J Glob Optim (2009) 43:83–95
DOI 10.1007/s10898-008-9294-x

Constrained inverse min–max spanning tree problems under the weighted Hamming distance

Longcheng Liu · Qin Wang

Received: 28 July 2007 / Accepted: 7 March 2008 / Published online: 21 March 2008

Abstract In this paper, we consider the constrained inverse min–max spanning tree problems under the weighted Hamming distance. Three models are studied: the problem under the bottleneck-type weighted Hamming distance and two mixed types of problems. We present their respective combinatorial algorithms that all run in strongly polynomial times.

Keywords Min–max spanning tree · Inverse problems · Hamming distance · Strongly polynomial algorithms

1 Introduction

Let $G = (V, E, c)$ be a connected graph, where $V = \{1, 2, \ldots, n\}$ is the node set, $E = \{e_1, e_2, \ldots, e_m\}$ is the edge set and c is the edge cost vector defined on E. Let Γ denote the collection of all spanning trees of G. For a spanning tree $T \in \Gamma$, write $c^b(T) = \max\{c(e)|e \in T\}$ and call it the cost of T. The min–max spanning tree is to find a $T^* \in \Gamma$ such that $c^b(T^*) = \min\{c^b(T)|T \in \Gamma\}$. It is known that min–max spanning tree problem can be solved in strongly polynomial time [1].

Conversely, an inverse min–max spanning tree problem is to modify the edge cost vector as little as possible such that a given spanning tree becames a min–max spanning tree. Yang et al. [2] showed that the inverse min–max spanning tree problem and the inverse maximum capacity path problem under l_1 and l_∞ norms are strongly polynomial time solvable, where the modification cost is measured by l_1 and l_∞ norms. Liu et al. [3] showed that the inverse min–max spanning tree problem under the weighted sum–type Hamming distance is

This research is supported by the National Natural Science Foundation of China (Grant No. 10601051).

L. Liu (✉)
Department of Mathematics, Zhejiang University, Hangzhou, China
e-mail: llcly@126.com

Q. Wang
Department of Mathematics, China Jiliang University, Hangzhou, China
e-mail: wq@cjlu.edu.cn

strongly polynomial time solvable. In this paper, we consider the constrained inverse min–max spanning tree problems under the weighted Hamming distance, in which we measure the modification cost by the weighted Hamming distance. Three variations will be discussed.

Let each edge e_i have an associated weight $w_i \geq 0$, and let w denote the edge weight vector. Let T^0 be a given spanning tree of graph G. Then for the inverse min–max spanning tree problem under the weighted bottleneck-type Hamming distance, we look for a new cost vector $d = (d_1, d_2, \ldots, d_m)$ such that

(a) T^0 is a min–max spanning tree with respect to d;
(b) for each $e_i \in E$, $-l_i \leq d_i - c_i \leq u_i$, where $l_i, u_i \geq 0$ are given lower and upper modification bounds;
(c) the maximum modification cost among all edges, i.e., $\max_{e_i \in E}\{w_i H(c_i, d_i)\}$, is minimized, where $H(c_i, d_i)$ is the Hamming distance between c_i and d_i, i.e., $H(c_i, d_i) = 0$ if $c_i = d_i$ and 1 otherwise.

We also consider two mixed type cases. For the first mixed type problem, we look for a new cost vector $d = (d_1, d_2, \ldots, d_m)$ such that in addition to the requests (a), (b) and (c), it also satisfies

(d) the total modification cost for all edges cannot exceed a given upper bound $M > 0$, i.e., $\sum_{i=1}^{m} w_i H(c_i, d_i) \leq M$.

For the second mixed type problem, we look for a new cost vector $d = (d_1, d_2, \ldots, d_m)$ such that in addition to the requests (a) and (b), it also satisfies

(c′) the total modification cost for all edges, i.e., $\sum_{i=1}^{m} w_i H(c_i, d_i)$, is minimized;
(d′) the maximum modification cost among all the edges cannot exceed a given upper bound $M > 0$, i.e., $\max_{e_i \in E}\{w_i H(c_i, d_i)\} \leq M$.

In general, for an inverse combinatorial optimization problem, a feasible solution is given which is not optimal under the current parameter values, and it is required to modify some parameters with minimum modification cost such that the given feasible solution becomes an optimal solution. A lot of such problems have been well studied when the modification cost is measured by (weighted) l_1, l_2, and l_∞ norms. Readers may refer to the survey paper [4] and papers cited therein. Recently, inverse problems under the weighted Hamming distance also received attention. In fact the weighted sum-type Hamming distance represents the weighted number of modifications. It corresponds to the situation in which we might care about only whether the parameter of an arc is changed, but without considering the magnitude of its change as long as the adjustment is restricted to a certain interval. Noting that not like the l_1, l_2 and l_∞ norms which are all convex and continuous about the modification, the Hamming distance $H(\cdot, \cdot)$ is discontinuous and nonconvex, which makes the known methods for l_1, l_2 and l_∞ norms unable to be applied directly to the problems under such distance measure.

He et al. [5] discussed the inverse minimum spanning tree problem under the weighted sum-type Hamming distance. For both unbounded and bounded cases, they presented strongly polynomial algorithms with a time complexity $O(n^3 m)$. Here n and m are the numbers of nodes and edges, respectively, in a given undirected network. He et al. [6] further discussed the inverse minimum spanning tree problem under the weighted bottleneck-type Hamming distance. For the unbounded case, they presented algorithms with a time complexity $O(nm)$, and for the constrained case, they presented an algorithm with a time complexity $O(n^3 m \log m)$. Duin and Volgenant [7] also discussed the unbounded case of the inverse minimum spanning

tree problem under the bottleneck-type Hamming distance. They presented an improved algorithm with a time complexity $O(n^2)$. They further extended the results to the inverse shortest path tree problem and the linear assignment problem. Zhang et al. [8] considered the center location improvement problem under the weighted Hamming distance. For the bounded case, they showed that even under the unweighted sum-type Hamming distance, achieving an algorithm with a worst-case ratio $O(\log n)$ is strongly NP-hard, but under the weighted bottleneck-type Hamming distance, a strongly polynomial algorithm with a time complexity $O(n^2 \log n)$ is available. Yang et al. [9] discussed inverse sorting problems under the weighted sum-type Hamming distance. For both unbounded and bounded cases, they presented strongly polynomial algorithms. Liu et al. [10] discussed inverse maximum flow problems under the weighted Hamming distance, for both sum-type and bottleneck-type, they presented strongly polynomial algorithms. Liu et al. [11] discussed inverse minimum cut problems under the weighted bottleneck-type Hamming distance, they presented strongly polynomial algorithm. Guan et al. [12] discussed inverse bottleneck optimization problems under weighted Hamming distance, for the discussed problems they presented strongly polynomial algorithms.

The paper is organized as follows. Section 2 contains some preliminary results. Sections 3 and 4 consider the problem under the weighted bottleneck-type Hamming distance and the two mixed type problems, respectively. We show that all these problems can be solved by strongly polynomial algorithms. Some final remarks are made in Sect. 5.

In the following, for each edge set Ω we define $w^s(\Omega) = \sum_{e_i \in \Omega} w_i$, $w^b(\Omega) = \max_{e_i \in \Omega} w_i$ and use similar notations $c^s(\Omega)$ and $c^b(\Omega)$ for vector c (here s and b stand for 'sum' and 'bottleneck', respectively).

2 Preliminary results

For the original min–max spanning tree problem, the following result is straightforward.

Lemma 2.1 *A spanning tree T of G is a min–max spanning tree under a cost vector c if and only if G becomes disconnected after deleting the edges whose costs are not less than $c^b(T)$.*

Now we consider the constrained inverse min–max spanning tree problem under the Hamming distance. The general inverse min–max spanning tree problem under the weighted bottleneck-type Hamming distance can be formulated as follows.

$$\begin{aligned} &\min \max_{e_i \in E} \{w_i H(c_i, d_i)\} \\ &\text{s.t. } T^0 \text{ is a min–max spanning tree of } G(V, E, d); \\ &-l_i \le d_i - c_i \le u_i, \quad 1 \le i \le m. \end{aligned} \tag{1}$$

Let T^* be a min–max spanning tree under the cost vector c, and assume $c^b(T^0) > c^b(T^*)$ for otherwise we need to do nothing.

Remark Note that the lemmas in this section also hold for other types of inverse problems discussed in later sections of the paper, and the proofs are similar.

Lemma 2.2 *There exists an optimal solution d^* of problem (1) such that $c^b(T^0) \ge d^{*b}(T^0)$.*

Proof In fact, if $c^b(T^0) < d^{*b}(T^0)$, then we can construct a new cost vector $\overline{d}$ by the following way:

$$\overline{d}_i = \begin{cases} c^b(T^0), & \text{if } e_i \in T^0 \text{ and } d_i^* > c^b(T^0), \\ d_i^*, & \text{otherwise.} \end{cases}$$

It is clear that $\overline{d}^b(T^0) = c^b(T^0) < d^{*b}(T^0)$. By Lemma 2.1, the graph $G = (V, E, d^*)$ becomes disconnected after deleting the edges whose cost satisfy $d_i^* \geq d^{*b}(T^0)$. Hence the graph $G = (V, E, \overline{d})$ becomes disconnected after deleting the edges whose cost satisfy $\overline{d}_i \geq \overline{d}^b(T^0)$, which means T^0 is a min–max spanning tree of graph $G = (V, E, \overline{d})$, i.e., $\overline{d}$ is a feasible solution of problem (1).

However, by the definition of $\overline{d}$ and the definition of Hamming distance we have

$$\max_{e_i \in E}\{w_i H(c_i, d_i^*)\} \geq \max_{e_i \in E}\{w_i H(c_i, \overline{d}_i)\}.$$

If $\max_{e_i \in E}\{w_i H(c_i, d_i^*)\} > \max_{e_i \in E}\{w_i H(c_i, \overline{d}_i)\}$, then d^* cannot be an optimal solution of problem (1), a contradiction. Hence, $\overline{d}$ is another optimal solution of problem (1), but it satisfies $\overline{d}^b(T^0) = c^b(T^0)$. The lemma holds. □

Based on Lemma 2.2, the following result is straightforward:

Lemma 2.3 *There exists an optimal solution d^* of problem (1) satisfies:*

(a) $d_i^* = c_i$ *if* $c_i \geq d^{*b}(T^0)$ *and* $e_i \in E\backslash T^0$;
(b) $d_i^* \geq c_i$ *if* $c_i < d^{*b}(T^0)$ *and* $e_i \in E\backslash T^0$;
(c) $d_i^* = d^{*b}(T^0)$ *if* $c_i \neq d_i^*$ *and* $e_i \in T^0$.

Moreover, we have the following lemma:

Lemma 2.4 *There exists an optimal solution d^* of problem (1) satisfies:*

(a) $d_i^* \leq d^{*b}(T^0)$ *if* $d_i^* < c_i$;
(b) $d_i^* \geq d^{*b}(T^0)$ *if* $d_i^* > c_i$.

Proof Suppose d^* is an optimal solution satisfies Lemma 2.3.

If $d_i^* < c_i$, then by the Lemma 2.3, we have $e_i \in T^0$, hence $d_i^* \leq d^{*b}(T^0)$, i.e., (a) holds.

Now let us consider (b). First, if $e_i \in T^0$, then by Lemma 2.3, we have $d_i^* = d^{*b}(T^0)$. Second, let us consider $e_i \in E\backslash T^0$. If (b) is not true, i.e., there exists an edge $e_k \in E\backslash T^0$ and $c_k < d_k^*$ such that $d_k^* < d^{*b}(T^0)$.

Define $\overline{d}$ as

$$\overline{d}_i = \begin{cases} c_i, & \text{if } i = k, \\ d_i^*, & \text{otherwise.} \end{cases}$$

Note that the difference between d^* and $\overline{d}$ is only on the edge e_k, so $d^{*b}(T^0) = \overline{d}^b(T^0)$. By Lemma 2.1, the graph $G = (V, E, d^*)$ becomes disconnected after deleting the edges whose cost satisfy $d_i^* \geq d^{*b}(T^0)$, combining with the fact that $\overline{d}_k = c_k < d_k^* < d^{*b}(T^0) = \overline{d}^b(T^0)$, we know the graph $G = (V, E, \overline{d})$ becomes disconnected after deleting the edges whose cost satisfy $\overline{d}_i \geq \overline{d}^b(T^0)$, which means that $\overline{d}$ is a feasible solution of problem (1). However, by the definition of $\overline{d}$ we have

$$\max_{e_i \in E}\{w_i H(c_i, d_i^*)\} \geq \max_{e_i \in E}\{w_i H(c_i, \overline{d}_i)\}.$$

If $\max_{e_i \in E}\{w_i H(c_i, d_i^*)\} > \max_{e_i \in E}\{w_i H(c_i, \overline{d}_i)\}$, then d^* cannot be an optimal solution of problem (1), a contradiction. Hence, $\overline{d}$ is another optimal solution of problem (1), but it satisfies $\overline{d}_k = c_k$. And by repeating the above procedure, we can conclude that there exists an optimal solution d^* of problem (1) such that $d_i^* \geq d^{*b}(T^0)$ if $d_i^* > c_i$ and $e_i \in E\backslash T^0$. From the above analysis, we know (b) holds. □

Here we first give a range for the value $d^{*b}(T^0)$. First, by Lemma 2.2, we have $c^b(T^0) \geq d^{*b}(T^0)$. On the other hand, since there are lower bounds on the reduction of costs, the smallest possible value of $d^{*b}(T^0)$ is $\underline{d} = \max\{c_i - l_i | e_i \in T^0\}$. So we have $\underline{d} \leq d^{*b}(T^0) \leq c^b(T^0)$. Second, we say that the value $d^{*b}(T^0)$ must be one of the value in $\{c_i | e_i \in T^0\} \cup \{\underline{d}\}$. If not, i.e., $d^{*b}(T^0) > \underline{d}$ and there exists two edges e_g and e_h such that $c_g < d^{*b}(T^0) < c_h$. In this case, define $\overline{d}$ as

$$\overline{d}_i = \begin{cases} c_g, & \text{if } d_i^* = d^{*b}(T^0), \\ d_i^*, & \text{otherwise.} \end{cases}$$

It is clear that $\overline{d}$ is also an optimal solution and the optimal value is the same by the definition of Hamming distance. Combining the above analysis we know the value $d^{*b}(T^0)$ must be one of the value in $P = \{\{c_i | e_i \in T^0\} \cup \{\underline{d}\}\} \cap [\underline{d}, c^b(T^0)]$. Then we express the different values in P as: $p_1 > p_2 > \cdots > p_\eta$.

3 Problem under the weighted bottleneck-type Hamming distance

The problem considered in this section is the inverse min–max spanning tree problem under the weighted bottleneck-type Hamming distance which can be formulated as problem (1).

Before we consider how to solve the problem (1) directly, let us consider a *restricted version* of the inverse min–max spanning tree problem under the bottleneck-type Hamming distance. That is, for a given value $p \in P$, we first consider how to make T^0 a min–max spanning tree under a cost vector d^p such that $d^{pb}(T^0) = p$, and d^p satisfies the bound restrictions and makes the objective value minimum. We may call this restricted version of the inverse min–max spanning tree problem *the inverse min–max spanning tree problem under the bottleneck-type Hamming distance with value p*.

First, let $T^0(p) = \{e_i \in T^0 | c_i > p\}$, $\overline{T^0}(p) = \{e_i \in T^0 | c_i = p\}$. Clearly, for each edge $e_i \in T^0(p)$, we need to reduce their costs to let the maximum cost on T^0 be equal to p. Due to Lemma 2.3 for each edge $e_i \in T^0(p)$ we have $d_i^p = p$ and the associate objective value is $w^b(T^0(p)) = \max_{e_i \in T^0(p)} w_i$. And by Lemma 2.3, for each edge $e_i \in \overline{T^0}(p)$ we do not need to change its cost.

Second, by Lemma 2.3, for each edge $e_i \in E \backslash T^0$ such that $c_i \geq p$, we do not need to change its cost.

Third, let $E(p) = \{e_i \in E | c_i < p\}$. Consider the graph $G(p) = (V, E(p))$. If $G(p)$ is not connected, we know that T^0 is already a min–max spanning tree with respect to the modified weight d^p and $d^{pb}(T^0) = p$, where $d_i^p = p$ for $e_i \in T^0(p)$ and $d_i^p = c_i$ for $e_i \in E \backslash T^0(p)$. And the objective value of problem (1) with respect to p is $w^b(T^0(p)) = \max_{e_i \in T^0(p)} w_i$.

Thus we only need to consider the case that $G(p)$ is a connected graph. In this case, by Lemma 2.1, we need to increase the costs of some edges in graph $G(p)$ such that $G(p)$ becomes disconnected after deleting those edges. Now we introduce the following restricted minimum bottleneck-type weight edge cut problem:

Restricted minimum bottleneck-type weight edge cut problem(RMBWECP):

Find an edge set $\Pi \subseteq E(p)$ such that

(1) for each edge $e_i \in \Pi$, $c_i + u_i \geq p$;
(2) $G(p)$ becomes disconnected after deleting all edges in Π;
(3) $\max_{e_i \in \Pi} w_i$ is minimized.

Theorem 3.1 *If the RMBWECP is feasible, then an optimal solution of the restricted version of problem* (1) *is*

$$d_i^* = \begin{cases} p, & \text{if } e_i \in T^0(p) \cup \Pi^*, \\ c_i, & \text{otherwise,} \end{cases} \tag{2}$$

and the associate objective value is $w^b(T^0(p) \cup \Pi^*)$*, where* Π^* *is an optimal solution of RMBWECP. Otherwise, the restricted version of problem* (1) *is infeasible.*

Proof In the case that RMBWECP is feasible, we first prove that d^* defined by (2) is a feasible solution of the restricted version of problem (1). From the definition of $E(p)$ and the first and second constraints of RMBWECP, we know the graph G becomes disconnected after deleting the edges whose costs are not less than p, which indicates T^0 is a min–max spanning tree of graph $G(V, E, d^*)$, and it is clear $-l_i \le d_i^* - c_i \le u_i$. Thus d^* defined by (2) is a feasible solution of the restricted version of problem (1) and the associate objective value is $w^b(T^0(p) \cup \Pi^*)$.

Next we prove $w^b(T^0(p) \cup \Pi^*)$ is the minimum objective value, thus d^* is an optimal solution of the restricted version of problem (1). If not, there exists an optimal solution $\overline{d}$ of problem (1) such that

(a) $\overline{d}^b(T^0) = p$;
(b) $\max\limits_{e_i \in E}\{w_i H(c_i, \overline{d}_i)\} < w^b(T^0(p) \cup \Pi^*)$.

Let

$$\Omega = \{e_i \in E | \overline{d}_i \ne c_i\}. \tag{3}$$

Then by the definition of Hamming distance, (b) is equivalent to

$$\max_{e_i \in E}\{w_i H(c_i, \overline{d}_i)\} = \max_{e_i \in \Omega} w_i < w^b(T^0(p) \cup \Pi^*) = \max_{e_i \in (T^0(p) \cup \Pi^*)} w_i. \tag{4}$$

Based on the above analysis, we can see that $T^0(p) \subseteq \Omega$, thus (4) is equivalent to

$$\max_{e_i \in \Omega \backslash T^0(p)} w_i < \max_{e_i \in \Pi^*} w_i. \tag{5}$$

Moreover, we say $\Omega \backslash T^0(p)$ is a feasible solution of RMBWECP. In fact, it is clear $\Omega \backslash T^0(p) \subseteq E(p)$. Based on the analysis before, we know $\overline{d}_i > c_i$ for all $e_i \in \Omega \backslash T^0(p)$. And by Lemma 2.4 we know $\overline{d}_i \ge \overline{d}^b(T^0) = p$ for all $e_i \in \Omega \backslash T^0(p)$, which indicate $c_i + u_i \ge p$ for all $e_i \in \Omega \backslash T^0(p)$. At last we claim $G(p)$ becomes disconnected after deleting all edges in $\Omega \backslash T^0(p)$. Define $\widetilde{E}(p) := E(p) \backslash \{\Omega \backslash T^0(p)\}$ and $\overline{E}(p) := \{E(p) \cup T^0(p)\} \backslash \Omega$. If not, i.e., $G(V, \widetilde{E}(p))$ is connected. Since $T^0(p) \cap E(p) = \emptyset$, we know $\widetilde{E}(p) = E(p) \backslash \Omega \subseteq \overline{E}(p)$, so $G(V, \overline{E}(p))$ is connected. And thus $G(V, \{E(p) \cup T^0(p) \cup \{e_i \in E \mid c_i \ge p\}\} \backslash \{\Omega \cup \{e_i \in E \mid c_i \ge p\}\}) = G(V, E \backslash \{\Omega \cup \{e_i \in E \mid c_i \ge p\}\})$ is connected. Let $L = \{e_i \in E \mid \overline{d}_i \ge p\}$. By the above analysis we know $L \subseteq \{\Omega \cup \{e_i \in E \mid c_i \ge p\}\}$ and thus $G(V, E \backslash L)$ is connected. But since T^0 is a min–max spanning tree of $G(V, E, \overline{d})$, $G(V, E \backslash L)$ is disconnected by Lemma 2.1, a contradiction. So $G(p)$ becomes disconnected after deleting all edges in $\Omega \backslash T^0(p)$. Hence $\Omega \backslash T^0(p)$ is a feasible solution of RMBWECP. For Π^* is an optimal solution of RMBWECP, we know

$$\max_{e_i \in \Omega \backslash T^0(p)} w_i \ge \max_{e_i \in \Pi^*} w_i, \tag{6}$$

which contradicts with (5). So $w^b(T^0(p) \cup \Pi^*)$ is exactly the minimum objective value.

At the same time, the above analysis told us if $\overline{d}$ is a feasible solution of the restricted version of problem (1), then $\Omega \backslash T^0(p)$ is a feasible solution of RMBWECP, where Ω is defined in (3). Thus if the RMBWECP is infeasible, then the restricted version of problem (1) is infeasible too. □

Therefore, finding an optimal solution of the restricted version of problem (1) is equivalent to finding an optimal solution of RMBWECP. To solve RMBWECP in strongly polynomial time, we modify the graph $G(p) = (V, E(p))$ in the following way: the node set and the edge set are unchanged; and the value of each edge is set as

$$v_i = \begin{cases} w_i, & \text{if } e_i \in E(p) \text{ and } c_i + u_i \geq p, \\ W + 1, & \text{otherwise,} \end{cases} \tag{7}$$

where $W = \sum_{e_i \in E(p)} w_i$

Definition 3.1 The minimum bottleneck value cut problem in graph is to find a set of edges whose deletion make the graph disconnected and the bottleneck values of edges in the set is minimized.

Theorem 3.2 *Let Π^* be a minimum bottleneck value cut of $G(V, E(p), v)$ with a value $v^b(\Pi^*)$.*

(1) *If $v^b(\Pi^*) \leq W$, then Π^* must be an optimal solution of RMBWECP.*
(2) *If $v^b(\Pi^*) > W$, then RMBWECP has no feasible solution.*

Proof

(1) First, if $v^b(\Pi^*) \leq W$, then $v_i = w_i$ for all $e_i \in \Pi^*$, i.e., for each $e_i \in \Pi^*$, $e_i \in E(p)$ and $c_i + u_i \geq p$. And it is clear $G(p)$ becomes disconnected after deleting all edges in Π^*. So, Π^* is a feasible solution of RMBWECP.
Moreover, it is easy to see that Π^* is an optimal solution of RMBWECP. If not, suppose there exists an edge set Π' which is feasible to RMBWECP, and $w^b(\Pi') < w^b(\Pi^*)$. Then from (7), we have

$$v^b(\Pi') = \max_{e_i \in \Pi'} w_i = w^b(\Pi') < w^b(\Pi^*) = \max_{e_i \in \Pi^*} w_i = v^b(\Pi^*),$$

which contradicts the fact that Π^* is a minimum bottleneck value cut of $G(V, E(p), v)$.
(2) Suppose that $v^b(\Pi^*) > W$ but RMBWECP has a feasible solution Π'. From (7), we know that $v_i = w_i$ for all $e_i \in \Pi'$. It implies that the value of Π' satisfies $v^b(\Pi') < W$ (as $W = \sum_{e_i \in E(p)} w_i$), which contradicts the fact that Π^* is a minimum bottleneck value cut of $G(V, E(p), v)$ with a value $v^b(\Pi^*) > W$. □

Based on the above analysis, we only need to find a minimum bottleneck value cut of $G(V, E(p), v)$. As we do not see any reference giving explicitly an algorithm for the purpose, here we present how to solve this problem in detail.

First, for a specific pair of nodes s and t, we define an $[s, t]$ cut as a set of edges whose deletion from the graph disconnect the graph into two components that s and t are in different components respectively. Here we present an algorithm to determine the $[s, t]$ cut with minimum bottleneck value in detail:

Algorithm 1

Step 1 Find a maximum (sum) spanning tree T^* of $G(V, E(p), v)$.
Step 2 Form a subset of T^*:

$$Q = \{e \in T^* | \text{in } T^* \backslash e, s \text{ and } t \text{ are disconnected}\}.$$

Step 3 Find $e^* \in Q$ such that

$$v(e^*) = \min\{v(e) \mid e \in Q\}.$$

Step 4 Identify the partition of V obtained by $T^* \backslash e^*$ as Z and $\overline{Z} = V \backslash Z$, where $s \in Z$ and $t \in \overline{Z}$. Then form the bottleneck value $[s, t]$ cut $\Pi^* = \{Z, \overline{Z}\}$.

We now justify the algorithm.

Theorem 3.3 *The edge set Π^* resulted from Algorithm 1 is the $[s, t]$ cut with minimum bottleneck value.*

Proof Obviously Π^* is an $[s, t]$ cut, and the only common edge of T^* and Π^* is e^*. Also, by the property of maximum spanning tree, we know that

$$v^b(\Pi^*) = \max_{e \in \Pi^*} v(e) = v(e^*).$$

Let Π be an arbitrary $[s, t]$ cut. If $v^b(\Pi) < v^b(\Pi^*) = v(e^*)$, then we know $\Pi \cap Q = \emptyset$. It is clear that the edges in Q form the unique $s - t$ path in T^*. So when we delete all the edges in Π, s and t are still connected, a contradiction. Hence we obtain

$$v^b(\Pi) \geq v(e^*) = v^b(\Pi^*),$$

i.e., Π^* is the $[s, t]$ cut with minimum bottleneck value. □

It is clear that the main computation of Algorithm 1 is to find a max-sum spanning tree. So Algorithm 1 runs in $O(k + n \log n)$ time (see [13]), where k and n are the numbers of edges and nodes of $G(V, E(p), v)$. The definition of a cut implies that if an edge set is an $[s, t]$ cut, it is also a $[t, s]$ cut. This observation implies that we can find a minimum bottleneck value cut of $G(V, E(p), v)$ by invoking $n(n-1)/2$ applications of Algorithm 1, which means that finding a minimum bottleneck value cut of $G(V, E(p), v)$ can be done in polynomial time.

Now we are ready to give a full description of an algorithm to solve the restricted version of problem (1).

Algorithm 2

Step 1 For the graph $G = (V, E, c)$, the given spanning tree T^0 and the given value p, determine the graph $G(p) = (V, E(p))$. If $G(p)$ is not connected, stop and output an optimal solution d^* of the restricted version of problem (1) as

$$d_i^* = \begin{cases} p, & \text{if } e_i \in T^0(p), \\ c_i, & \text{otherwise,} \end{cases}$$

and the associated optimal value is $w^b(T^0(p))$. Otherwise go to Step 2.
Step 2 Construct graph $G(V, E(p), v)$ according to formula (7). Find a minimum bottleneck value cut Π^* of the graph $G(V, E(p), v)$. If the value of the minimum cut satisfies $v^b(\Pi^*) > W$, then the restricted version of problem (1) has no feasible solution, stop. Otherwise, go to Step 3.

Step 3 Output an optimal solution d^* of the restricted version of problem (1) as

$$d_i^* = \begin{cases} p, & \text{if } e_i \in T^0(p) \cup \Pi^*, \\ c_i, & \text{otherwise,} \end{cases} \tag{8}$$

and the associated objective value is $w^b(T^0(p) \cup \Pi^*)$.

It is clear that Step 1 to check whether $G(p)$ is connected or not takes $O(n)$ time. From the above analysis, Step 2 can be done in polynomial time. Hence, Algorithm 2 is a strongly polynomial algorithm.

Now we consider the problem (1). In Sect. 2, we range the possible value of $d^{*b}(T^0)$ as $P = \{p_1, p_2, \ldots, p_\eta\}$. And above we discussed for a given value $p \in P$, we can get an associated bottleneck modification weight in polynomial time. So we can give an algorithm to solve problem (1) in strongly polynomial time as follows:

Algorithm 3

Step 0 Let $i = 1$ and $I = \emptyset$.

Step 1 For value p_i, run Algorithm 2 to solve the restricted version of problem (1). If the restricted version problem is infeasible, then go to Step 2; otherwise, we denote the objective value get from Algorithm 2 as V_i and $I = I \cup \{i\}$, then go to Step 2.

Step 2 $i = i + 1$, if $i \leq \eta$, go back to Step 1; otherwise go to Step 3.

Step 3 Output an optimal objective value $\min\limits_{i \in I} V_i$.

If we call solving the restricted version problem as an iteration, then Algorithm 3 needs to run η iteration. From Sect. 2, we know $\eta \leq n$, combining with the analysis of Algorithm 2, Algorithm 3 is a strongly polynomial algorithm.

4 Two mixed type problems

In this section, we study two sum-type and bottleneck-type mixed problems, which can be formulated as follows:

$$\begin{aligned} &\min \max_{e_i \in E} \{w_i H(c_i, d_i)\} \\ &\text{s.t. } T^0 \text{ is a min–max spanning tree of } G(V, E, d); \\ &-l_i \leq d_i - c_i \leq u_i, \quad 1 \leq i \leq m. \\ &\sum_{i=1}^{m} w_i H(c_i, d_i) \leq M. \end{aligned} \tag{9}$$

$$\begin{aligned} &\min \sum_{i=1}^{m} w_i H(c_i, d_i) \\ &\text{s.t. } T^0 \text{ is a min–max spanning tree of } G(V, E, d); \\ &-l_i \leq d_i - c_i \leq u_i, \quad 1 \leq i \leq m. \\ &\max_{e_i \in E} \{w_i H(c_i, d_i)\} \leq M. \end{aligned} \tag{10}$$

To consider the problem (9), we introduce the following problem:

$$\begin{aligned} &\min \sum_{i=1}^{m} w_i H(c_i, d_i) \\ &\text{s.t. } T^0 \text{ is a min–max spanning tree of } G(V, E, d); \\ &-l_i \leq d_i - c_i \leq u_i, \quad 1 \leq i \leq m. \end{aligned} \tag{11}$$

The following lemma is trivial.

Lemma 4.1 *Problem* (9) *has a feasible solution if and only if that problem* (11) *has a feasible solution and the optimal value of problem* (11) *is not greater than* M.

So we consider problem (11) first. This problem is a sum-type inverse optimization problem and there is already following available result, for the detail, the reader may refer to paper [3]:

Algorithm 4 [3]

Step 1 For the graph $G = (V, E, c)$, the given spanning tree T^0 and the given value p (here the p is the same as discussed bedore), determine the graph $G(p) = (V, E(p))$. If $G(p)$ is not connected, stop and output an optimal solution d^* of the restricted version of problem (11) as

$$d_i^* = \begin{cases} p, & \text{if } e_i \in T^0(p), \\ c_i, & \text{otherwise,} \end{cases}$$

and the associated optimal value $w^s(T^0(p))$. Otherwise go to Step 2.

Step 2 Construct graph $G(V, E(p), v)$ according to formula (7). Find a minimum value cut Π^* of the graph $G(V, E(p), v)$. If the value of the minimum cut satisfies $v^s(\Pi^*) > W$, then the restricted version of problem (11) has no feasible solution, stop. Otherwise, go to Step 3.

Step 3 Output an optimal solution d^* of the restricted version of problem (11) as

$$d_i^* = \begin{cases} p, & \text{if } e_i \in T^0(p) \cup \Pi^*, \\ c_i, & \text{otherwise,} \end{cases} \tag{12}$$

and the associated objective value is $w^s(T^0(p) \cup \Pi^*)$.

We are going to give an algorithm for solving problem (9). We first explain the main idea of the algorithm.

First, by comparing problems (1) and (9), we see that they have the same objective function and the first two groups of constraints, but problem (9) has one more constraint. Hence, if we denote by d^1 and d^* the optimal solutions of problems (1) and (9), respectively, we must have $\max\limits_{e_i \in E}\{w_i H(c_i, d_i^1)\} \le \max\limits_{e_i \in E}\{w_i H(c_i, d_i^*)\}$, i.e., the optimal value of (1) can be taken as a lower bound for the optimal value of (9). Clearly, the optimal values of (1) and (9) are actually weights of two edges. We denote this lower bound as $w_{\underline{t}+1}$.

Second, if d^2 is an optimal solution of problem (11) with the objective function value being not greater than M, then from Lemma 4.1 this d^2 is feasible to problem (9), and thus $\max\limits_{e_i \in E}\{w_i H(c_i, d_i^2)\}$ must be an upper bound for the optimal value of problem (9). We denote this upper bound as $w_{\bar{t}}$.

So the optimal value of problem (9) must be one of the cost values in the interval $[w_{\underline{t}+1}, w_{\bar{t}}]$, or for convenience, in the half open and half closed interval $(w_{\underline{t}}, w_{\bar{t}}]$.

Third, in order to determine the minimum weight of problem (9) from the interval $(w_{\underline{t}}, w_{\bar{t}}]$ quickly, we may use the bisection method. Take a cost value, say $w_{t'}$, which is in the quite middle of the above interval. Then we ask: whether the minimum value of problem (9) is not greater than $w_{t'}$? If yes, the search interval can be reduced to $(w_{\underline{t}}, w_{t'}]$, otherwise to $(w_{t'}, w_{\bar{t}}]$. And this question can be answered by computing problem (11) with the revision that if the weight w_τ of edge e_τ is greater than $w_{t'}$, then set $d_\tau = c_\tau$ (i.e., let the corresponding l_τ and

u_τ in the second group of constraints in (11) be 0), and then checking if the minimum value of the restricted problem (11) is not greater than M. Repeating the process several times until the interval is reduced to include a single cost value, i.e., the right endpoint, which is the optimal value of (9).

Let $\lceil a \rceil$ be the smallest integer which is not less than a. The algorithm can be formally described as follows.

Algorithm 5

Step 1 Let $w_0 = -1$, rearrange the weights $w_0, w_1, \ldots, w_m$ in an increasing order. Then we express their different values as: $-1 = w_{j_1} < w_{j_2} < \cdots < w_{j_k}$.

Step 2 Call Algorithm 3 to solve problem (1). If the algorithm finds problem (1) infeasible, then output that the problem (9) is infeasible, stop. Otherwise, let the optimal solution of (1) be d^1 and find the index $\underline{t}$ such that $\max\limits_{e_i \in E}\{w_i H(c_i, d_i^1)\} = w_{j_{\underline{t}+1}}$.

Step 3 Call Algorithm 4 to solve problem (11). If the algorithm finds that the optimal value of problem (11) is greater than M, then output that problem (9) is infeasible, stop. Otherwise, let the optimal solution of (11) be d^2 and find the index $\bar{t}$ such that $\max\limits_{e_i \in E}\{w_i H(c_i, d_i^2)\} = w_{j_{\bar{t}}}$.

Step 4 If $\bar{t} - \underline{t} = 1$, then output that d^2 is an optimal solution of problem (9) with the minimum cost $w_{j_{\bar{t}}}$, stop. Otherwise, we have $\bar{t} - \underline{t} > 1$, and go to Step 5.

Step 5 Let $t' = \lceil \frac{\bar{t}+\underline{t}}{2} \rceil$. Solve problem (11) with the restriction that

$$l_i = u_i = 0, \quad \text{if} \quad w_i > w_{j_{t'}}, \tag{13}$$

If the restricted problem (11) is infeasible, set $\underline{t} \leftarrow t'$ and return to Step 4. Otherwise, let the optimal solution of problem (11) be d^2 so that the minimum value is $\max\limits_{e_i \in E}\{w_i H(c_i, d_i^2)\}$, then go to Step 6.

Step 6 If $\sum_{e_i \in E} w_i H(c_i, d_i^2) \le M$, then set $\bar{t} \leftarrow t'$. Otherwise, set $\underline{t} \leftarrow t'$. Return to Step 4.

Theorem 4.1 *Algorithm* 5 *solves problem* (9) *with a time complexity* $O(n^4 \log m)$.

Proof If the algorithm stops at Steps 2 or 3, by Lemma 4.1, problem (9) is infeasible.

We next consider the case that problem (9) is feasible. We designate computations starting from Step 4 until switching back to the next Step 4 as one iteration, and prove that the algorithm can obtain the optimal solution of problem (9) by at most $\lceil \log m \rceil$ iterations. By the introduction before proposing the algorithm, we know that the optimal value of problem (9) is one of the $\bar{t} - \underline{t}$ distinct cost values in the initial interval $(w_{j_{\underline{t}}}, w_{j_{\bar{t}}}]$, where $\underline{t}$ and $\bar{t}$ are defined in Steps 2 and 3, i.e., before entering the first iteration. In the following we show that this conclusion is true for $w_{j_{\underline{t}}}$ and $w_{j_{\bar{t}}}$ obtained in any iteration. To prove this fact we should consider the following three cases, as in each of the three cases the search interval $(w_{j_{\underline{t}}}, w_{j_{\bar{t}}}]$ is reduced.

Case 1 The algorithm finds in Step 5 that problem (11) with the restriction (13) is infeasible. We show that in this case for every feasible solution d of (11), $\max\limits_{e_i \in E}\{w_i H(c_i, d_i)\} > w_{j_{t'}}$. In fact, if there exists a feasible solution d satisfying $\max\limits_{e_i \in E}\{w_i H(c_i, d_i)\} \le w_{j_{t'}}$, that is, for each $e_i \in E$, $w_i H(c_i, d_i) \le w_{j_{t'}}$, which implies that if $w_i > w_{jt'}$, then $d_i = c_i$. Thus d is a feasible solution of the restricted problem (11), a contradiction. Hence for every feasible solution d of (11), $\max\limits_{e_i \in E}\{w_i H(c_i, d_i)\} > w_{j_{t'}}$. It means that the optimal value of problem (9) is greater than $w_{j_{t'}}$, and hence must be in the interval $(w_{j_{t'}}, w_{j_{\bar{t}}}]$. In this case by Step 5,

we let the next $\underline{t}$ equal t', which guarantees that the optimal value of (9) is in the next search interval $(w_{j_{\underline{t}}}, w_{j_{\bar{t}}}]$.

Case 2 The restricted problem (11) has an optimal solution d^2 and $\max_{e_i \in E}\{w_i H(c_i, d_i^2)\} > M$. This means that for any d satisfying the constraints of problem (11), if it also meets condition (13), then we must have $\sum_{i=1}^{m} w_i H(c_i, d_i) > M$. As condition (13) is equivalent to that $\max_{e_i \in E}\{w_i H(c_i, d_i)\} \leq w_{j_{t'}}$, the above conclusion implies that for every feasible solution d to problem (9), we must have $\max_{e_i \in E}\{w_i H(c_i, d_i)\} > w_{j_{t'}}$. Therefore, the optimal value of (9) must be in the interval $(w_{j_{t'}}, w_{j_{\bar{t}}}]$. So, in the next interval $(w_{j_{\underline{t}}}, w_{j_{\bar{t}}}]$, we should let $\underline{t} = t'$, that is what we did in Step 6 in this case.

Case 3 The restricted problem (11) has an optimal solution d^2 and $\sum_{i=1}^{m} w_i H(c_i, d_i^2) \leq M$. This means that d^2 satisfies all constraints of problem (9), and $\max_{e_i \in E}\{w_i H(c_i, d_i^2)\} \leq w_{j_{t'}}$. So, the optimal value of (9) must be in the interval $(w_{j_{\underline{t}}}, w_{j_{t'}}]$, i.e., in the next interval $(w_{j_{\underline{t}}}, w_{j_{\bar{t}}}]$, $\bar{t} = t'$, see Step 6.

Combining the above three cases, we conclude that in each iteration the optimal value of the problem (9) is one of the $\bar{t} - \underline{t}$ distinct cost values in the interval $(w_{j_{\underline{t}}}, w_{j_{\bar{t}}}]$.

As we know, the bisection method guarantees that after at most $\lceil \log m \rceil$ iterations, the search interval $(w_{j_{\underline{t}}}, w_{j_{\bar{t}}}]$ must satisfy $\bar{t} - \underline{t} = 1$, i.e., there is only one cost value in the interval $(w_{j_{\underline{t}}}, w_{j_{\bar{t}}}]$, which is just $w_{j_{\bar{t}}}$. The corresponding solution is d^2 which is feasible. So, d^2 is an optimal solution of problem (9), and the optimal value is $w_{j_{\bar{t}}}$. So, the validity of the algorithm is proved.

Finally, we study the time complexity of Algorithm 5. It is clear that Step 1 takes $O(m \log m)$ time; from Sect. 3 we know that Step 2 takes $O(n^4)$ time; from [3] we know Step 3 takes $O(n^4)$ and Steps 4–6 take $O(n^4)$ time. As the algorithm iterates for at most $\lceil \log m \rceil$ times, it runs in $O((m + n^4) \cdot \log m) = O(n^4 \cdot \log m)$ time in the worst-case, and hence is a strongly polynomial time algorithm. □

Now let us consider the problem (10). By comparing problems (11) and (10), we see that they have the same objective function and the first two groups of constraints, but problem (10) has one more constraint: $\max_{e_i \in E}\{w_i H(c_i, d_i)\} \leq M$. This constraint means $d_i = c_i$ if $w_i > M$. It is very simple to realize this purpose by redefining the v_i in formula (7) as follows:

$$v_i = \begin{cases} w_i, & \text{if } e_i \in E(p), \quad c_i + u_i \geq p \text{ and } w_i \leq M, \\ W + 1, & \text{otherwise}, \end{cases} \tag{14}$$

Then by a similar argument as in [3], Algorithm 4 can be applied to solve the above problem (10).

5 Concluding remarks

In this paper we studied the inverse min–max spanning tree problem under the weighted Hamming distance. For bottleneck-type objective functions, we presented strongly polynomial algorithm to solve it. Furthermore, for two mixed type problems which include both sum type and bottleneck type measurements, we also give strongly polynomial algorithms.

As a future research topic, it will be meaningful to consider other inverse combinational optimization problems under Hamming distance. Studying computational complexity results and proposing optimal/approximation algorithms are promising.

References

1. Camerini, P.M.: The min–max spanning tree problem and some extensions. Inform. Process. Lett. **7**, 10–14 (1978)
2. Yang, X.G., Zhang, J.Z.: Some inverse min–max network problems under weighted l_1 and l_∞ norms with bound constraints on changes. J. Comb. Optim. **13**, 123–135 (2007)
3. Liu, L.C., Yao, E.Y.: Inverse min–max spanning tree problem under the weighted sum-type Hamming distance. In: Theor Comput Sci (2007). doi: 10.1016/j.tcs.2007.12.006
4. Heuberger, C.: Inverse Optimization: A survey on problems, methods, and results. J. Comb. Optim. **8**, 329–361 (2004)
5. He, Y., Zhang, B., Yao, E.: Wighted inverse minimum spanning tree problems under Hamming distance. J. Comb. Optim. **9**, 91–100 (2005)
6. He, Y., Zhang, B., Zhang, J.: Constrained inverse minimum spanning tree problems under the bottleneck-type Hamming distance. J. Glob. Optim. **34**(3), 467–474 (2006)
7. Duin, C.W., Volgenant, A.: Some inverse optimization problems under the Hamming distance. Eur. J. Oper. Res. **170**, 887–899 (2006)
8. Zhang, B., Zhang, J., He, Y.: The center location improvement problem under the Hamming distance. J. Comb. Optim. **9**, 187–198 (2005)
9. Yang, X.G., Zhang, J.Z.: Some new results on inverse sorting problems. In: Lecture Notes in Computer Science, vol. 3595, pp. 985–992 (2005)
10. Liu, L.C., Zhang, J.Z.: Inverse maximum flow problems under the weighted Hamming distance. J. Comb. Optim. **12**, 395–408 (2006)
11. Liu, L.C., Yao, E.Y.: Weighted inverse minimum cut problem under the bottleneck-type Hamming distance. Asia-Pac. J. Oper. Res. **24**(5), 1–12 (2007)
12. Guan, X., Zhang, J.: Inverse Bottleneck Optimization Problems under Weighted Hamming Distance. In: Lecture Notes in Computer Science, vol. 4041, pp. 220–230 (2006)
13. Schrijver, A.: Combinatorial Optimization, Polyhedra and Efficiency. Springer-Verlag, Berlin (2003)