3.5 imply that 1's of $\mathbf{a}_j$ are also consecutive for every $e_j \in P_{st}$.

Part 1: Let $e_j \notin P_{st}$. Note that $a_{vj} = 1$ for each $v \in L'$ with $e_j \in P_{sv}$. Equivalently, $a_{vj} = 1$ for each $v \in des(e_j) \cap L'$. Based on Algorithm 3.3, all leaves belonging to $des(e_j)$ appears consecutively in L'. Then 1's of $\mathbf{a}_j$ are consecutive.

Part 2: Let $a_{vj} = 0$ for some $e_j \in P_{st}$. Then $e_j \in P_{sv} \cap P_{st}$ and consequently, $v \in des(e_j)$. Since all leaves belonging to $des(e_j)$ are consecutive in L', it follows that 0's of $\mathbf{a}_j$ are also consecutive. □

Using Theorem 5.2, the problem (3.4) is converted into a linear programming problem whose +1's of each column of the coefficient matrix are consecutive. This problem can be transformed into an instance of the minimum cost flow problem [1]. We now describe this transformation. Assume that we have arranged the constraints (3.1b) in the order of L'

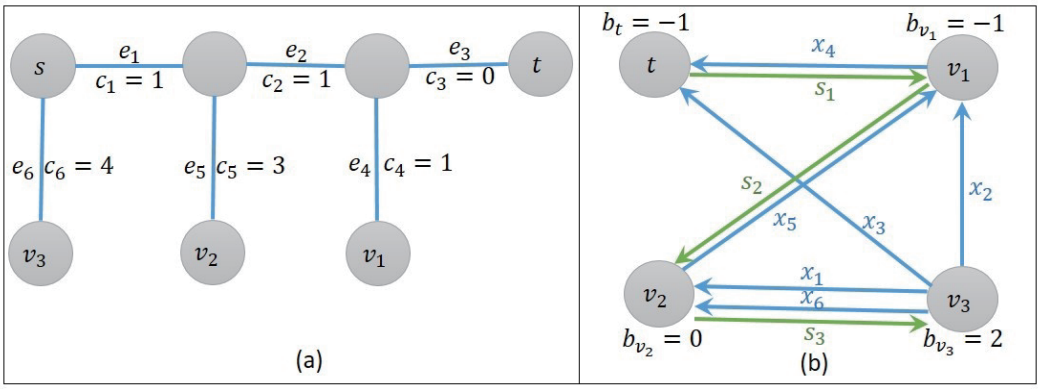

Figure 2: (a) A tree $T(V, E, \mathbf{c})$; (b) The auxiliary network of the corresponding minimum cost flow problem.

. It is remarkable that the first constraint (3.1b) is $\mathbf{0}^T\mathbf{x} \leq 0$. Thus, it can be rewritten as the equality form $\mathbf{0}^T\mathbf{x} = 0$. We bring the vth constraint (3.1b) into an equality form by introducing a surplus variable s_v, for each row $v \in L'\backslash\{t\}$. We next subtract the vth constraint from the $(v-1)$th constraint for each $v \in L'\backslash\{t\}$. These operations create an equivalent linear programming problem with exactly one $+1$ and exactly one -1 in each nonzero column. Note that some columns may be zero. Indeed, column $\mathbf{a}_j$ is zero if edge e_j belongs to P_{sv} for all $v \in L'$. In this situation, we can set $x_j = 0$ because the length modification of e_j changes the length of all paths by the same amount and has no effect on the feasibility of the problem (3.4). Due to the property of the coefficient matrix, the new linear programming problem is a minimum cost flow problem. This minimum cost flow problem is defined on an auxiliary directed network introduced as follows:

- Each leaf vertex $v \in L'$ corresponds to a vertex of the auxiliary network.
- The auxiliary network contains an arc associated with each decision variable x_j and also, an arc associated with each surplus variable s_v. Each arc corresponding to s_v emanates from $v-1$ and terminates at v. Each arc corresponding to x_j emanates from $v_j - 1$ and terminates at v'_j where v_j is the row corresponding to the first element 1 of $\mathbf{a}_j$ and v'_j is the row corresponding to the last element 1 of $\mathbf{a}_j$.
- The cost of each arc x_j is w_j and the cost of each arc s_v is 0.
- The capacity of each arc x_j is u'_j and all arcs s_v are uncapacitated.
- $b_v = \sum_{e_j \in P_{sv}\backslash P_{st}} c_j - \sum_{e_j \in P_{st}\backslash P_{sv}} c_j$ represents the supply/demand of each vertex $v \in L'$.

As each minimum cost flow problem can be solved efficiently [1], we have established the following result.

Theorem 3.7. *The problem (3.1) can be solved in strongly polynomial time.*

We illustrate this transformation using an example.

Example 3.8. Consider the problem (3.1) defined on the network shown in Figure 3.a where $w_j = l_j = u_j = 1$ for each $e_j \in E$. The problem can be formulated as the linear programming problem (3.4) where

$$A = \begin{array}{c|cccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ t(=v_0) & 0 & 0 & 0 & 0 & 0 & 0 \\ v_1 & 0 & 0 & 1 & 1 & 0 & 0 \\ v_2 & 0 & 1 & 1 & 0 & 1 & 0 \\ v_3 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}, \mathbf{w} = \mathbf{u}' = [1,1,1,1,1,1]^T, \quad \mathbf{b} = [0,1,2,2]^T.$$

Note that the constraints of the inequality system $A\mathbf{x} \leq \mathbf{b}$ are arranged by Algorithm 3.3. By introducing surplus variable s_i for each constraint v_i, $i = 1,2,3$, we state the constraints $A\mathbf{x} \leq \mathbf{b}$ in the equality form. Now, we subtract the constraint v_i from the constraint v_{i-1} for each $i = 1,2,3$. An equivalent linear programming problem is obtained with $A' \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} = \mathbf{b}'$ instead of $A\mathbf{x} \leq \mathbf{b}$ where

$$A' = \begin{array}{c|ccccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & s_1 & s_2 & s_3 \\ t & 0 & 0 & -1 & -1 & 0 & 0 & 1 & 0 & 0 \\ v_1 & 0 & -1 & 0 & 1 & -1 & 0 & -1 & 1 & 0 \\ v_2 & -1 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 1 \\ v_3 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & -1 \end{array}, \mathbf{b}' = [-1,-1,0,2]^T.$$

This linear programming problem is an instance of the minimum cost flow problem defined on the network shown in Figure 3.b. The optimal solution of this minimum cost flow problem is

$$x_2 = x_3 = 1,\ x_1 = x_4 = x_5 = x_6 = s_1 = s_2 = s_3 = 0$$

with the optimal objective value 2. Therefore, the optimal solution of the inverse eccentric problem (3.1) is

$$\hat{c}_1 = 1,\ \hat{c}_2 = 2,\ \hat{c}_3 = 1,\ \hat{c}_4 = 1,\ \hat{c}_5 = 3 \text{ and } \hat{c}_6 = 4$$

with the optimal objective value 2.

Theorem 3.9. *The problem (3.1) has an integer optimal solution if the vectors* $\mathbf{l}, \mathbf{u}$ *and* $\mathbf{c}$ *are integers.*

Proof. The result follows immediately from the integrality property of network flows [1]. □

4 Inverse eccentric vertex problem under the weighted l_∞ norm

In this section, we consider the inverse eccentric vertex problem on the tree $T(V,E,\mathbf{c})$ under the weighted l_∞ norm. This problem can be formulated as follows:

$$\min \quad z = \max_{e_j \in E}\{w_j|\hat{c}_j - c_j|\} \tag{4.1a}$$

$$\text{s.t.} \quad \sum_{e_j \in P_{sv}} \hat{c}_j \leq \sum_{e_j \in P_{st}} \hat{c}_j \quad \forall v \in L, \tag{4.1b}$$

$$\max\{0, c_j - l_j\} \leq \hat{c}_j \leq c_j + u_j \quad \forall e_j \in E, \tag{4.1c}$$

where the parameters are defined as in the problems (1.1) and (3.1).

By Theorem 3.9, the optimal objective value of the problem (3.1) is an integer whenever the problem data are integral. This property is not valid for the problem (4.2). For an instance, consider the problem (4.2) defined on the network shown in Figure 3.a where $w_j = l_j = u_j = 1$ for each $e_j \in E$. It is easy to see that an optimal solution of the problem

is $\hat{c}_1 = \frac{5}{3},\ \hat{c}_2 = \frac{5}{3},\ \hat{c}_3 = \frac{2}{3},\ \hat{c}_4 = \frac{1}{3},\ \hat{c}_5 = \frac{7}{3},\ \hat{c}_6 = 4$ with the objective optimal value $\frac{2}{3}$.

Similar to Lemma 3.2, the following result is also valid for the problem (4.1) because the feasible sets of both the problems (3.2) and (4.1) are the same.

Lemma 4.1. *If the problem (4.1) is feasible, then it has an optimal solution* $\hat{\mathbf{c}}$ *that satisfies the following conditions:*

(a) $\hat{c}_j \geq c_j, \qquad \forall e_j \in P_{st}$;

(b) $\hat{c}_j \leq c_j, \qquad \forall e_j \notin P_{st}$.

Proof. The proof is straightforward. □

Based on Lemma 4.1, for obtaining an optimal solution to the problem (4.1), one can increase the length of some edges $e_j \in P_{st}$ and decrease the length of some edges $e_j \notin P_{st}$ to satisfy the constraint (4.1b). Set $x_j = |\hat{c}_j - c_j|$ for every $e_j \in E$. Thus we have

- $x_j = \hat{c}_j - c_j$ for every $e_j \in P_{st}$;
- $x_j = c_j - \hat{c}_j$ for every $e_j \in E \backslash P_{st}$.

By assuming $\lambda = \max_{e_j \in E}\{w_j|\hat{c}_j - c_j|\}$, we can convert the problem (4.1) into the following linear programming problem:

$$\begin{aligned}
\min \quad & z = \lambda && && (4.2a)\\
\text{s.t.} \quad & \sum_{e_j \in P_{sv}\Delta P_{st}} x_j \geq \sum_{e_j \in P_{sv}\backslash P_{st}} c_j - \sum_{e_j \in P_{st}\backslash P_{sv}} c_j && \forall v \in L, && (4.2b)\\
& w_j x_j \leq \lambda && \forall e_j \in E, && (4.2c)\\
& 0 \leq x_j \leq u_j && \forall e_j \in P_{st}, && (4.2d)\\
& 0 \leq x_j \leq l'_j && \forall e_j \in E\backslash P_{st}, && (4.2e)
\end{aligned}$$

where $P_{sv}\Delta P_{st} = (P_{sv}\backslash P_{st}) \cup (P_{st}\backslash P_{sv})$ and $l'_j = \min\{l_j, c_j\}$ for every $e_j \in E$.
Suppose that the problem (4.2) has a feasible solution with the objective value λ. One can increase (or decrease) the modified lengths of this solution as long as the maximum weighted change so that it still remains feasible. This observation is a motivation for introducing a special type of feasible solutions. For a fixed value λ, we define $\mathbf{x}^\lambda$ as

$$x_j^\lambda = \begin{cases} u_j & e_j \in P_{st} \text{ with } w_j = 0,\\ \min\{u_j, \frac{\lambda}{w_j}\} & e_j \in P_{st} \text{ with } w_j \neq 0,\\ l'_j & e_j \in E\backslash P_{st} \text{ with } w_j = 0,\\ \min\{l'_j, \frac{\lambda}{w_j}\} & e_j \in E\backslash P_{st} \text{ with } w_j \neq 0, \end{cases} \qquad \forall e_j \in E. \tag{4.3}$$

The following lemma concerns the feasibility of the solution $(\mathbf{x}^\lambda, \lambda)$.

Lemma 4.2. *If the problem (4.2) contains a feasible solution with the objective value* λ_0*, then the solution* $(\mathbf{x}^\lambda, \lambda)$ *is feasible to the problem where* $\mathbf{x}^\lambda$ *is defined by (4.3) for each* $\lambda \geq \lambda_0$.

Proof. The proof is trivial. □

Based on Lemma 4.2, we can restrict our attention to solutions $(\mathbf{x}^\lambda, \lambda)$ and look for an optimal solution among such solutions. The following corollary provides formally this result.

Corollary 4.3. *If the optimal objective value of the problem (4.2) is λ^*, then $(\mathbf{x}^{\lambda^*}, \lambda^*)$ is an optimal solution.*

Corollary 4.4. *The problem (4.2) is feasible if and only if $\mathbf{x}^{\lambda_{\max}}$ satisfies the constraint (4.2b) where $\lambda_{\max} = \max\{\max_{e_j \in E\setminus P_{st}}\{w_j l'_j\}, \max_{e_j \in P_{st}}\{w_j u_j\}\}$.*

Proof. The sufficiency is trivial. We only prove the necessity. Let $\mathbf{x}$ be a feasible solution of the problem (4.2) with the objective value λ_0. Since $\mathbf{x}^{\lambda_{\max}} = \mathbf{x}^{\lambda}$ for each $\lambda \geq \lambda_{\max}$, we can assume that $\lambda_0 \leq \lambda_{\max}$. Therefore, the problem contains a solution $(\mathbf{x}, \lambda_0)$ whose objective value is at most $\lambda_{\max}$. Based on Lemma 4.2, this implies that $(\mathbf{x}^{\lambda_{\max}}, \lambda_{\max})$ is feasible to the problem. □

Obviously, if t is an eccentric vertex of s with respect to the initial length vector $\mathbf{c}$, then $\lambda = 0$ is the optimal objective value of the problem (4.2). From Corollary 4.4, we can always restrict our attention to the interval $[0, \lambda_{\max}]$ for finding the optimal objective value of the problem (4.2). Hence, the problem is reduced to finding the least value $\lambda \in [0, \lambda_{\max}]$ so that the vector $\mathbf{x}^{\lambda}$ satisfies the constraints (4.2b). Suppose that $\lambda_0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_m$ is a sorted list of elements of the set $\{w_j l'_j : e_j \in E\setminus P_{st}\} \cup \{w_j u_j : e_j \in P_{st}\} \cup \{0\}$. Obviously, $\lambda_0 = 0$ and $\lambda_m = \lambda_{\max}$. Suppose that the problem (4.2) is feasible and its optimal objective value is greater than zero. Then the problem contains an optimal solution with objective value belonging to $(\lambda_0, \lambda_m]$. Our proposed algorithm contains two phases. In the first phase, the algorithm finds an interval $(\lambda_{i-1}, \lambda_i]$ for some $i \in \{1, 2, \ldots, m\}$ so that the optimal objective value belongs to it. In the second phase, the optimal objective value is computed by using the result obtained from the first phase. For finding an interval $(\lambda_{i-1}, \lambda_i]$ containing the optimal objective value, it is sufficient to look for an index $i \in \{1, 2, \ldots, m\}$ so that $(\mathbf{x}^{\lambda_{i-1}}, \lambda_{i-1})$ is not feasible while $(\mathbf{x}^{\lambda_i}, \lambda_i)$ is feasible. Such an index i is identified by using the binary search technique. In the second phase, the algorithm computes the optimal objective value $\lambda^* \in (\lambda_{i-1}, \lambda_i]$.

We now show how to compute the optimal objective value λ^*. Based on Corollary 4.3, $\mathbf{x}^{\lambda^*}$ is an optimal solution of the problem. By substituting $\mathbf{x}^{\lambda^*}$ in the constraints (4.2b), for each $v \in L$, we have

$$\begin{aligned}
\sum_{e_j \in P_{sv}} c_j - \sum_{e_j \in P_{st}} c_j \quad &\leq \sum_{e_j \in P_{sv} \Delta P_{st}} x_j^{\lambda^*} \\
&= \sum_{e_j \in P_{sv}\setminus P_{st} \text{ with } w_j \neq 0} \min\{\frac{\lambda^*}{w_j}, l'_j\} + \sum_{e_j \in P_{sv}\setminus P_{st} \text{ with } w_j = 0} l'_j \\
&+ \sum_{e_j \in P_{st}\setminus P_{sv} \text{ with } w_j \neq 0} \min\{\frac{\lambda^*}{w_j}, u_j\} + \sum_{e_j \in P_{st}\setminus P_{sv} \text{ with } w_j = 0} u_j \\
&= \lambda^* \sum_{e_j \in (P_{sv} \Delta P_{st})\setminus E^{\lambda_i}} \frac{1}{w_j} + \sum_{e_j \in (P_{sv}\setminus P_{st}) \cap E^{\lambda_i}} l'_j + \sum_{e_j \in (P_{st}\setminus P_{sv}) \cap E^{\lambda_i}} u_j
\end{aligned}$$

where $E^{\lambda_i} = \{e_j \in E : \lambda_j < \lambda_i\}$. Consequently,

$$\lambda^* \geq \frac{\sum_{e_j \in P_{sv}} c_j - \sum_{e_j \in P_{st}} c_j - \sum_{e_j \in (P_{sv}\setminus P_{st}) \cap E^{\lambda_i}} l'_j - \sum_{e_j \in (P_{st}\setminus P_{sv}) \cap E^{\lambda_i}} u_j}{\sum_{e_j \in (P_{sv} \Delta P_{st})\setminus E^{\lambda_i}} w_j^{-1}}.$$

Note that the right h and side of the last inequality depends on $v \in L$. Therefore, the optimal objective value can be computed as follows:

$$\lambda^* = \max_{v \in L}\{\frac{\sum_{e_j \in P_{sv}} c_j - \sum_{e_j \in P_{st}} c_j - \sum_{e_j \in (P_{sv}\setminus P_{st}) \cap E^{\lambda_i}} l'_j - \sum_{e_j \in (P_{st}\setminus P_{sv}) \cap E^{\lambda_i}} u_j}{\sum_{e_j \in (P_{sv} \Delta P_{st})\setminus E^{\lambda_i}} w_j^{-1}}\}. \quad (4.4)$$

The second phase computes the optimal objective value of the problem (4.2) by using (4.4). We are now ready to state formally our proposed algorithm for solving the problem (4.2).

Algorithm 4.5. Input: A tree $T(V,E)$ with the edge length vector $\mathbf{c}$, the lower bound l_j and the upper bound u_j for each $e_j \in E$.

Step 1: Sort elements of $\{w_j l'_j : e_j \in E\backslash P_{st}\} \cup \{w_j u_j : e_j \in P_{st}\} \cup \{0\}$ in increasing order where $l'_j = \min\{l_j, c_j\}$ for each $e_j \in E$. Suppose that $\lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_m$ is the sorted list.

Step 2: If $\mathbf{x}^{\lambda_m}$ does not satisfy some constraints (4.2b), then the problem (4.2) is infeasible and stop (see Corollary 4.4).

Step 3: If $\mathbf{x}^{\lambda_0}$ satisfies the constraints (4.2b), then this solution is an optimal solution to the problem (4.2) and stop.

Step 4: Set $i_{lower} = 0$ and $i_{upper} = m$.

Step 5: Set $i = [\frac{i_{lower}+i_{upper}}{2}]$. If the solution $\mathbf{x}^{\lambda_i}$ satisfies the constraints (4.2b), then set $i_{upper} = i$. Otherwise, set $i_{lower} = i$. Repeat this step until $i_{upper} - i_{lower} \leq 1$.

Step 6: Set $i = i_{upper}$. Compute $\lambda^* \in (\lambda_{i-1}, \lambda_i]$ by using (4.4) and stop.

Output: If the problem (4.2) is feasible, then the solution $\mathbf{x}^{\lambda^*}$ is an optimal solution to the problem with the objective function λ^*.

We now analyze the complexity of the algorithm. The number of iterations of Step 5 is $O(\log n)$ due to the binary search technique. Furthermore, the feasibility of $\mathbf{x}^{\lambda_i}$ can be checked in $O(n^2)$ time. Thus, the running time of Step 5 is $O(n^2 \log n)$. On the other hand, Step 6 computes λ^* in $O(n^2)$ time. We have thus established the following result.

Theorem 4.6. *Algorithm 4.5 solves the problem (4.2) in $O(n^2 \log n)$ time.*

5 Inverse eccentric vertex problem under the sum-type Hamming distance

Assume that $G(V,E,\mathbf{c})$ is a network together with nonnegative bound vectors $\mathbf{l}, \mathbf{u}$ and a nonnegative weight vector $\mathbf{w}$ defined on edges of G. In this section, we consider the inverse eccentric vertex problem under the sum-type Hamming distance formulated as follows:

$$\min \quad z = \sum_{e_j \in E} w_j H(c_j, \hat{c}_j) \tag{5.1a}$$

$$s.t. \quad d_{\hat{\mathbf{c}}}(v) \leq d_{\hat{\mathbf{c}}}(t) \quad \forall v \in V, \tag{5.1b}$$

$$\max\{0, c_j - l_j\} \leq \hat{c}_j \leq c_j + u_j \quad \forall e_j \in E, \tag{5.1c}$$

where the notations are defined as in the problem (1.1).

Here, we discuss the complexity of the problem (5.1). The decision version of this problem is defined in the following.

Inverse Eccentric Decision (IED) problem:

Instance: An undirected network $G(V,E,\mathbf{c})$ with two specified vertices s and t, a lower bound vector $\mathbf{l} \geq 0$, an upper bound vector $\mathbf{u} \geq 0$, a penalty vector $\mathbf{w} \geq 0$ and a nonnegative number K.

Question: Is there a new length vector $\hat{\mathbf{c}}$ that is feasible to the problem (5.1) and $\sum_{e_j \in E} w_j H(c_j, \hat{c}_j) \leq K$?

Theorem 5.1. *The IED problem is NP-complete even on path networks.*

Proof. In order to prove the NP-completeness of the IED problem, we show that the Knapsack Decision (KD) problem, which is known to be NP-complete [16], is reduced to this problem. The KD problem is stated as follows:
Knapsack Decision (KD) problem:

Instance: A set of n items, a nonnegative size s_j and a nonnegative number p_j for each item j, two nonnegative numbers S and P.

Question: Does there exist a subset I of items such that $\sum_{j \in I} s_j \leq S$ and $\sum_{j \in I} p_j \geq P$?

It is obvious that the IED problem belongs to the class NP. For each instance of the KD problem, we construct an instance of the IED problem in the following manner. The vertex set is $V = \{0, 1, 2, \ldots, n+1\}$ in which the vertices 0 and $n+1$ are respectively the origin and the destination. The edge set is $E = \{(j-1, j) : j = 1, 2, \ldots, n\} \cup \{(0, n+1)\}$. Each edge $(j-1, j)$, $j = 1, 2, \ldots, n$, has a length of s_j, a weight of p_j, a lower bound of s_j and its upper bound is zero. The length of $(0, n+1)$ is equal to S and its penalty is 0. We set both the lower and the upper bounds of $(0, n+1)$ equal to zero and $K = \sum_{j=1}^{n} p_j - P$. Figure 3.a illustrates this reduction.

Suppose that I is a certificate of a yes instance of the KD problem. It is shown that the vector $\hat{\mathbf{c}}$ defined by

$$\hat{c}_{ij} = \begin{cases} S & (i,j) = (0, n+1), \\ s_j & i = j-1 \text{ and } j \in I, \\ 0 & i = j-1 \text{ and } j \notin I, \end{cases} \qquad \forall (i,j) \in A,$$

is a certificate to the corresponding instance of the IED problem. By construction, we have

$$d_{\hat{\mathbf{c}}}(0, n) = \sum_{j=1}^{n} \hat{c}_{j-1,j} = \sum_{j \in I} s_j \leq S = d_{\hat{\mathbf{c}}}(0, n+1).$$

This guarantees that the destination vertex $n+1$ is an eccentric vertex of 0. On the other hand, we observe that

$$\sum_{e_j \in A} w_j H(c_j, \hat{c}_j) = \sum_{j \notin I} p_j = \sum_{j=1}^{n} p_j - \sum_{j \in I} p_j \leq \sum_{j=1}^{n} p_j - P = K.$$

Now, suppose that $\hat{\mathbf{c}}$ is a certificate of a yes instance of the introduced IED problem. Since $l_{0,n+1} = u_{0,n+1} = 0$, it follows that $\hat{c}_{0,n+1} = c_{0,n+1}$. Let I be a subset of items that $j \in I$ if and only if $\hat{c}_{j-1,j} = c_{j-1,j}$ for each $j = 1, 2, \ldots, n$. It can be easily seen that the set I is a certificate of the corresponding KD problem. This completes the proof. □

Since any path graph can be considered as a subgraph of trees and cycles, Theorem 5.1 implies that the problem (5.1) is also NP-hard on trees and cycles. This result is interesting because the problem under the l_1 norm on trees and cycles is polynomially solvable [21]. However, the reduction of Theorem 5.1 does not prove that the problem (5.1) is NP-hard in the strong sense. This is the subject of the following theorem.

Before we state the next theorem, we give some relevant definitions. Suppose that x is a

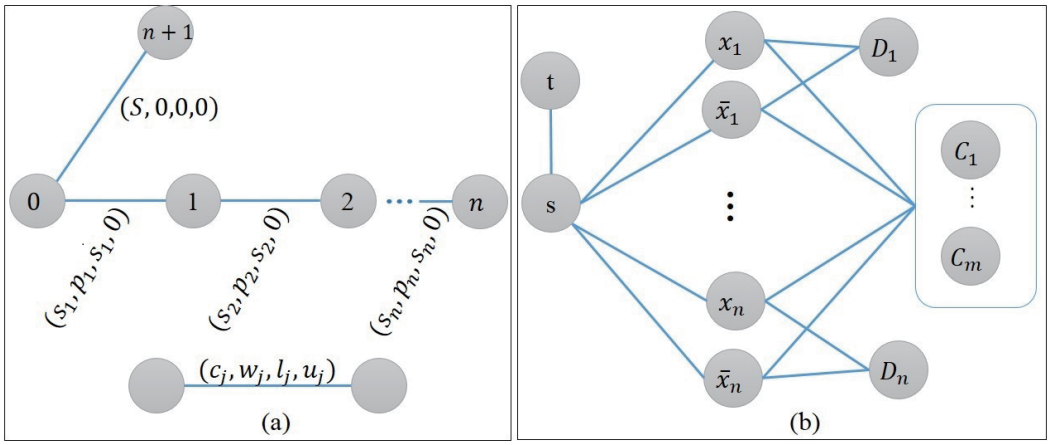

Figure 3: (a) An IED instance constructed from the KD problem; (b) An IED instance constructed from the satisfiability problem.

boolean variable. We denote by $\bar{x}$ the negation of boolean variable x. Given a set of boolean variables $X = \{x_1, x_2, \ldots, x_n\}$, a truth assignment for X is a function $t : X \to \{false, true\}$. A literal is either a boolean variable or its negation. A clause on X consists of some literals over X, and it is true under a truth assignment if and only if at least one of its members is true under that assignment. Note that if a clause contains x_j and its negation, then it is true for each truth assignment. Thus, we can assume that any clause does not contain both of x_j and $\bar{x}_j$.

Theorem 5.2. *The IED problem is strongly NP-complete even on bipartite networks.*

Proof. The proof is based on a reduction from the satisfiability problem defined as follows:
Satisfiability (SAT) problem:

Instance: A set $X = \{x_1, x_2, \ldots, x_n\}$ of boolean variables, m clauses $C = \{C_1, C_2, \ldots, C_m\}$ on X.

Question: Is there a truth assignment for X such that all the clauses in C are true?

The SAT problem was the first known NP-complete problem [10]. For a given instance of the SAT problem, we construct an instance of the IED problem in the following way.

- For each boolean variable x_j, the network contains three vertices x_j, $\bar{x}_j$ and D_j. In addition, for each clause C_i, the network has a vertex C_i. We also add two vertices s and t to the network which are respectively the origin and the destination. Thus, the vertex set is
$$V = \bigcup_{j=1}^{n} \{x_j, \bar{x}_j, D_j\} \cup \bigcup_{i=1}^{m} C_i \cup \{s, t\}.$$
- For each $j = 1, 2, \ldots, n$, the network contains four edges $(s, x_j), (s, \bar{x}_j), (x_j, D_j)$ and $(\bar{x}_j, D_j)$. For every $i = 1, 2, \ldots, n$ and every $j = 1, 2, \ldots, n$, if $x_j \in C_i$, then edge (x_j, C_i) is added to the network. If $\bar{x}_j \in C_i$, then edge $(\bar{x}_j, C_i)$ is added to the network. Thus, $E = \bigcup_{j=1}^{n} \{(s, x_j), (s, \bar{x}_j), (x_j, D_j), (\bar{x}_j, D_j)\} \cup \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} \{(x_j, C_i) : x_j \in C_j\} \cup \bigcup_{i=1}^{m} \bigcup_{j=1}^{n} \{(\bar{x}_j, C_i) : \bar{x}_j \in C_j\}$.

The length of edges (s, x_j) and $(s, \bar{x}_j)$ is equal to 1 and the length of all the other edges is zero.

- Each edge has a weight of 1.
- The upper bound of all edges is 1. The lower bound of edges (s, x_j) and $(s, \bar{x}_j)$ is 1 and the lower bound of all the other edges is zero.
- We set $K = n$.

Fiqure 3.b illustrates this reduction. Obviously, the introduced network is bipartite. By noting the values of lower and upper bounds, we can only modify the length of edges (s, x_j) and $(s, \bar{x}_j)$, $j = 1, \ldots, n$, to make t form an eccentric vertex of s.

Suppose that a yes instance of the SAT is given and $\mathbf{x}^0$ is its certificate. Consider the solution of the corresponding instance of the IED problem defined in the following manner: for each $j = 1, \ldots, n$, if $x_j^0 = true$, then the length of (s, x_j) is decreased to 0, and otherwise, the length of $(s, \bar{x}_j)$ is decreased to 0. Obviously, the number of modified arcs is n. To prove that t is an eccentric vertex of s, we must show that the distance from s to any other vertex is zero. The distance from s to each vertex D_j equals to zero because one of the two edges (s, x_j) and $(s, \bar{x}_j)$ is modified. The shortest path from s to each x_j is the path $s - x_j$ if $x_j^0 = true$ and otherwise, it is the path $s - \bar{x}_j - D_j - x_j$. Similarly, the shortest path from s to each $\bar{x}_j$ is the path $s - \bar{x}_j$ if $x_j^0 = false$ and it is the path $s - x_j - D_j - \bar{x}_j$ otherwise. For every $i = 1, 2, \ldots, m$, there exists at least one index $j \in \{1, 2, \ldots, n\}$ so that either $x_j \in C_i$ and $x_j^0 = true$ or $\bar{x}_j \in C_i$ and $x_j^0 = false$ because $\mathbf{x}^0$ is a solution of the SAT problem. In the former, the path $s - x_j - C_i$ is the shortest path with zero length and in the latter, the path $s - \bar{x}_j - C_i$ is the shortest path with zero length.

Now, suppose that $\hat{\mathbf{c}}$ is a solution to a yes instance of the IED problem. Note that if there exists a $j \in \{1, \ldots, n\}$ so that both the edges (s, x_j) and $(s, \bar{x}_j)$ are not modified, then the length of the shortest path from s to D_j is equal to 1 and consequently, t is not an eccentric vertex of s. This fact guarantees that the solution $\hat{\mathbf{c}}$ has two properties: (1) the number of modified arcs of $\hat{\mathbf{c}}$ is exactly n; (2) for each $j = 1, 2, \ldots, n$, exactly one of two edges $\{(s, x_j)$ and $(s, \bar{x}_j)\}$ is modified. Define the boolean vector $\mathbf{x}^0$ as follows: For each $j = 1, \ldots, n$, if the length of (s, x_j) is modified, then $x_j^0 = true$ and otherwise, $x_j^0 = false$. It is easy to verify that $\mathbf{x}^0$ is a solution to the corresponding SAT instance. □

Note that the proof of Theorem 5.2 is also valid for the l_1 and l_2 norms. Hence, we obtain a further conclusion as follows.

Corollary 5.3. *The inverse eccentric vertex problem under the l_1 and l_2 norms is strongly NP-hard even on bipartite networks.*

6 Conclusion and further research

In this article, we considered the inverse eccentric vertex problem on trees. When the length modifications are measured by the weighted l_1 norm, we showed that the problem is reduced to a minimum cost flow problem on an auxiliary network. For the weighted l_∞ norm, we applied the binary search technique to present an efficient algorithm for solving the problem. For the weighted sum-type Hamming distance, we proved that the problem is NP-hard on trees and strongly NP-hard on bipartite networks.

For further researches, it will be meaningful to study the problem under other distances,

e.g., the bottleneck-type Hamming distance and also, design the efficient algorithms for solving the problem on other specific networks.

References

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

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

[3] R.K. Ahuja and J.B. Orlin, Combinatorial algorithms for inverse network flow problems, *Networks* 40 (2002) 181–187.

[4] M. Aman and J. Tayyebi, Capacity inverse minimum cost flow problem under the weighted Hamming distances, *Iranian Journal of Operations Research* 5 (2014) 12–25.

[5] M. Aman, H. Hassanpour and J. Tayyebi, A modified algorithm for solving the inverse minimum cost flow problem under the bottleneck-type hamming distance, *Bull. Transilv. Univ. Braşov Ser.C* 9 (2016) 97–110.

[6] M.S. Bazaraa, J. Jarvis and H.D. Sherali, Linear programming and network flows, John Wiley & Sons, 2011.

[7] M. Bezad and J.E. Simpson, Eccentric sequences and eccentric sets in graphs, *Discrete Math.* 16 (1976) 178–193.

[8] D. Burton and P.L. Toint, On an instance of the inverse shortest paths problem, *Math. Program.* 53 (1992) 45–61.

[9] D. Burton and P.L. Toint, On the use of an inverse shortest paths algorithm for recovering linearly correlated costs, *Math. Program.* 63 (1994) 1–22.

[10] S.A. Cook, The Complexity of Theorem-Proving Procedures, In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 1971, pp. 151–158.

[11] Deaconu A (2008) The inverse maximum flow problem with lower and upper bounds for the flow. *Yugosl. J. Oper. Res.* 18: 13–22

[12] M. Demange and J. Monnot, An introduction to inverse combinatorial problems. In: Vangelis Th. Paschos, Paradigms of Combinatorial Optimization (Problems and New approaches), Wiley, London-Hoboken, 2010.

[13] Ç Güler and H. Hamacher, Capacity inverse minimum cost flow problem, *J. Comb. Optim.* 19 (2010) 43–59.

[14] C. Heuberger, Inverse optimization: A survey on problems, methods, and results, *J. Comb. Optim.* 8 (2004) 329–361.

[15] Y. Jiang, L. Liu, B. Wuc and E. Yao, Inverse minimum cost flow problems under the weighted Hamming distance, *European J. Oper. Res.* 207 (2010) 50–54.

[16] R.M. Karp, Reducibility among combinatorial problems, In: Complexity of computer computations, New York: Plenum Press, 1972, pp. 85–103.

[17] L. Liu and E. Yao, Weighted inverse maximum perfect matching problems under the Hamming distance, *J. Global Optim.* 55 (2013) 549–557.

[18] L. Liu, Y. Chen, B. Wu and E. Yao, Weighted inverse minimum cut problem under the sum-type hamming distance, In: Frontiers in Algorithmics and Algorithmic Aspects in Information and Management, Springer Berlin Heidelberg, 2012, pp. 26–35.

[19] L. Liu and J. Zhang, Inverse maximum flow problems under the weighted Hamming distance, *J. Comb. Optim.* 12 (2006) 395–408.

[20] L. Liu and E. Yao, Capacity inverse minimum cost flow problems under the weighted Hamming distance, *Optim. Lett.* (2015) doi:10.1007/s11590-015-0919-y.

[21] M. Mneimneh and K. Sakallah, Computing vertex eccentricity in exponentially large graphs: QBF formulation and solution, In: Proceedings of 6th international conference SAT03, volume 2919 of LNCS, 2003.

[22] K.T. Nguyen and A. Chassein, Inverse eccentric vertex problem on networks, *CEJOR Cent. Eur. J. Oper. Res.* 23 (2015) 687–698.

[23] K.B. Reid and W. Gu, Peripheral and eccentric vertices in graphs, *Graphs Combin.* 8 (1992) 361–375.

[24] A. Tarantola, Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation, Elsevier, Amsterdam, 1978.

[25] J. Tayyebi and M. Aman, Note on "inverse minimum cost flow problems under the weighted hamming distance", *European J. Oper. Res.* 234 (2014) 916–920.

[26] J. Tayyebi and M. Aman, On inverse linear programming problems under the bottleneck-type weighted Hamming distance, *Discrete Appl. Math.* (2016) doi:10.1016/j.dam.2015.12.017.

[27] C. Yang, J. Zhang, and Z. Ma, Inverse maximum flow and minimum cut problems, *Optimization* 40 (1997) 147–170.

[28] J. Zhang and Z. Liu, Calculating some inverse linear programming problems, *J. Comput. Appl. Math.* 72 (1996) 261–273.

Manuscript received 20 February 2016
revised 14 July 2016
accepted for publication 4 August 2016

Javad Tayyebi
Department of Industrial Engineering
Birjand University of Technology, Birjand, Iran
E-mail address: javadtayyebi@birjandut.ac.ir; javadtayyebi@birjand.ac.ir

Massoud Aman
Department of Mathematics, Faculty of Mathematical
Sciences and Statistics, University of Birjand, Birjand, Iran
E-mail address: mamann@birjand.ac.ir